INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – CÂMPUS CUBATÃO

GEOVANA ALVES DA CONCEIÇÃO

GUSTAVO HENRIQUE DA SILVA MELO

JULIA SANTOS AMORIM

MARIA CLARA GENEROSO DA SILVA

PEDRO HENRIQUE DE ALMEIDA ARAÚJO

VICTOR RAPHAEL RIBEIRO

YASMIN SANTOS SILVA

LINGUAGEM GROOVY

CUBATÃO/SP
2021

GEOVANA ALVES DA CONCEIÇÃO
GUSTAVO HENRIQUE DA SILVA MELO
JULIA SANTOS AMORIM
MARIA CLARA GENEROSO DA SILVA
PEDRO HENRIQUE DE ALMEIDA ARAÚJO
VICTOR RAPHAEL RIBEIRO
YASMIN SANTOS SILVA
CTII 418

LINGUAGEM GROOVY

Apostila apresentada à disciplina de Projeto de Sistemas (PJS) junto ao Curso Técnico em Informática Integrado ao Ensino Médio do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP).

Disciplina: Prática de Projeto de Sistemas.
Docente: Maurício Neves Asenjo.

CUBATÃO/SP
2021

SUMÁRIO

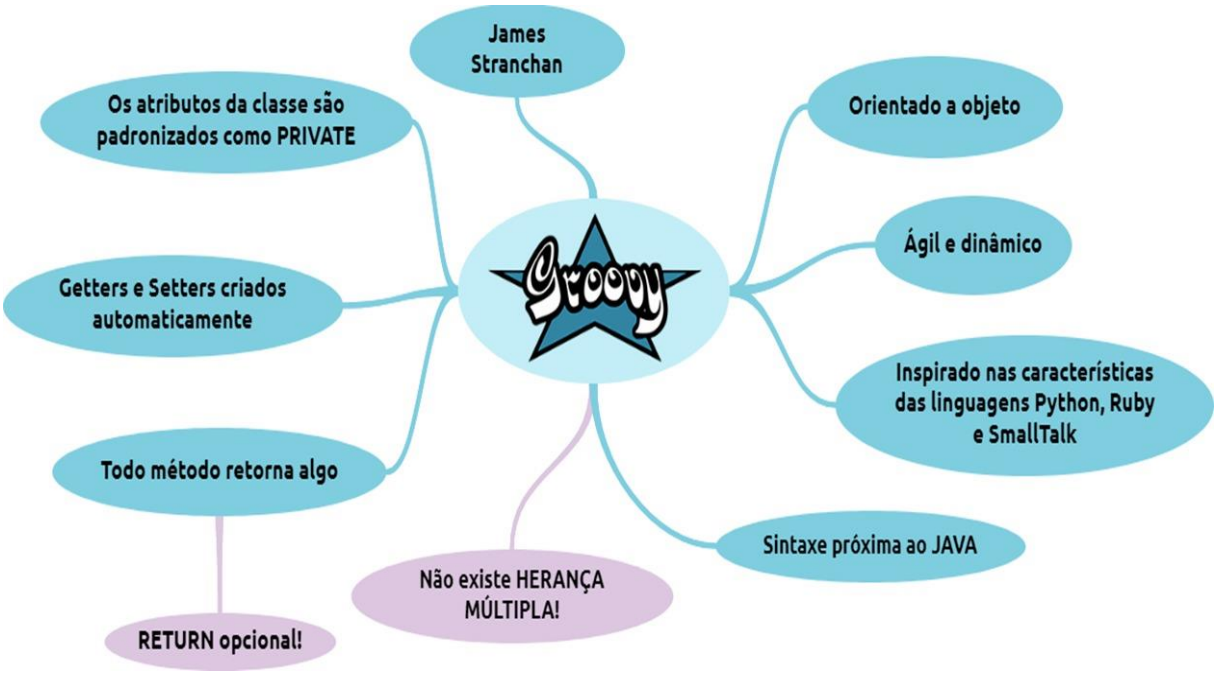
1. INTRODUÇÃO

Tudo começou com um plano de criação de uma linguagem a qual James Strachan, Engenheiro de Software, no dia 29 de agosto de 2003, pensava em desenvolver. “Minha ideia inicial é fazer uma linguagem Dinâmica, que seja compilada diretamente em classes Java e que tenha toda a produtividade e elegância encontrada em Ruby e Python.” (STRACHAN, 2003).

A partir desta ideia, James aliou-se com Bob McWhirter, e juntos, ainda em 2003, desenvolveram o projeto Groovy. Este, no que lhe concerne, trata-se de uma linguagem de programação 100% orientada a objeto, projetada para compilar nas aplicações da Java Virtual Machine (JVM), que possui sintaxe simples, agilidade e dinamicidade, não faz herança múltipla, *getters* e *setters* são criados automaticamente e ficam transparentes na classe com os quais foram criados, deixando o código muito mais limpo quando se compara ao modo tradicional do Java e tendo a possibilidade de escrevê-los reduzidamente, algumas bibliotecas são importadas de forma automática ao projeto, não precisando declarar a importação delas, as *strings* são tratadas em duas classes distintas - *String* e *GStrings* - as quais podem ser delimitadas por aspas duplas ou simples, ocupar uma ou várias linhas, inserir diretamente dentro delas operações aritméticas e variáveis, além disso, para satisfação de muitos programadores, o ponto e vírgula e os parênteses, no final de cada linha codificada, são optativos. Vale enfatizar também que quando uma linguagem de programação utiliza as ferramentas Java, ela pode ser executada no Windows, Linux, Mac, porque a JVM realiza as interpretações necessárias às execuções dos programas, facilitando a portabilidade, haja vista que o Java permite que um mesmo programa seja executado em vários sistemas operacionais, gerando códigos genéricos. Por causa dessa ligação direta ao Java, desenvolvedores conseguem se adequar com maior facilidade, afinal de contas, uma aplicação Java válida pode ser usada como aplicação Groovy, pela mesma sintaxe, portanto, se houver a criação de um objeto .jar, na sequência, uma alteração para .groovy, ele será reconhecido pelo Groovy.

Na atualidade, o Groovy encontra-se em um crescimento significativo. Tal afirmação é legitimada de acordo com os dados do TIOBE Index, um coletor de acesso das pesquisas de linguagens de programação na internet, onde ele ocupa a posição décima quarta dentro de uma lista com mais de 200 linguagens. Devido à riqueza de recursos, essa crescente opção de programação foi vista no âmbito computacional de algumas empresas fortemente conhecidas, a saber: Netflix, Android, MasterCard, Vodafone, Mutual of Omaha, Instituo Nacional do Câncer.

FIGURA 1 – MAPA MENTAL DA LINGUAGEM GROOVY



Fonte: Própria (2021)

2. IDEs E INSTALAÇÕES NECESSÁRIAS

O conhecido termo IDE é utilizado para referenciar os ambientes de desenvolvimento integrado necessários para o desenvolvimento em qualquer linguagem. É possível encontrar algumas opções de IDE para desenvolvimento na linguagem Groovy entre as mais populares está o Visual Studio Code amplamente utilizado para programação em diferentes linguagens, mas com enfoque em programação web, apresenta a possibilidade de uso de diversos plugins, o Visual Studio Code é considerado um editor de texto, pois apresenta menos recursos se comparado a uma IDE propriamente dita, outro editor de texto disponível é o Sublime Text.

O Eclipse é uma IDE completíssima que permite o desenvolvimento em diversas linguagens, inclusive o Groovy, além de diversas ferramentas que auxiliam no processo de desenvolvimento. Estes são apenas alguns exemplos de uma infinidade de opções, como o Intellij IDEA.

2.1. IDE ESCOLHIDA

Entre as IDE´s apresentadas escolheu-se para trabalhar o ambiente de desenvolvimento o Eclipse 2021-03. Ele é um ambiente gratuito, pode ser desenvolvido nos sistemas operacionais Windows, Mac, Linux e Solaris e possui código aberto. A última qualidade possibilita o desenvolvimento de software nas linguagens Java, JavaScript, HTML5, PHP, C/C++, Groovy, Ruby.

A plataforma conta com algumas ferramentas que facilitam o entendimento, melhoram a sintaxe e auxiliam no momento da criação da sintaxe de programação, sendo elas syntax highlighting, code completion e refactoring.

A ferramenta realce de sintaxe, ou melhor “syntax highlighting”, em inglês; é muito utilizada para facilitar a leitura e compreensão do código em análise.

O processo sistemático de reestruturação de código, “refactoring”, prove um código limpo sem alterar sua estrutura externa. Essa ferramenta faz parte do dia a dia do programador, pois quando o “software” necessita de atualizações ou correções de alguns problemas ele é fundamental, uma vez que antes de adicionar a atualização é necessário verificar se a sintaxe está estruturada para recebe-la, caso não esteja reestruturamos o código atual afim de facilitar a adição da atualização. Após este processo reestruturamos novamente dessa vez com o intuito o código claro e limpo para uma consulta posterior.

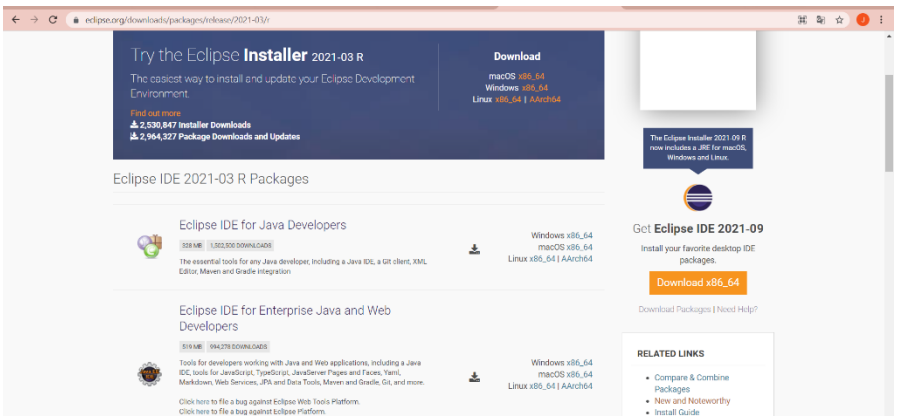
O auto completar de códigos é uma ferramenta que auxilia os programadores no processo de criação da sintaxe no intuito de completar os nomes dos campos, métodos, classes e até mesmo palavras-chaves.

O Eclipse IDE também é utilizado para o desenvolvimento de aplicativos para web, empresas, desktops e dispositivos móveis, além de todas as qualidades apresentadas o grupo está habituado a programar nesta plataforma.

2.1.1. DOWNLOAD E INSTALAÇÃO DA IDE - ECLIPSE 2021-03

Entre no link a seguir: <https://www.eclipse.org/downloads/packages/release/2021-03/r>.

FIGURA 2 – Download IDE: Passo 1



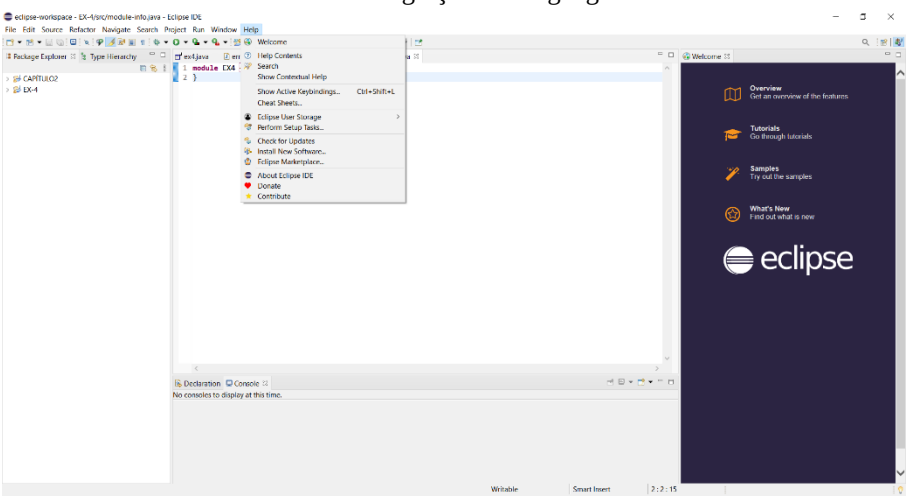
Fonte: Própria (2021)

Aceite os termos e a licença do Eclipse Clique no download de acordo com a configuração da sua máquina.

2.1.2. INTEGRAÇÃO DA LINGUAGEM NA IDE

Clique em “Help”/”Ajuda” na barra superior.

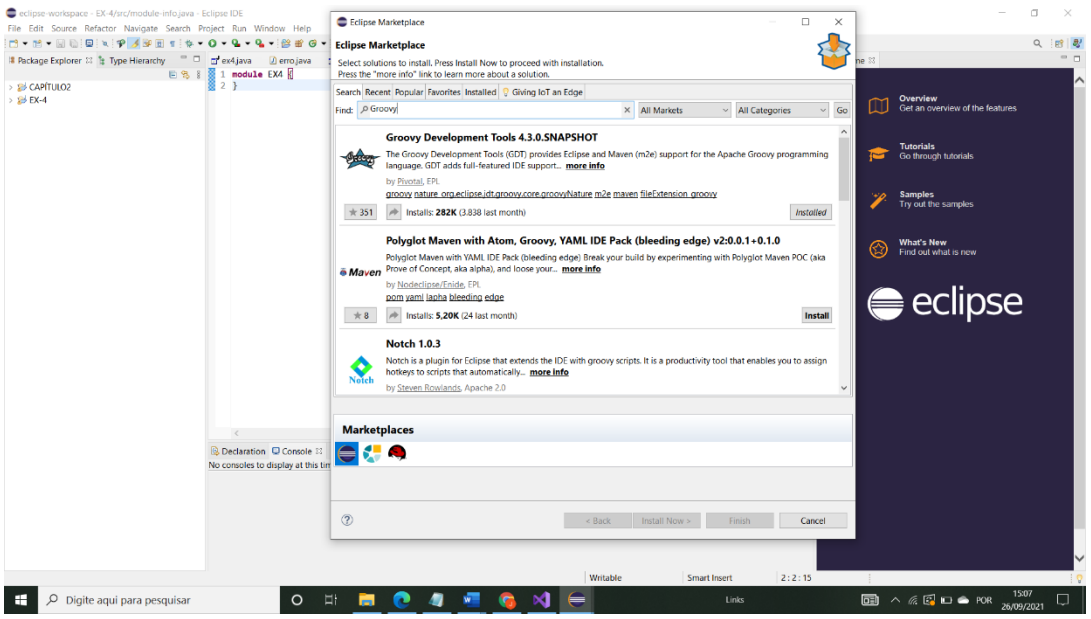
FIGURA 3 – Integração da Linguagem na IDE: Passo 1



Fonte: Própria (2021)

Na opção “Eclipse Marketplace”:

FIGURA 4 – Integração da Linguagem na IDE: Passo 2

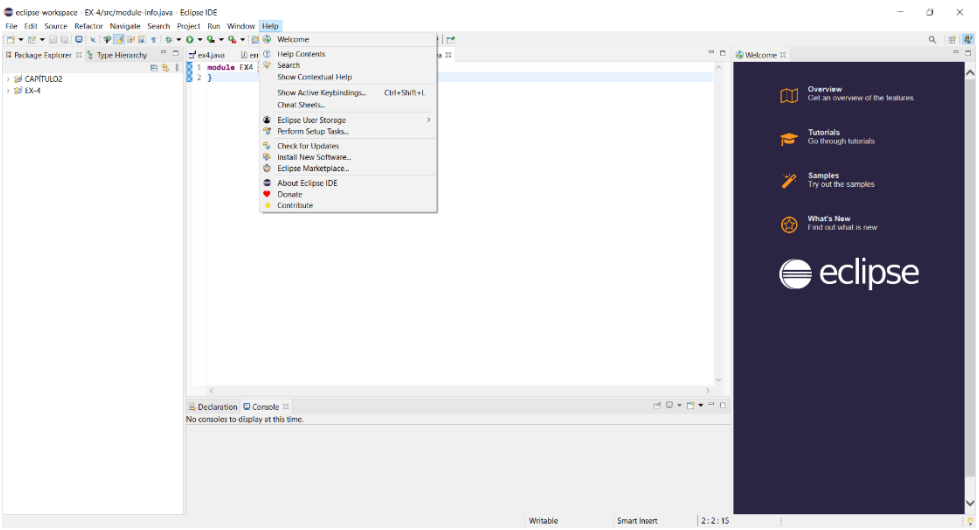


Fonte: Própria (2021)

Pesquise por “Groovy” e instale o pacote “Groovy Development Tools 4.3.0.SNAPSHOT”.

Após a instalação, retorne em em “Help”/”Ajuda” na barra superior.

FIGURA 5 – Integração da Linguagem na IDE: Passo 3

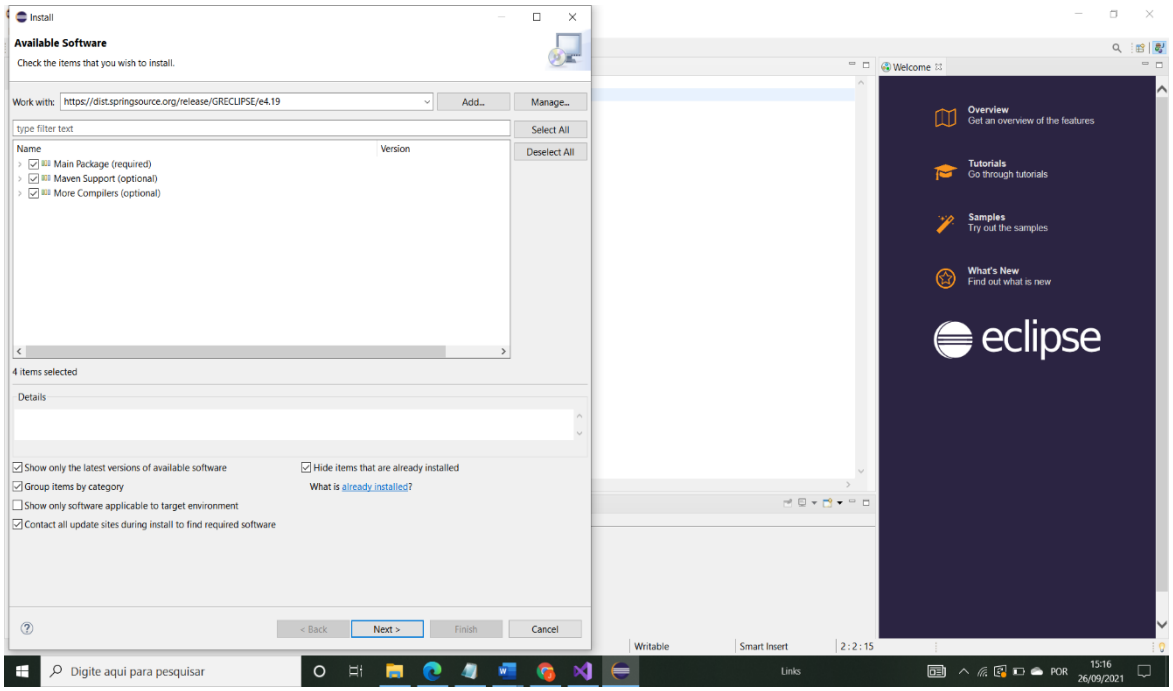


Fonte: Própria (2021)

Clique na opção “Install a new software” / “Instalar um novo software”.

FIGURA 6 – Integração da Linguagem na IDE: Passo 4

Cole o link a seguir: https://dist.springsource.org/release/GRECLIPSE/e4.19 e selecione as três opções como na imagem abaixo. Em seguida clique em “Next”/”Próximo” e finalize a instalação.

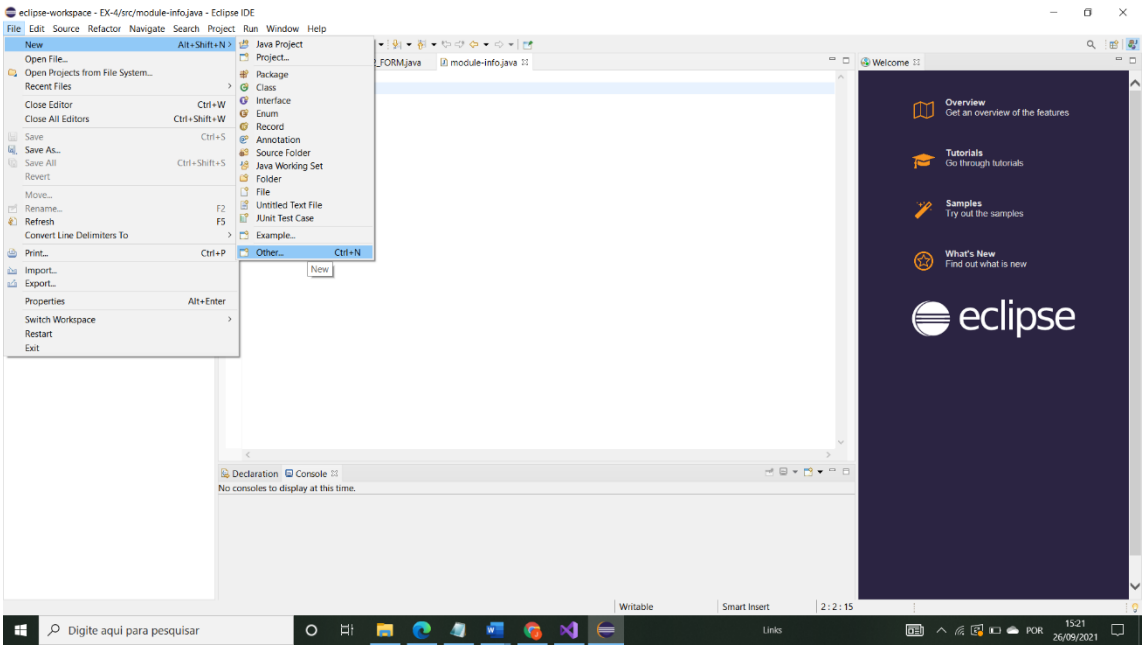


Fonte: Própria (2021)

3. DESENVOLVIMENTO DE UM "HELLO WORLD"

Clique em novo projeto.

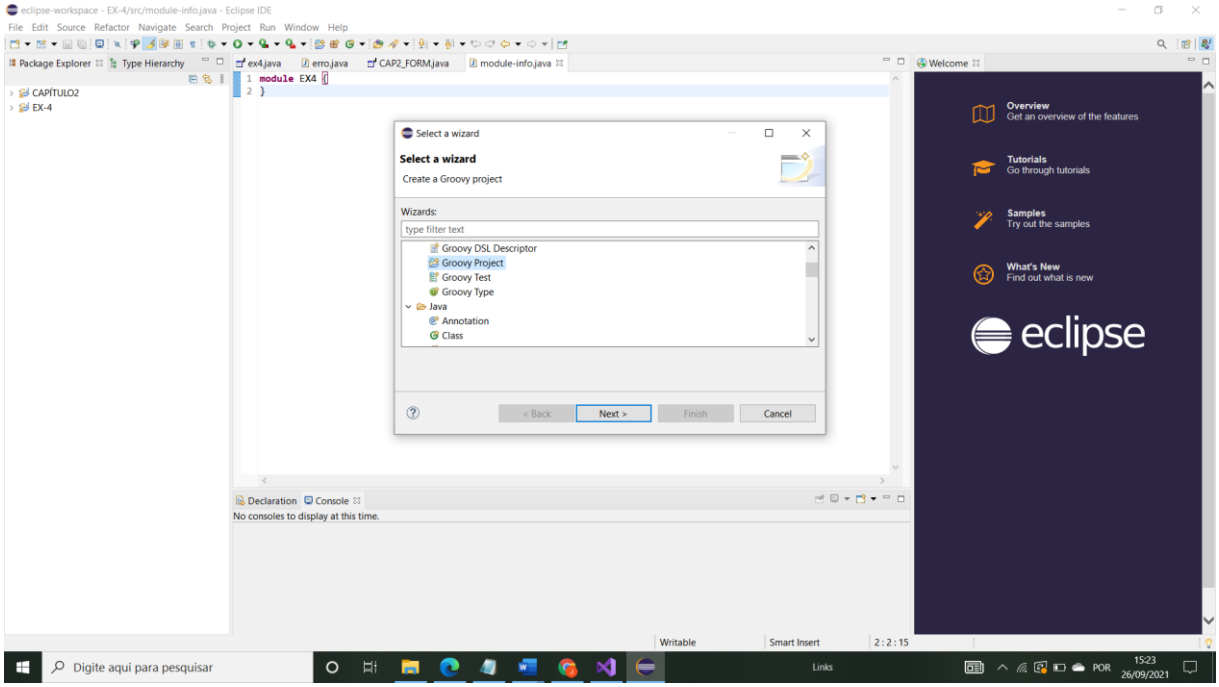
FIGURA 7– "Hello Word": Passo 1



Fonte: Própria (2021)

Clique em File, em seguida em "New"/"Novo" e por fim em "Other"/"Outros".

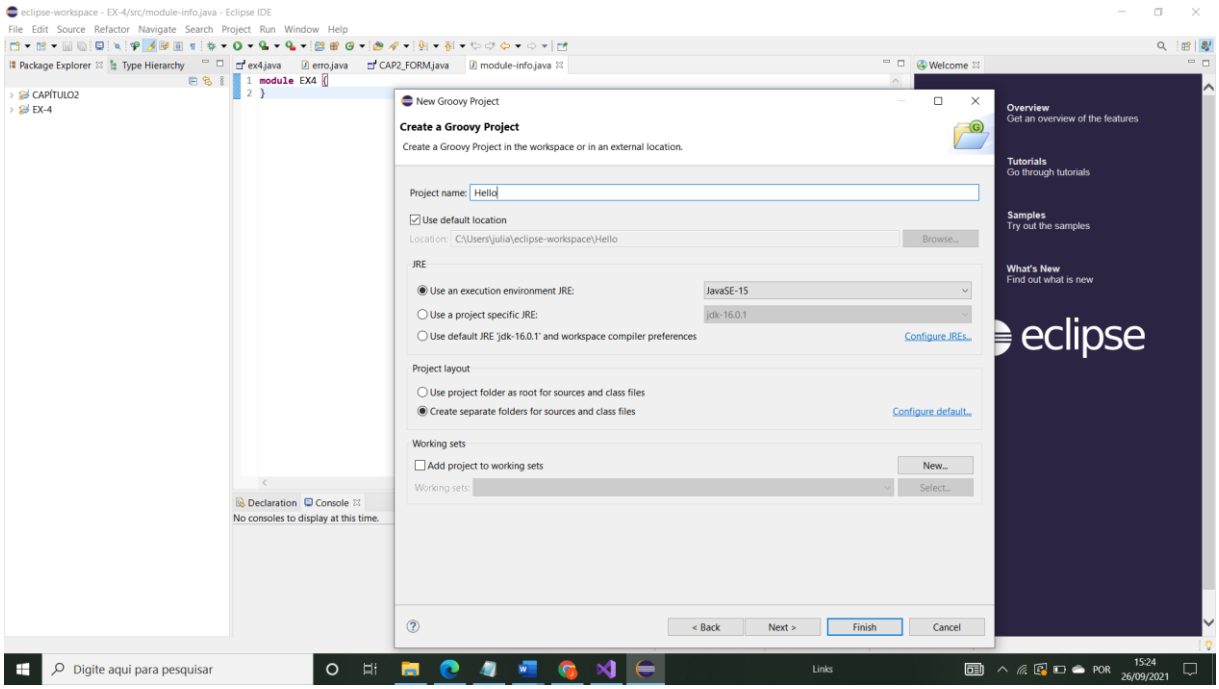
FIGURA 8 – “Hello Word”: Passo 1



Fonte: Própria (2021)

Selecione “Groovy Project” e clique em “Next”/”Próximo”.

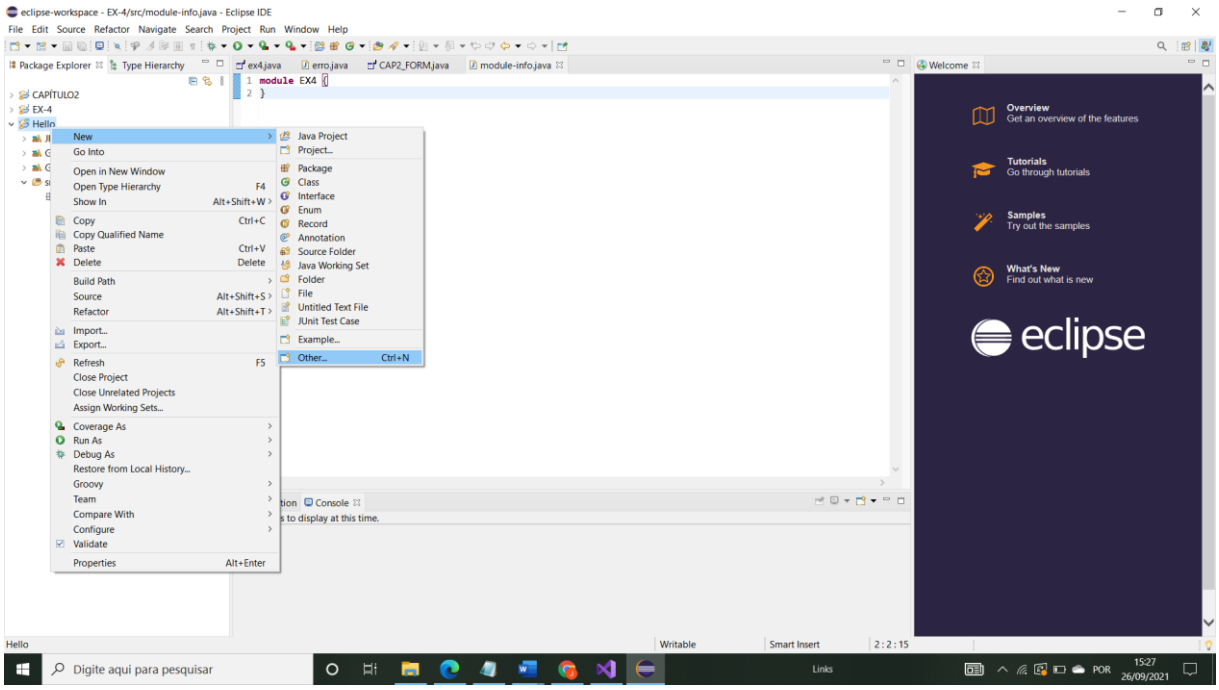
FIGURA 9 – “Hello Word”: Passo 1



Fonte: Própria (2021)

Atribua um nome ao seu projeto groovy e click em “finish”/”Finalizar”.

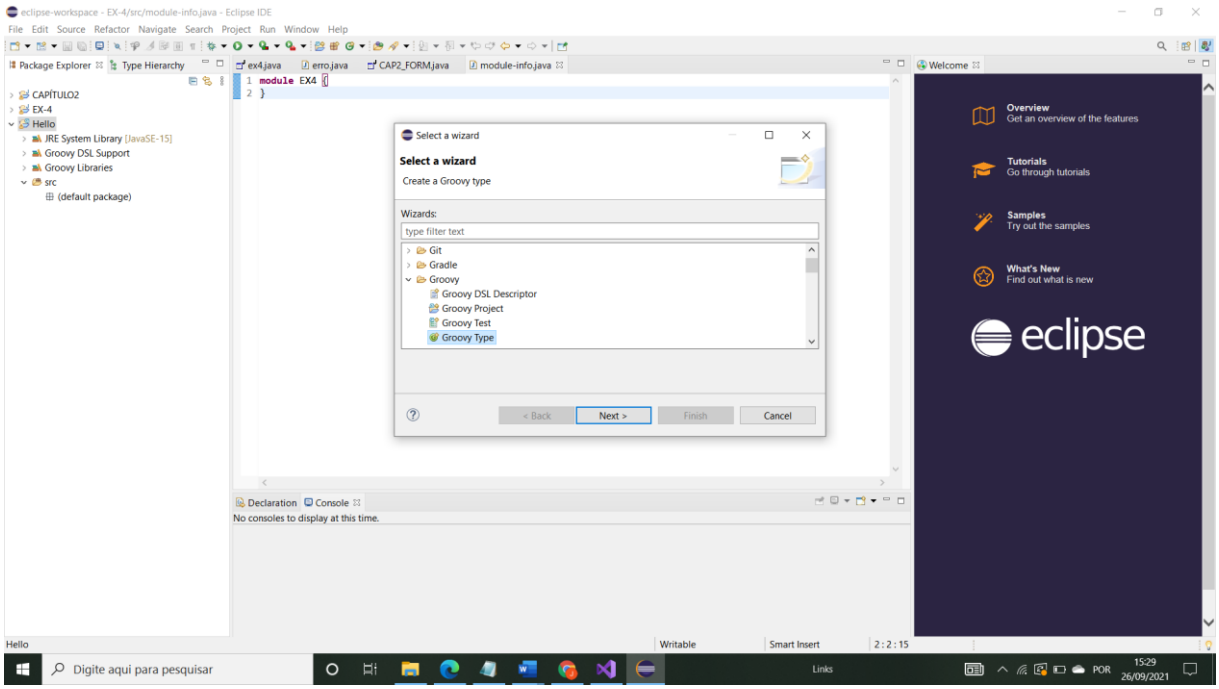
FIGURA 10 – “Hello Word”: Passo 2



Fonte: Própria (2021)

Clique com o botão direito do mouse no projeto e em seguida em “New”/”Novo”. Após isso clique em “Other”/”Outros”.

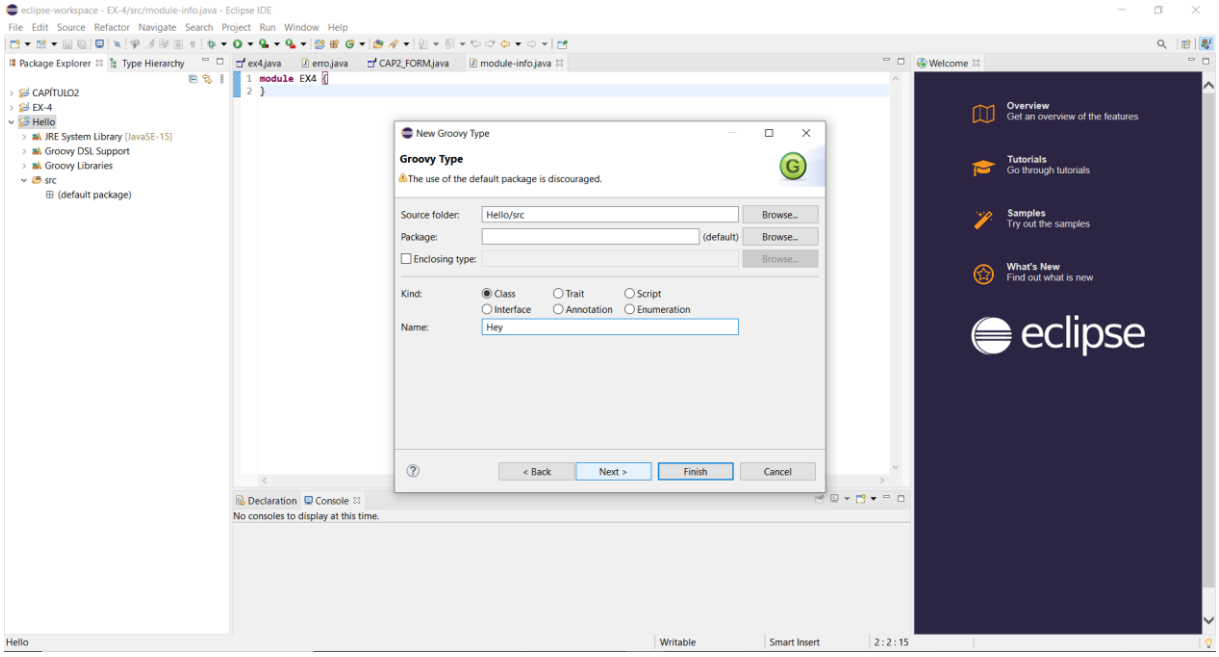
FIGURA 11 – “Hello Word”: Passo 3



Fonte: Própria (2021)

Escolha agora a opção “Groovy Type” e clique em “Next”/”Próximo”.

FIGURA 12 – “Hello Word”: Passo 4

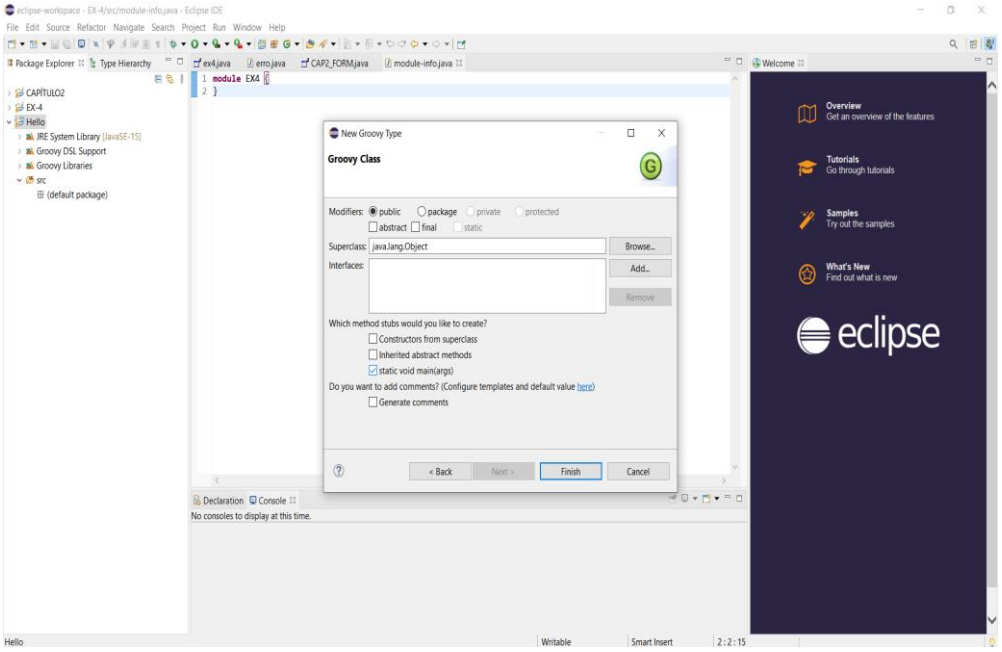


Fonte: Própria (2021)

Atribua um nome ao arquivo groovy e clique em “Next”/”Próximo”.

Em seguida, selecione a opção “static void main(args)” como na imagem a seguir:

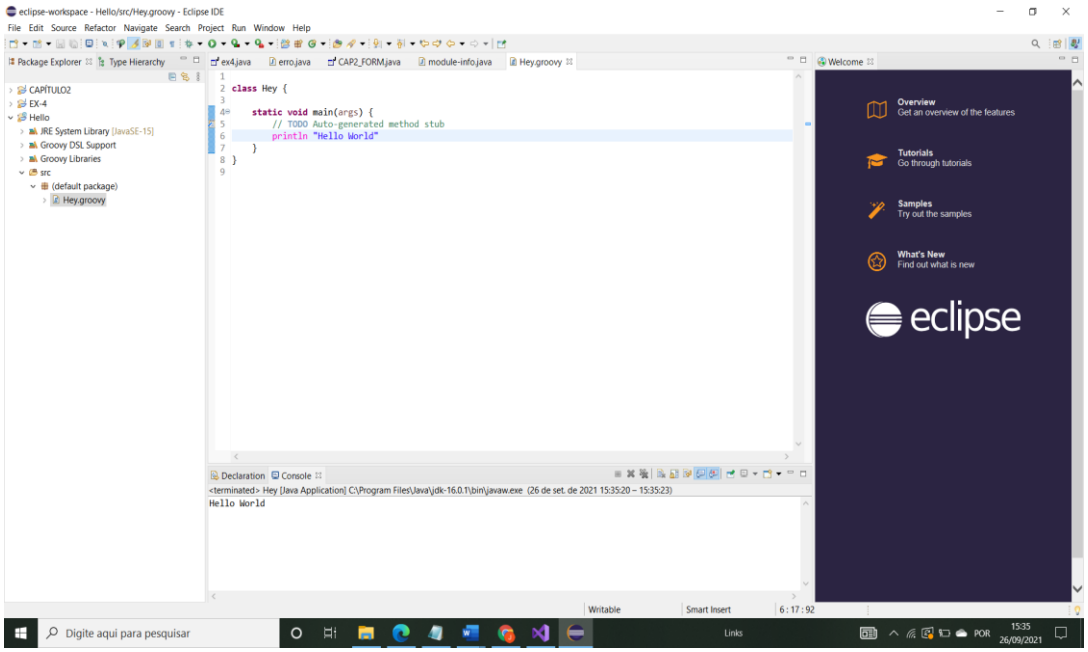
FIGURA 13 – “Hello Word”: Passo 5



Fonte: Própria (2021)

Com a classe aberto, vamos criar o Hello World!

FIGURA 14 – “Hello Word”: Passo 6



Fonte: Própria (2021)

Digite o comando para a saída da mensagem Hello World.

4. EXERCÍCIOS BÁSICOS

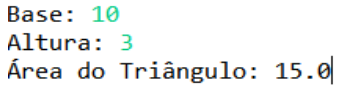
Por meio de uma seleção bastante criteriosa acerca de vários exercícios relacionados a aprendizagem básica de qualquer linguagem de programação, nessa parte, 16 exercícios foram desenvolvidos no intuito de fornecer a você, caro leitor, um entendimento sobre declaração de variáveis, comandos de entrada e saída, funções matemáticas, decisão lógica, construção de laços de repetição, manuseamento dos operadores e das variáveis indexadas, isto é, vetores e matrizes, entre outros conceitos essenciais na linguagem Groovy.

1 - A partir da digitação da base e altura de um triângulo o programa deverá calcular sua área e exibi-la no monitor.

```
class AreadoTriangulo {
    static void main(args) {

        //Entrada dos dados pelo usuário:
        print "Base: "
        //Leitura da entrada de dados e atribuição da variável b (BASE):
           def b = System.in.newReader().readLine().toDouble()

        print "Altura: "
        //Leitura da entrada de dados e atribuição da variável h (ALTURA):
           def h = System.in.newReader().readLine().toDouble()

        //Declaração da variável areaT (Área total):
           def areaT
        //Equação da área do triângulo:
        areaT = (b * h) / 2

        //Exibição do resultado final após o cálculo:
        print "Área do Triângulo: " + areaT
    }
}
```

Após clicar em executar, o programa perguntará a você qual a base e a altura, ou seja, a entrada de dados, que, de acordo com a imagem abaixo, é 10 e 3, respectivamente. Quando fornecidas essas informações, clicando em *Enter*, mostrar-se-á o resultado da área desse triângulo, uma vez que a fórmula do calcula foi programada anteriormente. Note que as variáveis foram definidas através do comando *def*, localmente, ele substitui o tipo (*String*, *Int*, *Double*, etc), de modo que esse tipo seja reconhecido automaticamente ao longo do programa.

Figura 15 – Resposta da área total



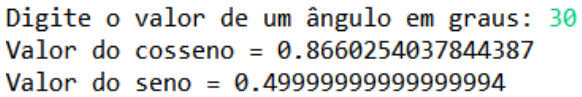
Fonte: Própria (2021)

2 - O programa deverá pedir para você digitar o valor de um ângulo em graus e na sequência mostrar o valor do seno e cosseno deste ângulo.

```
class Angulos {
        static void main(args) {

       //Entrada dos dados pelo usuário:
       print "Digite o valor de um ângulo em graus: "
       //Leitura da entrada de dados e atribuição da variável angulo (ângulo):
          def angulo = System.in.newReader().readLine().toDouble()

        //Fórmula para converter um número em radianos:
        angulo = Math.toRadians(angulo)

        //Exibição do resultado em seno e cosseno após a conversão:
        println "Valor do seno = " + Math.sin(angulo)
            println "Valor do cosseno = " + Math.cos(angulo)
    }
}
```

Repare que os pontos e vírgulas não são usados, o que facilita no desenvolvimento dos programas. Talvez, você tenha pensado que seria necessário fazer uma fórmula para transformar os números em senos e cossenos, no entanto, no programa, usou-se apenas uma função matemática chamada *Math*, ela, por sua vez, faz inúmeras funções da matemática, seja elevar um número ao quadro, seja transformá-lo em seno e cosseno (*sin* e *cos*), conforme se vê nos códigos acima. Na exibição do programa, entrou-se com o valor 30. Veja os resultados obtidos:

Figura 16 – Resposta dos ângulos



Fonte: Própria (2021)

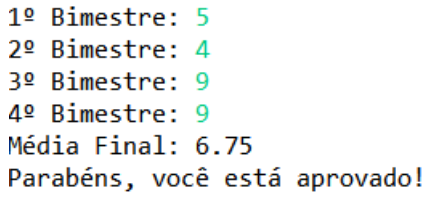
3 - O programa deverá solicitar a digitação de suas 4 notas bimestrais, feito isso deverá calcular e exibir a sua média final (média aritmética entre as 4 notas). Feito isso deverá também mostrar as mensagens: "Você está aprovado!", "Você está reprovado!" ou "Você está de exame" de acordo com o seguinte critério:

```
class Boletim {
        static void main(args) {

        //Entrada das notas do primeiro ao quarto bimestre pelo usuário:
        print "1° Bimestre: "
        //Leitura da entrada de dados e atribuição da variável B1 (BIMESTRE 1):
        def B1 = System.in.newReader().readLine().toDouble()

        print "2° Bimestre: "
        //Leitura da entrada de dados e atribuição da variável B2 (BIMESTRE 2):
        def B2 = System.in.newReader().readLine().toDouble()

        print "3° Bimestre: "
        //Leitura da entrada de dados e atribuição da variável B3 (BIMESTRE 3):
        def B3 = System.in.newReader().readLine().toDouble()

        print "4° Bimestre: "
        //Leitura da entrada de dados e atribuição da variável B4 (BIMESTRE 4):
        def B4 = System.in.newReader().readLine().toDouble()

        //Equação da média final dos bimestres:
        def mediaFinal = (B1 + B2 + B3 + B4) / 4

        //Exibição da média final após o cálculo:
        println "Média Final: " + mediaFinal

        //Decisão lógica - if e else - desta forma, se a nota for igual ou superior a 6, o aluno
será aprovado.
        if(mediaFinal >= 6) {
            print "Parabéns, você está aprovado!"
        //Decisão lógica - if e else - desta forma, se a nota for inferior a 3, o aluno será
reprovado.
        } else if (mediaFinal < 3) {
                print "Infelizmente, você está reprovado!"
        //Decisão lógica - if e else - desta forma, se a nota for diferente das condições acima,
o aluno estará de exame.
         } else
                print "Você não atingiu a média, sendo assim, ficará de exame. Aguarde a
data da prova!"
      }
}
```

Pode-se observar, neste programa, o uso de decisão lógica, isto é, um modo de programação baseado em fornecer algumas condições de acordo com algumas informações predefinidas. Usou-se, então, o *if* e *else*, que são, quando traduzidos, o “se” e o “senão”. Note que se a média final do aluno for maior que 6, ele estará aprovado, por exemplo. Na equação

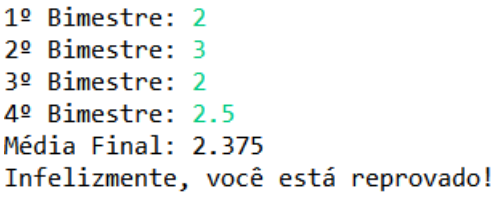
da média, há uma somatória das notas dentro de parênteses, isto se deve à regra da própria matemática.

Figura 17 – Aluno aprovado



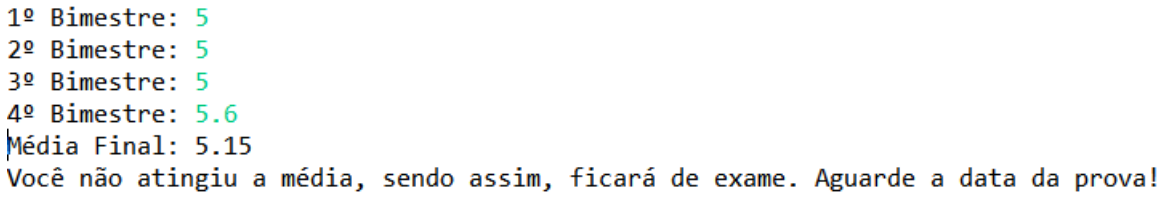
Fonte: Própria (2021)

Figura 18 – Aluno reprovado



Fonte: Própria (2021)

Figura 19 – Aluno em exame



Fonte: Própria (2021)

4 - O programa deverá nos solicitar a digitação de dois números e um caractere, sendo que este poderá ser "+", "-", "*" ou "/". Mediante o caractere digitado fazer o respectivo cálculo e exibir o resultado, se o caractere não corresponder a nenhum dos 4 caracteres em questão exibir mensagem de erro. Este programa obrigatoriamente deverá ser feito usando um ninho de ifs.

A classe *Scanner*, que será utilizada neste exercício em alguns daqui para frente, tem por objetivo separar a entrada dos textos em blocos, gerando os conhecidos *tokens*, os quais são sequências de caracteres separados por delimitadores que por padrão correspondem aos espaços em branco, tabulações e mudança de linha. Com essa classe podem ser convertidos textos para tipos primitivos, sendo que esses textos podem ser considerados como objetos do tipo String, *InputStream* e arquivos. Em suma, o uso desse recurso, no Groovy, facilita o trabalho de leitura das atribuições das variáveis. Note que os outros recursos usados já são de seu conhecimento.

```
class Ex4 {
static void main(args) {

//Definição do scanner que irá receber os algarismos
Scanner sc1 = new Scanner(System.in)
// Definição das variáveis que armazenará o cálculo escolhido
double soma=0, sub=0, mult=0, div=0

//Entrada de dados pelo usuário e armazenamento do dado na variável "a"
print "Digite o primeiro algarismo: "
double a = sc1.nextDouble()

//Entrada de dados pelo usuário e armazenamento do dado na variável "b"
print "Digite o segundo algarismo: "
double b = sc1.nextDouble()

//Entrada do caractere desejado
println "Entre com o caractere desejado para realização do cálculo (+, -, *, /)"
// Leitura do caractere escolhido e armazenando em uma string para fazer a comparação
Scanner scanner = new Scanner(System.in,"ISO-8859-1")
String op = scanner.nextLine()

/*Se o caractere escolhido for o "+", ele entrará neste if, executará o cálculo depois o resultado será exibido na tela*/
        if (op == "+"){
                soma = a + b
                print "Resultado do cálculo: " + soma
        }
        /*Se o carctere escolhido for o "-", ele entrará neste if, executará o cálculo depois o resultado será exibido na tela*/
        else if (op == "-"){
                sub = a - b
                print "Resultado do cálculo: " + sub
        }
        /*Se o carctere escolhido for o "*", ele entrará neste if, executará o cálculo depois o resultado será exibido na tela*/
        else if (op == "*") {
                mult = a * b
                print "Resultado do cálculo: " + mult
        }
        /*Se o carctere escolhido for o "/", ele entrará neste if, executará o cálculo depois o resultado será exibido na tela*/
        else if(op == "/") {
                div = a / b
                print "Resultado do cálculo: " + div
        }
        /*Caso não seja nenhuma das opções acima ele exibirá uma mensagem de erro*/
        else {
                print "Caractere não encontrado. Digite uma das operações seguintes(+, -, *, /)"
      }
   }
}
```

De acordo com cada escolha do usuário, ele entrará com os dois valores e escolherá a operação matemática a ser seguida. Repare cada uma delas abaixo.

Figura 20 – Opção somar

```
Digite o primeiro algarismo: 4
Digite o segundo algarismo: 8
Entre com o caractere desejado para realização do cálculo (+, -, *, /)
+
Resultado do cálculo: 12.0
```

Fonte: Própria (2021)

Figura 21 – Opção subtrair

```
Digite o primeiro algarismo: 4
Digite o segundo algarismo: 8
Entre com o caractere desejado para realização do cálculo (+, -, *, /)
-
Resultado do cálculo: -4.0
```

Fonte: Própria (2021)

Figura 22 – Opção multiplicar

```
Digite o primeiro algarismo: 4
Digite o segundo algarismo: 8
Entre com o caractere desejado para realização do cálculo (+, -, *, /)
*
Resultado do cálculo: 32.0
```

Fonte: Própria (2021)

Figura 23 – Opção dividir

```
Digite o primeiro algarismo: 4
Digite o segundo algarismo: 8
Entre com o caractere desejado para realização do cálculo (+, -, *, /)
/
Resultado do cálculo: 0.5
```

Fonte: Própria (2021)

As decisões lógicas também facilitam o tratamento de alguns possíveis erros durante as digitações dos usuários. Na situação abaixo, note que foi digitado um operador inexistente, assim, por meio da programação, elaborou-se uma forma de solucionar esse problema, que é solicitar ao usuário uma nova digitação até que ela esteja de acordo com o programa.

Figura 24 – Opção incorreta

```
Digite o primeiro algarismo: 9
Digite o segundo algarismo: 8
Entre com o caractere desejado para realização do cálculo (+, -, *, /)
gh
Caractere não encontrado. Digite uma das operações seguites(+, -, *, /)
```

Fonte: Própria (2021)

5 - Idem ao exercício anterior, porém agora a solução deverá ser desenvolvida usando um switch-case.

```
class Ex5 {
static void main(args) {

//Definição do scanner que irá receber os algarismos
Scanner sc1 = new Scanner(System.in)
// Definição das variáveis que armazenará o cálculo escolhido
double soma=0, sub=0, mult=0, div=0

//Entrada de dados pelo usuário e armazenamento do dado na variável "a"
print "Digite o primeiro algarismo: "
double a = sc1.nextDouble()

//Entrada de dados pelo usuário e armazenamento do dado na variável "b"
print "Digite o segundo algarismo: "
double b = sc1.nextDouble()

//Entrada do caractere desejado
println "Entre com o caractere desejado para realização do cálculo (+, -, *, /)"
// Leitura de caractere escolhido e armazenando em uma string para fazer a comparação
Scanner scanner = new Scanner(System.in,"ISO-8859-1")
String op = scanner.nextLine()

//Esse switch irá comprar a variável "op" com todas as opções disponíveis
switch (op) {
/*Se o caractere escolhido for o "+", ele entrará neste if, executará o cálculo depois o resultado será
exibido na tela*/
     case "+":
     soma = a + b
     print "Resultado do cálculo: " + soma
     break

/*Se o caractere escolhido for o "-", ele entrará neste if, executará o cálculo depois o resultado será
exibido na tela*/
     case "-":
     sub = a - b
     print "Resultado do cálculo: " + sub
     break

/*Se o caractere escolhido for o "*", ele entrará neste if, executará o cálculo depois o resultado será
exibido na tela*/
     case "*":
     mult = a * b
     print "Resultado do cálculo: " + mult
     break

/*Se o caractere escolhido for o "/", ele entrará neste if, executará o cálculo depois o resultado será
exibido na tela*/
      case "/":
     div = a / b
     print "Resultado do cálculo: " + div
     break

/*Caso não seja nenhuma das opções acima ele exibirá uma mensagem de erro*/
     default:
     println "Caractere não encontrado. Digite uma das operações seguites(+, -, *, /)"
          }
      }
}
```

A exibição final, após a execução do programa, será a mesma elaborada na questão 4. O que você deve absorver é que a decisão lógica – *Switch case* – possui algumas características quanto a escrita, como é o caso do break e do default, os dois-pontos em cada case, por exemplo. A propósito, trate essa decisão como casos de escolha, onde você, usuário, tem opções a serem escolhidas e, após essa escolha, seguirá um caminho programado dentro de cada um desses casos.

6 - O programa deve começar nos solicitando a digitação de um número, inteiro e positivo, a partir de então o programa deverá exibir na tela a tabuada deste número. A entrada de dados não deverá ser validada, vamos acreditar que o usuário sempre digitará um valor devido.

```
class Numero {
static void main(args) {

Scanner leitura = new Scanner(System.in)

//Declaração das variáveis globais num (número digitado pelo usuário), multiplicador (índice que vai de 0 a 10 na tabuada) e resultado (total da conta):
int num, multiplicador = 0, resultado

print "Número: "
//Leitura da entrada de dados e atribuição da variável num (NÚMERO):
num = leitura.nextInt()

//Enquanto o índice for menor ou igual a 10, a tabuada será exibida:
while(multiplicador <= 10) {
        //Equação para calcular a tabuada:
        resultado = num * multiplicador
        //Exibição da tabuada na tela:
        println num + " x " + multiplicador + " = " + resultado
        //Incrementando o contador, lembrando que essa incrementação vai de 0 a 10:
        multiplicador++
        }
    }
}
```

A diferença entre esse programa e os outros é que nesse você percebe um termo da programação chamado laço de repetição, que consiste, basicamente, em gerar estruturas repetitivas dentro de uma determinada regra, neste caso, para o número multiplicativo da tabuada. Repare que o laço utilizado foi o *While* – enquanto – ou seja, enquanto o multiplicador for menor do que 10, lembrando que ele vai de 1 a 10, a programação não sairá do laço e

mostrará a tabuada do número digitado pelo usuário. Perceba que as outras programações existentes no programa já são de seu conhecimento. Note que, na execução do programa abaixo, o usuário digitará um número e o programa retornará com a tabuada do 0 ao 10 para ele.

Figura 25 – Tabuada do 8

```
Número: 8
8 x 0 = 0
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

Fonte: Própria (2021)

Figura 26 – Tabuada do 4

```
Número: 4
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Fonte: Própria (2021)

7 - O programa deverá exibir na tela os "n" primeiros termos da série: 2, 5, 10, 17, 26... onde o valor de "n" deverá ser inicialmente digitado. Em tempo esclareço que tal série tem como termo geral ($x^2 + 1$) onde x = {1, 2, 3, 4, 5 ...}.

```
class Sequência {
static void main(args) {

Scanner leitura = new Scanner(System.in)
//Declaração das variáveis globais num (número digitado pelo usuário) e resultado (valor obtido após a fórmula geral da sequência).
int num, i, resultado

print "Número: "
//Leitura da entrada de dados e atribuição da variável num (NÚMERO):
num = leitura.nextInt()

//Laço de repetição - FOR
for (i = 1; i <= num; i++) {
        resultado = Math.pow(i, 2) + 1
       print resultado + ", "
    }
  }
}
```

Neste exercício, a única diferença dos demais é que se tem um novo laço de repetição, este, por sua vez, chama-se *For*, que é uma estrutura bastante utilizada quando se precisa de executar diversas vezes um mesmo bloco de código. Nela, a declaração, a inicialização e a incrementação dos valores ficam no próprio cabeçalho da estrutura.

A primeira região do for é reservada para declaração e inicialização de variáveis. É importante notar que se pode declarar quantas variáveis forem necessárias. Normalmente, declara-se uma única variável. Já a segunda região – região central do *For* - é onde se coloca a condição

para que o laço seja repetido, esta, no que lhe concerne, pode ser qualquer uma, porém, é comum e recomendável colocar uma condição utilizando a variável ou variáveis declaradas na primeira parte dele. Em síntese, essa região é responsável por determinar quantas vezes o laço será repetido. Agora, a terceira parte do *For* é um local reservado para incrementar valores à variável que foi declarada na primeira parte e comparada na segunda.

Repare também na função matemática *Math*, dessa vez, utilizou-se a *pow*, que eleva um número ao quadrado. Ainda convém lembrar que se você quisesse programar utilizando um outro laço de repetição, por exemplo, o *While*, como no exercício abaixo, o resultado seria o mesmo, ou seja, os laços de repetição podem ser utilizados da maneira que você preferir, mas é claro que, algumas vezes, a escolha de um, para alguns códigos, pode ser mais fácil do que outro.

```
while (i <= num) {
    resultado = Math.pow(i, 2) + 1
    print resultado + ", "
    i++
}
```

Após a execução do programa, esta será a exibição se por acaso sugerir os cinco primeiros termos da sequência.

Figura 27 – Sequência de 5 números

```
Número: 5
2, 5, 10, 17, 26,
```

Fonte: Própria (2021)

8 - Temos aqui um clássico, o programa deverá listar no vídeo os termos da série de Fibonacci (1, 1, 2, 3, 5, 8, 13, 21 ...) menores que 1000.

```
class Fibonacci {
static void main(args) {

        //Definição das variáveis a serem utilizadas no programa
        def i, a = 1, b = 0, c
        //Laço de repetição usado para fornecer a sequência de Fibonacci:
        for (i = 1; i < 40; i++) {
              c = a + b
            //Se os valores da sequência ultrapassarem 1000, o programa parará (break):
            if(c >= 1000) {
                break
             } else {
                      print c + ", "
                   a = b
                     b = c
             }
         }
    }
}
```

Note que, nesse exercício, tem-se uma estrutura de repetição *for* junto com um comando de decisão lógica *if* e *else*, portanto, são múltiplas as formas de montar um código em Groovy.

Figura 28 – Sequência de Fibonacci

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
```

Fonte: Própria (2021)

9 - Entrar com dois valores via teclado, onde o segundo deverá ser maior que o primeiro. Caso contrário solicitar novamente a digitação do segundo valor, o que deve ser repetido até que o usuário atenda a condição definida.

```
class DoisValores {
static void main(args) {

        Scanner leitura = new Scanner(System.in)
        //Declaração das variáveis
        double num1, num2

        print "1º Número: "
        //Leitura da entrada de dados e atribuição da variável num1(NÚMERO 1):
        num1 = leitura.nextDouble()

        print "2º Número: "
        //Leitura da entrada de dados e atribuição da variável num2(NÚMERO 2):
        num2 = leitura.nextDouble()

        //Laço de repetição usado para mostrar que enquanto o segundo número for menor que o primeiro, o usuário deverá digitar novamente:
        while (num1 >= num2) {
                println "Atenção, o segundo número deve ser maior que o primeiro."
                print "2º Número: "
                num2 = leitura.nextDouble()
        }
                 //Exibição após as condições descritas:
                println num1 + " e " + num2
                print "Parabéns, você digitou os números de acordo com as regras!"

        }
}
```

Quando executado, esta será a exibição vista pelo usuário, que inserirá os valores e o programa comparará para saber se está dentro da regra programa ou não.

Figura 29 – Comparação de valores

```
1º Número: 2
2º Número: 1
Atenção, o segundo número deve ser maior que o primeiro.
2º Número: 2
Atenção, o segundo número deve ser maior que o primeiro.
2º Número: 8
2.0 e 8.0
Parabéns, você digitou os números de acordo com as regras!
```

Fonte: Própria (2021)

O usuário digitou o número dois, como primeiro número, e um, no segundo número, entretanto, a regra do programa elaborado consistem em dizer que o segundo número deve ser maior que o primeiro, sendo assim, o laço de repetição *While* é o melhor caminho para resolver esse problema, uma vez que ele não permite a saída final do programa enquanto não ajustada pelo usuário, isto é, enquanto ele não digitar um número maior do que o primeiro na segunda parte. Após esse ajuste, onde o usuário digitou o número 8, o programa encerrou com a mensagem de êxito.

10 - Entrar via teclado com o sexo de determinado usuário, aceitar somente "F" ou "M" como respostas válidas, caso o valor digitado seja indevido repetir o processo até que se digite um dos dois caracteres válidos.

```
class Sexo {
static void main(args) {

        Scanner leitura = new Scanner(System.in)

        //Declaração da variável sexo:
        String sexo

        print "Sexo (F/M): "
        //Leitura da entrada de dados e atribuição da variável sexo (Sexo do usuário):
        sexo = System.in.newReader().readLine().toString().toUpperCase()

        //Enquanto o sexo do usuário for diferente de F ou M, o programa ficará nesse looping,
perguntando-lhe o sexo:
        while (sexo != "F" && sexo != "M") {
                //Exibição do erro do sexo digitado para que o usuário compreenda que a digitação foi
incorreta:
                println "Você digitou um caractere diferente de F ou M, repare: " + sexo
                print "Sexo (F/M): "
                sexo = System.in.newReader().readLine().toString().toUpperCase()
        }
        //Caso o sexo do usuário seja F ou M, o programa virá diretamente para essa parte, ou seja, não
entrará no looping, porque estará de acordo com o solicitado:
                print "Agora sim, você digitou um caractere válido, note: " + sexo
        }
}
```

A lógica a ser utilizada, nesse programa, é a mesma dos outros, dessa forma, nesse exercício, enquanto o sexo digitado não for F ou M, um *looping* solicitando o reparo da informação fornecida pelo usuário será necessário e, quando ela estiver de acordo com a regra predefinida, seguirá o caminho da programação, isto é, exibirá que o sexo digitado está de acordo com o que foi programado.

Figura 30 – Análise do sexo masculino

```
Sexo (F/M): t
Você digitou um caractere diferente de F ou M, repare: T
Sexo (F/M): a
Você digitou um caractere diferente de F ou M, repare: A
Sexo (F/M): m
Agora sim, você digitou um caractere válido, note: M
```

Fonte: Própria (2021)

Figura 31 – Análise do sexo feminino

```
Sexo (F/M): i
Você digitou um caractere diferente de F ou M, repare: I
Sexo (F/M): q
Você digitou um caractere diferente de F ou M, repare: Q
Sexo (F/M): f
Agora sim, você digitou um caractere válido, note: F
```

Fonte: Própria (2021)

No Groovy, em substituição do laço de repetição *Do while*, o qual delimita uma condição a ser realizada antes do enquanto, todavia, não tem em sua construção de códigos, usa-se um *While* com um *If*, sendo que a exibição final é a mesma, repare neste trecho de código que pode ser substituído no programa anterior:

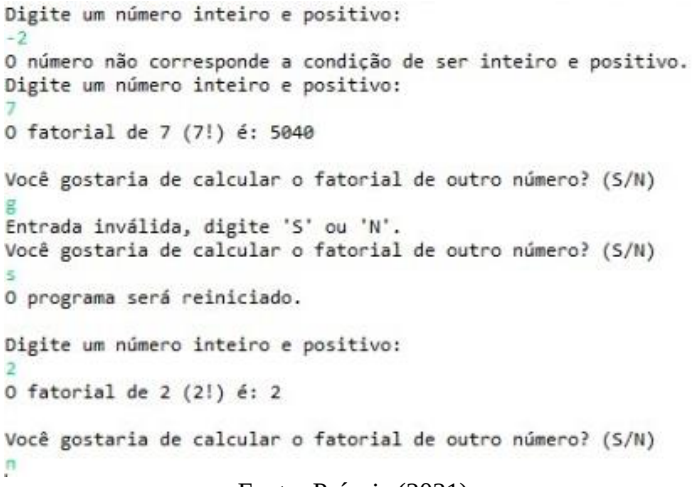
```
    while (sexo != "F" && sexo != "M") {
  //Exibição do erro do sexo digitado para que o usuário compreenda que a digitação foi incorreta:
          println "Você digitou um caractere diferente de F ou M, repare: " + sexo
          print "Sexo (F/M): "
          sexo = System.in.newReader().readLine().toString().toUpperCase()

          //Caso o sexo do usuário seja F ou M, o programa virá diretamente para essa parte, ou seja, não
entrará no looping, porque estará de acordo com o que foi solicitado:
          if(sexo == "F" || sexo == "M") {
          print "Agora sim, você digitou um caractere válido, note: " + sexo
          break
          }
    }
```

11 - O programa deverá nos permitir a digitação de um número inteiro e positivo, essa entrada deverá ser repetida até que o usuário satisfaça a condição. Feito isso o programa deverá calcular e exibir o fatorial deste número. Ao fim questionar se desejamos continuar ou não e essa entrada só deverá aceitar como resposta "S" ou "N" e a pergunta deverá ser repetida até que o usuário responda corretamente. Se a resposta for "S" então deveremos voltar ao início do programa e repetir tudo novamente, caso contrário o programa deverá ser encerrado.

```
class Fatorial {
static void main(args) {

        Scanner ler = new Scanner(System.in);
        boolean resp = true
        String repetir

        //Usando while para verificar no final se quer refazer o programa
        while(resp == true){

        //Pedindo a entrada de valor inteiro e armazenando
        println "Digite um número inteiro e positivo: "
        int num =  ler.nextInt()

        //Repetindo a pergunta até o valor ser um número positivo
        while (num <= 0) {
              println "O número não corresponde a condição de ser inteiro e positivo."
                println "Digite um número inteiro e positivo: "
                num =  ler.nextInt()
        }

        //Calculando o fatorial
        int fat=1
        for(int i=1; i<=num; i++) {
                fat *= i;
        }

        //Exibindo o resultado do fatorial
        println "O fatorial de " + num+" ("+ num + "!) " + "é: " + fat
        println ""
        println "Você gostaria de calcular o fatorial de outro número? (S/N)"
        repetir = ler.next()

        repetir = repetir.toUpperCase()
        //Verificando se a entrada confere com as condiçoes S ou N
        while (repetir != "S" && repetir != "N") {
                println "Entrada inválida, digite 'S' ou 'N'."
                println "Você gostaria de calcular o fatorial de outro número? (S/N)"
                repetir = ler.next()
                repetir = repetir.toUpperCase()
        }

        //Utilizando a decisão lógica para verificar a resposta dele e executar a escolha de repetir o
programa ou encerrar
           switch (repetir) {
                case 'S':
                        resp = true
                        println "O programa será reiniciado."
                        println " "
                        break
                case 'N':
                        resp = false
                        break
           }
       }
   }
}
```

Quando clicado em executar, esta será a exibição de acordo com cada informação fornecida por você (informações em verde) e baseada na programação elaborada.

Figura 32 – Fatorial



Fonte: Própria (2021)

12 - O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso exibir os valores digitados em tela na ordem inversa à da digitação.

```
class DigitacaoInversa {
static void main(args) {

        //Criando o vetor para armazenar os 10 números digitados e o contador i para o índice
de repetição:
        def numeros = new Double[10]
        int i

        //Laço de repetição para solicitar os 10 números e armazená-los nas posições do vetor:
        for (i=0; i < 10; i++) {
                print "" + (i+1) + "º número: "
                numeros[i] =  System.in.newReader().readLine().toDouble()
        }

        //Exibição vazia para pular linha:
        println ""

        //Laço de repetição para exibir a ordem invesa da digitação:
        for (i=9; i >= 0; i--) {
                        print numeros[i] + ", "
        }
    }
}
```

Figura 33 – Exibição invertida dos números

```
1º número: 9
2º número: 6
3º número: 6
4º número: 3
5º número: 2
6º número: 5
7º número: 9
8º número: 8
9º número: 7
10º número: 45

45.0, 7.0, 8.0, 9.0, 5.0, 2.0, 3.0, 6.0, 6.0, 9.0,
```

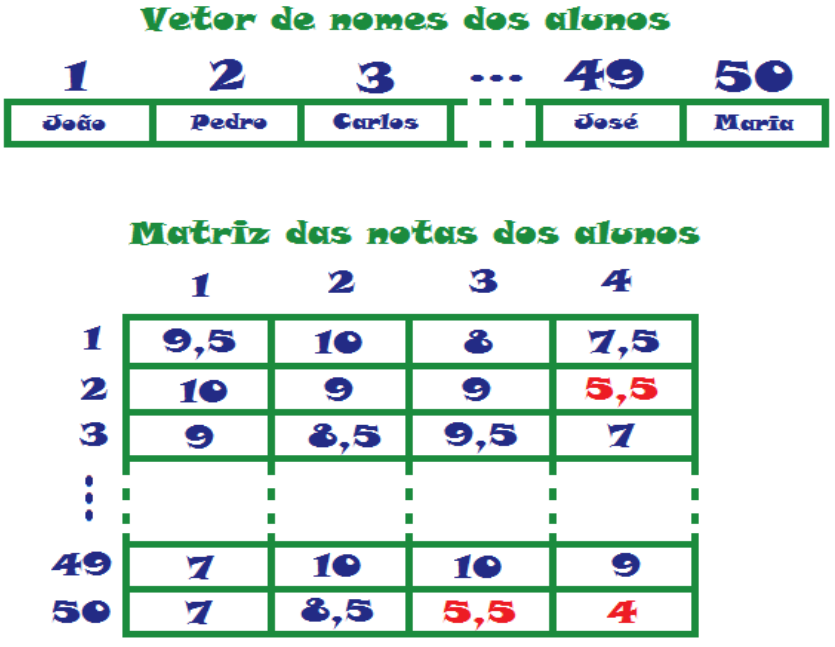
Fonte: Própria (2021)

Nessa parte do programa, há o uso de variáveis indexadas, que são *Arrays*, ou melhor, os vetores e as matrizes.

Os vetores são *Arrays* unidimensionais, que armazenam várias variáveis do mesmo tipo. Neles, podem utilizar um vetor com diversos tamanhos, os quais são determinados na programação, como no exercício anterior, onde determinou que o tamanho dele seria 10. Agora, as matrizes são *Arrays* multidimensionais, basicamente, são vetores de vetores. Vale destacar também que o item do vetor (ou matriz) é acessado por um número chamado índice, este, por sua vez, quando se trata de vetor, recebe apenas um valor, mas, em se tratando de matriz, dois valores (linha e coluna).

Para que a sua compreensão fique facilitada, analise esta ilustração lembrando das explicações fornecidas acima:

Figura 34 – Dicas de vetores e matrizes



Fonte: Própria (2021)

O laço de repetição For também foi usado, de modo a preencher cada espaço do vetor “números”. O primeiro recebe e armazena cada índice vetorial; o segundo fica responsável em inverter a ordem que foi digitada pelo usuário, por isso, em sua última posição da tríade sequencial do For, foi utilizado o “i--“, o qual tem a função de voltar em ordem contrária cada item do vetor preenchido.

13 - O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso criar um laço capaz de calcular a somatória desses 10 valores e então exibir a média desses valores.

```
class Ex8 {
static void main(args) {
        //Definição do scanner que irá receber os algarismos
        Scanner sc1 = new Scanner(System.in)

        // eclaração do array de 10 posições
        int [] num
        num = new int [ 10 ]

        //Declaração das variáveis
        int soma=0
        double resultado=0

        //Laço de repetição para variar as posições do vetor.
        for(int i = 0; i < num.length; ++i) {

        //Print para o usuário digitar os números e armazenar o valor digitado no vetor
                print "Digite o " + (i+1) + "º valor: "
                num[i] = sc1.nextInt()

                //A cada número digitado é somado na variável
                soma += num[i]
                }

                //Realização do cálculo da média de todos os números que o usuário inseriu
                resultado = soma/10

                //Print na tela para visualização do resultado
                println "Resultado da média: " + resultado
        }
}
```

Figura 35 – Exibição da média dos vetores

```
Digite o 1º valor: 1
Digite o 2º valor: 2
Digite o 3º valor: 3
Digite o 4º valor: 4
Digite o 5º valor: 5
Digite o 6º valor: 6
Digite o 7º valor: 7
Digite o 8º valor: 8
Digite o 9º valor: 9
Digite o 10º valor: 10
Resultado da média: 5.5
```

Fonte: Própria (2021)

14 - O programa deverá nos permitir digitar e armazenar dez números na memória do computador. Feito isso criar um laço capaz de identificar o maior e o menor dos números digitados e exibi-los ao final.

```
class Ex9 {
static void main(args) {
        //Definição do scanner que irá receber os algarismos
        Scanner sc = new Scanner (System.in)

        //Declaração do array de 10 posições
        int [] valor = new int[10]

        //Declaração das variáveis
        int maior = 0, menor = 0

        //Laço de repetição para variar as posições do vetor.
        for(int i = 0; i < valor.length; i++) {
                //Print para o usuário digitar os números e armazenar o valor digitado no vetor
                print "Digite o " + (i+1) +"º valor: "
                valor[i] = sc.nextInt()

                //If para realizar a verificação do maior valor e armazenar na variável
                if(valor[i] > maior) {
                                maior = valor[i]
                }
        }
                //Laço de repetição para percorrer o array em busca no menor valor
        for (int j = 0; j < valor.length; j++) {
        /*If para armazenar o valor da 1º posição na variável menor e depois começar a verificação com os
valores das posições seguintes do vetor*/
                        if (j <= 0) {
                        menor = valor[j]
                        }
                        //If para realizar a verificação do menor valor e armazenar na variável
                        else if (valor[j] < menor) {
                                menor = valor[j]
                        }
        }
                //Print na tela com os resultados da pesquisa
                println "Maior valor = "+ maior
                println "Menor valor = "+ menor
    }
}
```

Figura 36 – Exibição do maior e do menor valor no vetor

```
Digite o 1º valor: 5
Digite o 2º valor: 6
Digite o 3º valor: 7
Digite o 4º valor: 8
Digite o 5º valor: 9
Digite o 6º valor: 10
Digite o 7º valor: 20
Digite o 8º valor: 30
Digite o 9º valor: 40
Digite o 10º valor: 50
Maior valor = 50
Menor valor = 5
```

Fonte: Própria (2021)

15 - O programa deverá nos permitir digitar e armazenar o nome e idade de dez pessoas. Feito isso deverá nos solicitar a digitação de um nome e então proceder a pesquisa e informar a idade do sujeito pesquisado caso ele se encontre armazenado, caso contrário informar o fato através de mensagem "Pessoa não localizada", ao final verificar se nova consulta é desejada, validar a resposta do usuário no sentido de só aceitar "S" ou "N". Obviamente que no caso de nova consulta a digitação dos dez nomes/idades não deve ser repetido.

```
class ex10 {
static void main(def args) {
        //Declarando as variáveis que serão utilizadas
        String [] nome = new String [10]
        int [] idade = new int [10]
        String resposta = null

        //Laço de repetição para digitação de nomes e idades de 10 pessoas
        for(int i = 0; i < 10; i++) {
        println "Informe o nome da pessoa n° " + (i+1) + ":"
        nome[i] = System.in.newReader().readLine().toUpperCase()
        println "Informe a idade da pessoa n° " + (i+1) + ":"
        idade[i] = System.in.newReader().readLine().toInteger()
        }

        //Perguntando se deseja realizar uma pesquisa
        while (resposta != "S" && resposta != "N") {
        println "Deseja realizar uma pesquisa? (S ou N)"
        resposta = System.in.newReader().readLine().toUpperCase()
        }

        //Com a resposta "S", entra no laço de repetição para realizar a pesquisa
        while (resposta == "S") {
        //Declarando e inicializando variáveis que serão utilizadas para pesquisa
        int idadep = 0
        String nomep = null
        resposta = null
        //Perguntando nome e armazenando em uma variável
        println "Digite o nome: "
        nomep = System.in.newReader().readLine().toUpperCase()

        //Laço de repetição para procurar o nome digitado
        for (int i = 0; i < 10; i++) {
        //Se encontrar o nome, a idade da pessoa é exibida
        if (nomep == nome[i])
        {
        idadep = idade[i]
        println "A idade é: " + idadep + "anos"
        }
        //Caso não encontre a pessoa, uma mensagem é exibida informando que a pessoa não foi localizada
        if (idadep == 0 && i == 9)
        {
        println "Pessoa não localizada!"
        }
    }
```

```
        //Com o fim da pesquisa anterior, pede que informe se será necessária mais uma pesquisa
        while(resposta != "S" && resposta != "N"){
        println "Deseja realizar uma pesquisa? (S ou N) "
        resposta = System.in.newReader().readLine().toUpperCase()
        }
        }

        println "FIM"
        }
}
```

Figura 37 – Localização de pessoa

```
Informe o nome da pessoa nº 1:
Pessoa 01
Informe a idade da pessoa nº 1:
11
Informe o nome da pessoa nº 2:
Pessoa 02
Informe a idade da pessoa nº 2:
12
Informe o nome da pessoa nº 3:
Pessoa 03
Informe a idade da pessoa nº 3:
13
Informe o nome da pessoa nº 4:
Pessoa 04
Informe a idade da pessoa nº 4:
14
Informe o nome da pessoa nº 5:
Pessoa 05
Informe a idade da pessoa nº 5:
15
Informe o nome da pessoa nº 6:
Pessoa 06
Informe a idade da pessoa nº 6:
16
Informe o nome da pessoa nº 7:
Pessoa 07
Informe a idade da pessoa nº 7:
17
Informe o nome da pessoa nº 8:
Pessoa 08
Informe a idade da pessoa nº 8:
18
Informe o nome da pessoa nº 9:
Pessoa 09
Informe a idade da pessoa nº 9:
19
```

Fonte: Própria (2021)

Figura 38 – Continuação do exercício 15

```
Informe o nome da pessoa nº 10:
Pessoa 10
Informe a idade da pessoa nº 10:
20
Deseja realizar uma pesquisa? (S ou N)
s
Digite o nome:
Pessoa 07
A idade é: 17
Deseja realizar uma pesquisa? (S ou N)
e
Deseja realizar uma pesquisa? (S ou N)
g
Deseja realizar uma pesquisa? (S ou N)
s
Digite o nome:
Pessoa 14
Pessoa não localizada
Deseja realizar uma pesquisa? (S ou N)
n
FIM
```

Fonte: Própria (2021)

16 - Determinado cinema tem 20 fileiras (de 1 a 20) com 15 cadeiras (de 1 a 15) cada uma, portanto estamos falando de uma sala com 300 assentos, que deve ser reproduzida através de um array de 20 linhas por 15 colunas. O programa deve começar solicitando do usuário a digitação de seu nome, o número da fileira e cadeira em que deseja se sentar, se o assento estiver vazio reservá-lo registrando no array seu nome, caso contrário informar que o assento está ocupado. Feito isso o programa deverá nos questionar se desejamos nova reserva, validando nossa resposta e repetindo todo o processo.

```
class ex11 {
static void main(def args) {
                //Criando matriz para armazenar posição de cadeiras reservadas
                String [][] cadeiras = new String [20][15]

                //Criando variáveis que serão utilizadas
                String resposta = null
                String nome = null
                int coluna, fileira = 0

                //Laço de repetição iniciando a execução do programa e verificando a resposta
                while(resposta != "S" && resposta != "N") {
                        println "Pronto para reservar sua cadeira? "
                        resposta = System.in.newReader().readLine().toUpperCase()
                }

                //Laço de repetição principal para reserva da cadeira caso a resposta fornecida seja "S"
                while(resposta == "S"){
                resposta = null
                fileira = 0
                coluna = 0

                //Armazenando nome digitado pelo usuário
                println "Digite o nome: "
                nome = System.in.newReader().readLine().toUpperCase()

                //Laços de repetição verificando se o valor digitado é válido para escolha de cadeira
                while(fileira < 1 || fileira > 20){
                println "Digite a fileira: "
                fileira = System.in.newReader().readLine().toInteger()
                }

                while(coluna < 1 || coluna > 15) {
                println "Digite o número da cadeira: "
                coluna = System.in.newReader().readLine().toInteger()
                }
```

```
                //Criando variáveis que definem posição da cadeira reservada na matriz cadeiras
                int a = fileira - 1
                int b = coluna - 1

                //Verificando se cadeira está ocupada, caso esteja ocupada mensagem será exibida
                if(cadeiras[a][b] != null)
                {
                println "Cadeira ocupada por " + cadeiras[a][b]
                }
                //Caso não esteja ocupada o nome da pessoa será armazenado na posição
                else{
                cadeiras[a][b] = nome
                }

                //Perguntando se deseja reservar nova cadeira para continuar no laço de repetição
principal responsável pela reserva de cadeiras
                while(resposta != "S" && resposta != "N"){
                println "Deseja reservar outra cadeira?"
                resposta = System.in.newReader().readLine().toUpperCase()
                }
                }
                //Se a resposta for "N", o programa será finalizado com a saída do laço de repetição
                println "FIM"
        }
}
```

Figura 39 – Cadeiras de cinema

```
Pronto para reservar sua cadeira?
s
Digite o nome:
Rebeca
Digite a fileira:
01
Digite o número da cadeira:
01
Deseja reservar outra cadeira?
f
Deseja reservar outra cadeira?
h
Deseja reservar outra cadeira?
s
Digite o nome:
Maria Clara
Digite a fileira:
1
Digite o número da cadeira:
1
Cadeira ocupada por REBECA
Deseja reservar outra cadeira?
n
FIM
```

Fonte: Própria (2021)

4.1. TRABALHANDO COM INTERFACE GRÁFICA

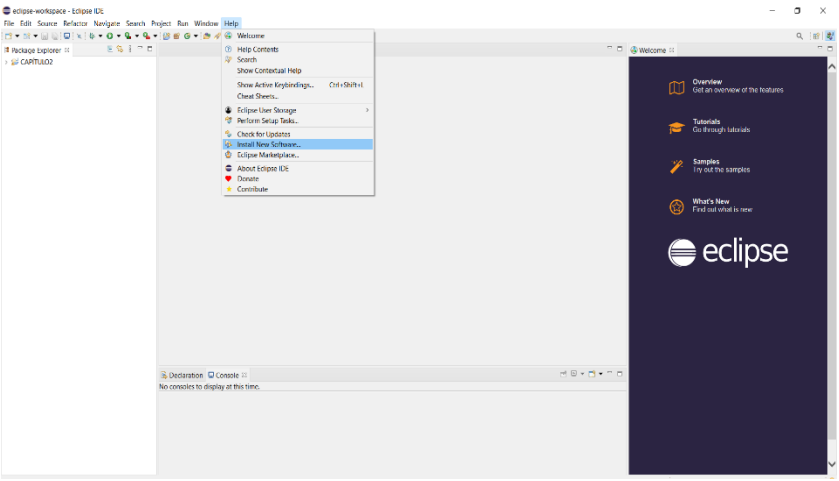
A partir de agora, os exercícios continuarão sendo simples no intuito de fazer com que você aprenda a programar na linguagem Groovy, porém, nesta nova etapa, um novo conceito será abordado, este, por sua vez, é o de interface gráfica, que consiste em um modelo de programação onde há interação com os dispositivos digitais através de elementos gráficos. Em síntese, os programas passam a ter um modo de exibição mais aprimorado, com vários recursos gráficos.

Para que se trabalhe com a interface, é necessário fazer a seguinte instalação, a qual será desenvolvido passo a passo a partir de já:

Passo 1: ABRA O ECLIPSE IDE – 2021 – 03.

Passo 2: CLIQUE EM “HELP/AJUDA”; “INSTALL NEW SOFTWARE”.

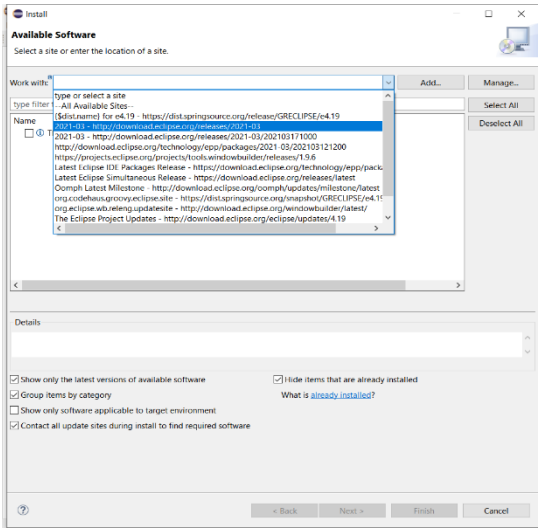
Figura 40 – Passo 2 para interface



Fonte: Própria (2021)

Passo 3: CLIQUE NO ITEM SELECIONADO EM AZUL ABAIXO.

Figura 41 – Passo 3 para interface

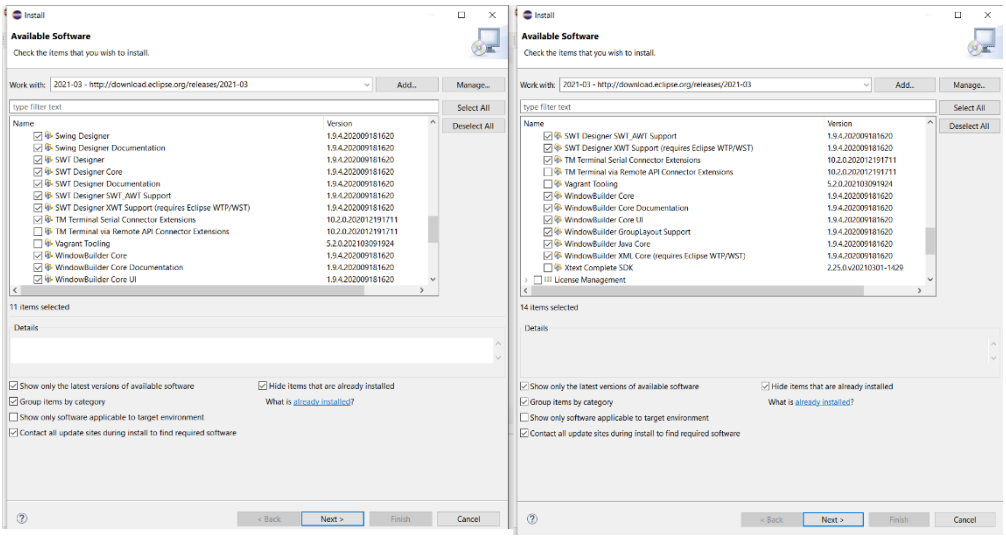


Fonte: Própria (2021)

Passo 4: CLIQUE NA SETA PARA ABRIR MAIS DE “GENERAL PURPOSE TOOLS”.

Selecione todos os itens que se inicia com “Swing”; selecione todos os itens que se iniciam com “SWT”; selecione todos os itens que se iniciam do “WindowsBuilder”. Clique em “Next/Próximo” após selecionar os itens necessários. Em seguida, clique em “Finish/Finalizar”.

Figura 42 – Passo 4 para interface

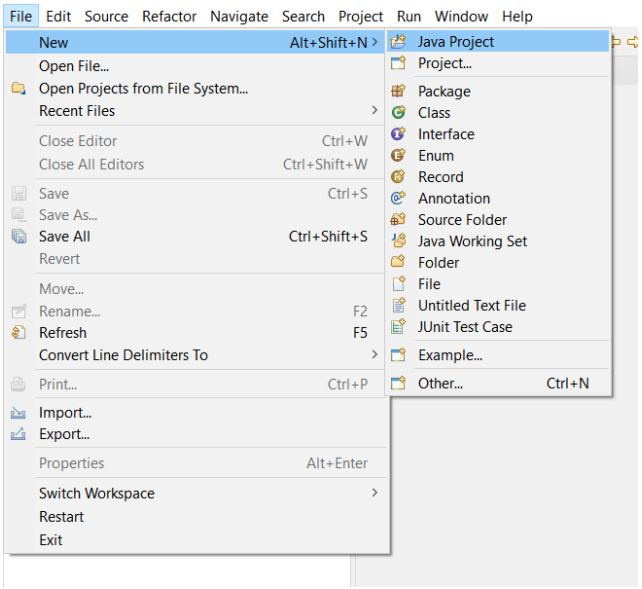


Fonte: Própria (2021)

A instalação das ferramentas para construir os “forms” já está finalizada.

Passo 5: CLIQUE EM “FILE”; “NEW”; “JAVA PROJECT”.

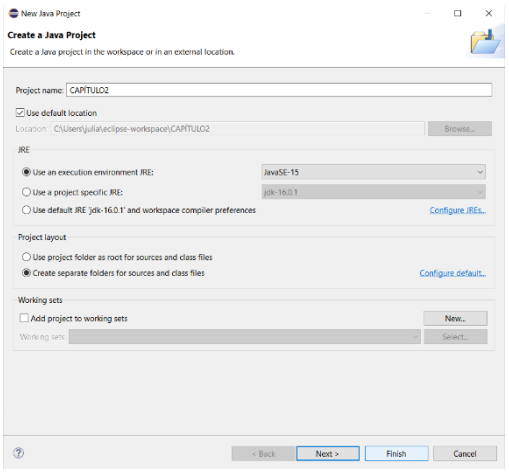
Figura 43 – Passo 5 para interface



Fonte: Própria (2021)

Passo 6: ATRIBUA UM NOME AO PROJETO E CLIQUE EM “FINISH/FINALIZAR”.

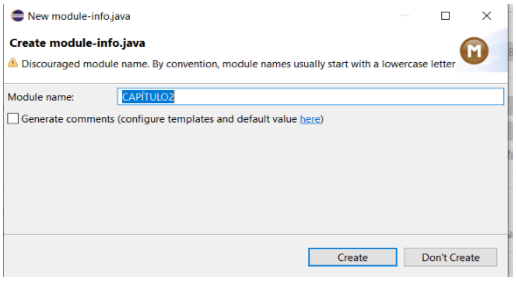
Figura 44 – Passo 6 para interface



Fonte: Própria (2021)

Passo 7: O ECLIPSE EXIBIRÁ A OPÇÃO DE CRIAR UM MÓDULO, QUE, NO CASO, NÃO SERÁ NECESSÁRIA. DESSA FORMA, CLIQUE EM “DON’T CREATE/NÃO CRIAR”.

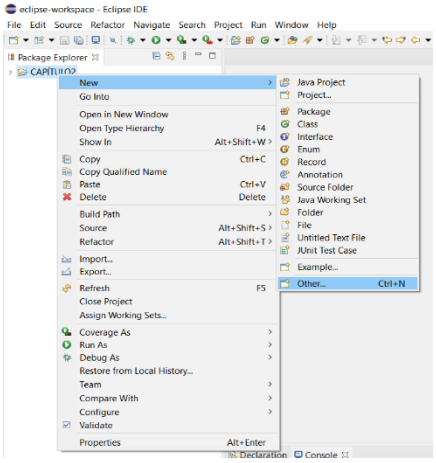
Figura 45 – Passo 7 para interface



Fonte: Própria (2021)

Passo 8: CLIQUE COM O BOTÃO DIREITO SOBRE O ARQUIVO DO PROJETO; CLIQUE EM “NEW/NOVO”; CLIQUE EM “OTHER/OUTROS”.

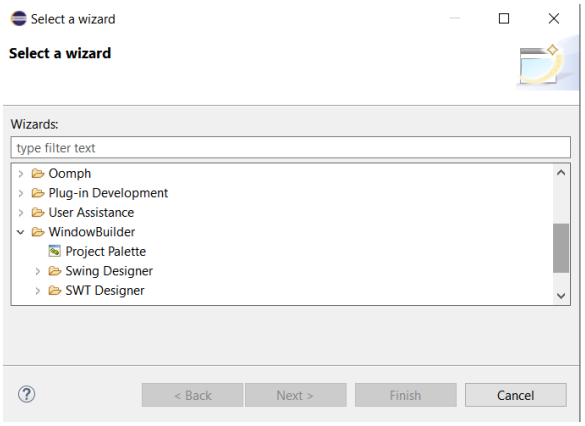
Figura 46 – Passo 8 para interface



Fonte: Própria (2021)

Passo 9: CLIQUE EM “WINDOW BULDIER”; “SWING DESIGNER”.

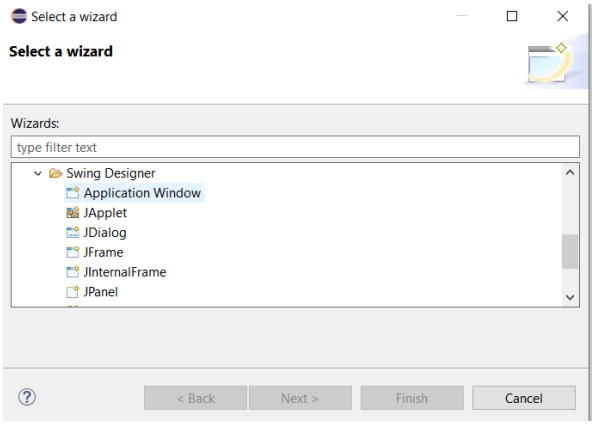
Figura 47 – Passo 9 para interface



Fonte: Própria (2021)

Passo 10: CLIQUE EM “APPLICATION WINDOW”.

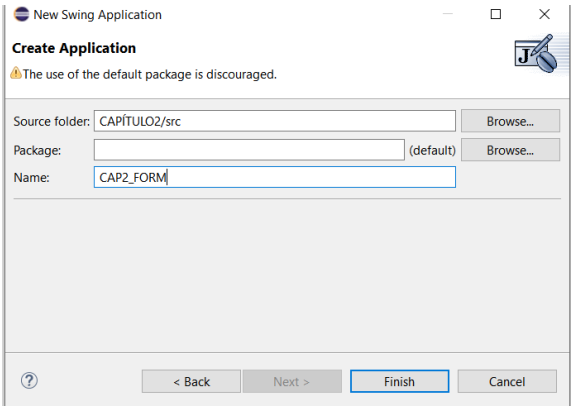
Figura 48 – Passo 10 para interface



Fonte: Própria (2021)

Passo 11: ATRIBUA UM NOME PARA O FORM; CLIQUE EM “FINISH/FINALIZAR”.

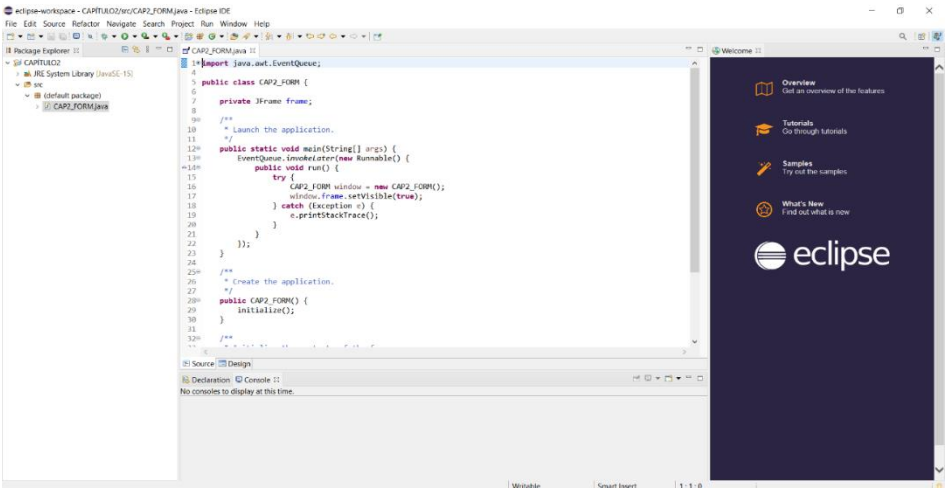
Figura 49 – Passo 11 para interface



Fonte: Própria (2021)

Passo 12: CLIQUE EM “DESIGN” PARA ABRIR A PALETA COM OS COMPONENTES.

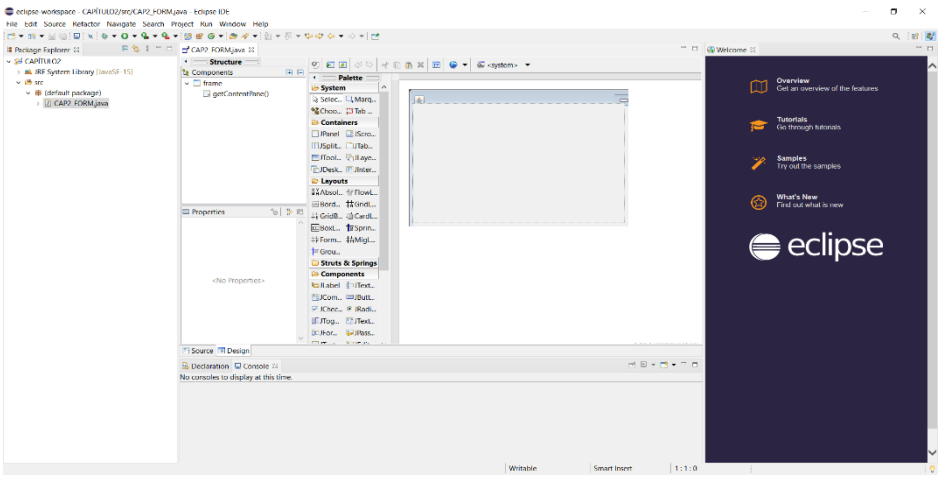
Figura 50 – Passo 12 para interface



Fonte: Própria (2021)

Passo 13: Com a paleta dos componentes aberta, basta apenas arrastar os itens necessários ao seu “form”.

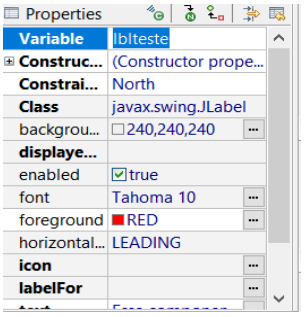
Figura 51 – Passo 13 para interface



Fonte: Própria (2021)

Nesta parte, é possível vizualizar as propriedades do componentes selecionados, como o nome da variável, cor, tamanho entre outros:

Figura 52 – Observação do passo 13



Fonte: Própria (2021)

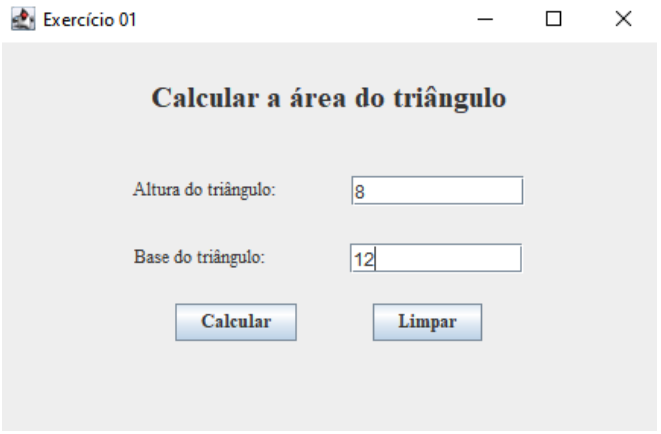
4.1.1. EXERCÍCIOS COM INTERFACE GRÁFICA

1 - A partir da digitação da base e altura de um triângulo o programa deverá calcular sua área e exibi-la no monitor.

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import javax.swing.JPanel;
import java.awt.FlowLayout;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JTextPane;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import static javax.swing.JOptionPane.showMessageDialog;
public class frm_ex02 {

	private JFrame frmExerccio;
	private JTextField txt_altura;
	private JTextField txt_base;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					frm_ex02 window = new frm_ex02();
					window.frmExerccio.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	/**
	 * Create the application.
	 */
	public frm_ex02() {
		initialize();
	}
	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frmExerccio = new JFrame();
		frmExerccio.setTitle("Exerc\u00EDcio 01");
		frmExerccio.setBounds(100, 100, 450, 300);
		frmExerccio.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		JPanel panel_titulo = new JPanel();
		FlowLayout fl_panel_titulo = (FlowLayout) panel_titulo.getLayout();
		fl_panel_titulo.setVgap(25);
		frmExerccio.getContentPane().add(panel_titulo, BorderLayout.NORTH);

		JLabel lblNewLabel_2 = new JLabel("Calcular a \u00E1rea do tri\u00E2ngulo");
		lblNewLabel_2.setFont(new Font("Times New Roman", Font.BOLD, 20));
		panel_titulo.add(lblNewLabel_2);
```

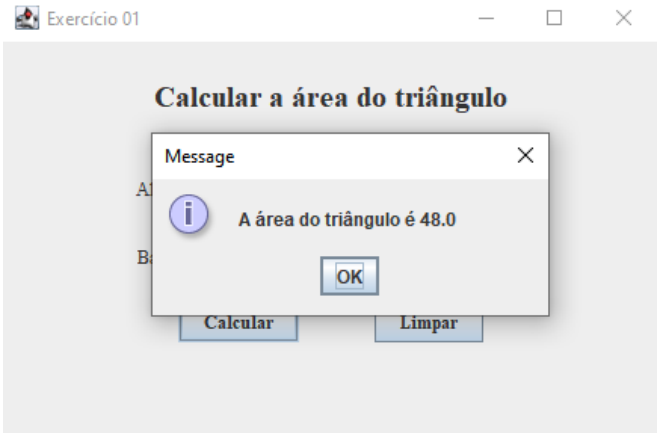
```
JPanel panel_conteudo = new JPanel();
                frmExerccio.getContentPane().add(panel_conteudo, BorderLayout.CENTER);

                JPanel panel_altura = new JPanel();
                FlowLayout fl_panel_altura = (FlowLayout) panel_altura.getLayout();
                fl_panel_altura.setHgap(50);
                fl_panel_altura.setVgap(10);
                panel_conteudo.add(panel_altura);

                JLabel lblNewLabel = new JLabel("Altura do tri\u00E2ngulo:");
                lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 12));
                panel_altura.add(lblNewLabel);

                txt_altura = new JTextField();
                panel_altura.add(txt_altura);
                txt_altura.setColumns(10);

                JPanel panel_base = new JPanel();
                FlowLayout fl_panel_base = (FlowLayout) panel_base.getLayout();
                fl_panel_base.setHgap(50);
                fl_panel_base.setVgap(10);
                panel_conteudo.add(panel_base);

                JLabel lblNewLabel_1 = new JLabel("Base do tri\u00E2ngulo:  ");
                lblNewLabel_1.setFont(new Font("Times New Roman", Font.PLAIN, 12));
                panel_base.add(lblNewLabel_1);

                txt_base = new JTextField();
                panel_base.add(txt_base);
                txt_base.setColumns(10);

                JPanel panel_botoes = new JPanel();
                FlowLayout fl_panel_botoes = (FlowLayout) panel_botoes.getLayout();
                fl_panel_botoes.setHgap(50);
                panel_conteudo.add(panel_botoes);

                JButton btn_calcular = new JButton("Calcular");
                btn_calcular.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                double base = Double.parseDouble(txt_base.getText());
                                double altura = Double.parseDouble(txt_altura.getText());

                                double area = (base * altura)/2;

                                showMessageDialog(null,"A área do triângulo é " + area);
                        }
                });
                btn_calcular.setFont(new Font("Times New Roman", Font.BOLD, 12));
                panel_botoes.add(btn_calcular);

                JButton btn_limpar = new JButton("Limpar");
                btn_limpar.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                txt_base.setText("");
                                txt_altura.setText("");
                        }
                });
                btn_limpar.setFont(new Font("Times New Roman", Font.BOLD, 12));
                panel_botoes.add(btn_limpar);
        }
}
```

Figura 53 – Área do triângulo por interface gráfica



Fonte: Própria (2021)

Figura 54 – Área do triângulo por interface gráfica (exibição)



Fonte: Própria (2021)

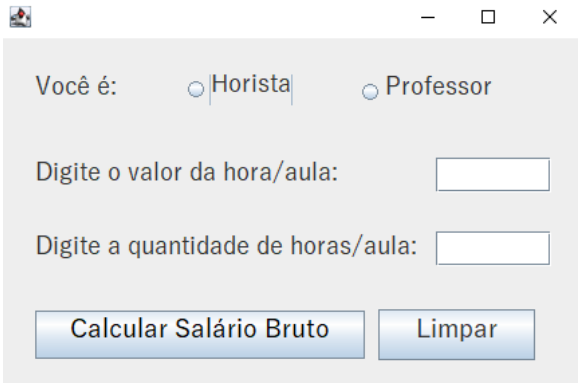
2 – Faça o cálculo do salário bruto de um professor e de um horista, que serão selecionados por meio de dois "radioButton", através do valor da hora aula e a quantidade de horas trabalhadas, dados estes que serão informados pelo usuário.

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import java.awt.Font;
import javax.swing.JButton;
import java.awt.Color;
import javax.swing.JRadioButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;

public class Exercicio2 {

	private JFrame frame;
	private JTextField valorText;
	private JTextField qtdText;
```

```
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				Exercicio2 window = new Exercicio2();
				window.frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

public Exercicio2() {
	initialize();
}

private void initialize() {
	frame = new JFrame();
	frame.setBounds(100, 100, 450, 300);
	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	frame.getContentPane().setLayout(null);

	JLabel lblNewLabel = new JLabel("Voc\u00EA \u00E9:");
	lblNewLabel.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
	lblNewLabel.setBounds(27, 27, 117, 26);
	frame.getContentPane().add(lblNewLabel);

	JLabel lblNewLabel_1 = new JLabel("Digite o valor da hora/aula:");
	lblNewLabel_1.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
	lblNewLabel_1.setBounds(27, 85, 239, 38);
	frame.getContentPane().add(lblNewLabel_1);

	JRadioButton rdbtn = new JRadioButton("Horista");
	JRadioButton rdbtn_1 = new JRadioButton("Professor");

	JLabel lblDigiteOValor = new JLabel("Digite a quantidade de horas/aula:");
	lblDigiteOValor.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
	lblDigiteOValor.setBounds(27, 140, 307, 38);
	frame.getContentPane().add(lblDigiteOValor);

	JButton calcBtn = new JButton("Calcular Sal\u00E1rio Bruto");
	calcBtn.addActionListener(new ActionListener() {

	 public void actionPerformed(ActionEvent e) {

	//transferindo o texto da JTextField para uma variável float, para conseguir calcular o salário
			float qtd = Float.parseFloat(valorText.getText());
			float valor = Float.parseFloat(qtdText.getText());
     float sb;
     String msg = "SALÁRIO BRUTO:";

    //verificando qual radioButton está selecionado para o cálculo
     if (rdbtn.isSelected())
     {
         sb = qtd * valor;
     }
     else
     {
         sb = (qtd * valor) * 1.25f;
     }
```

```
        //exibindo uma caixa de mensagem com o salário bruto
            JOptionPane.showMessageDialog(null, msg + sb);

                    }
                });
                calcBtn.setForeground(Color.BLACK);
                calcBtn.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 19));
                calcBtn.setBounds(27, 203, 246, 36);
                frame.getContentPane().add(calcBtn);

                JButton limparBtn = new JButton("Limpar");
                limparBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                valorText.setText("");
                  qtdText.setText("");
                  rdbtn.setSelected(false);
                  rdbtn_1.setSelected(false);

                        }
                });
                limparBtn.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 19));
                limparBtn.setBounds(283, 202, 117, 38);
                frame.getContentPane().add(limparBtn);

                rdbtn.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
                rdbtn.setBounds(137, 27, 109, 23);
                frame.getContentPane().add(rdbtn);

                rdbtn_1.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
                rdbtn_1.setBounds(267, 19, 109, 43);
                frame.getContentPane().add(rdbtn_1);

                valorText = new JTextField();
                valorText.setColumns(10);
                valorText.setBounds(326, 89, 86, 26);
                frame.getContentPane().add(valorText);

                qtdText = new JTextField();
                qtdText.setColumns(10);
                qtdText.setBounds(326, 144, 86, 26);
                frame.getContentPane().add(qtdText);
        }
}
```

Figura 55 – Exibição do exercício 2 por interface gráfica



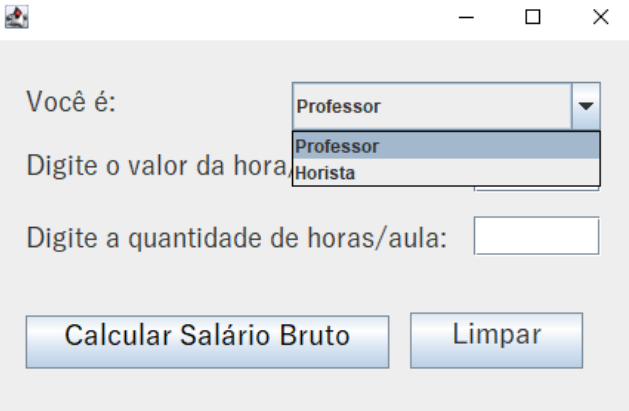
Fonte: Própria (2021)

3 – Faça o cálculo do salário bruto de um professor e de um horista, que serão selecionados por meio de uma "comboBox", através do valor da hora aula e a quantidade de horas trabalhadas, dados estes que serão informados pelo usuário.

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import java.awt.Font;
import javax.swing.JTextField;
import java.awt.Color;
import javax.swing.JOptionPane;

public class Exercicio3 {
	private JFrame frame;
	private JTextField valorText;
	private JTextField qtdText;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Exercicio3 window = new Exercicio3();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	public Exercicio3() {
		initialize();
	}
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 450, 300);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		//Criando um ComboBox com as opções "Professor" e "Horista"
		JComboBox voceE = new JComboBox();
		voceE.addItem("Professor");
		voceE.addItem("Horista");

		JButton calcBtn = new JButton("Calcular Sal\u00E1rio Bruto");
		calcBtn.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 19));
		calcBtn.setForeground(new Color(0, 0, 0));
		calcBtn.setBounds(22, 186, 246, 36);
		calcBtn.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				//transferindo o texto da JTextField para uma variável float, para conseguir
calcular o salário
				  float qtd = Float.parseFloat(valorText.getText());
				  float valor = Float.parseFloat(qtdText.getText());
		float sb;
		String msg = "SALÁRIO BRUTO:";
```

```
//a posição 0 corresponde a primeira, que neste caso é "Professor"
                if (voceE.getSelectedIndex() == 0 )
                {
                    sb = qtd * valor;
                }
                else
                {
                    sb = (qtd * valor) * 1.25f;
                }
                //exibindo uma caixa de mensagem com o salário bruto
                JOptionPane.showMessageDialog(null, msg + sb);
                        }
                  });

                  frame.getContentPane().setLayout(null);
                  frame.getContentPane().add(calcBtn);

                  JButton limparBtn = new JButton("Limpar");
                  limparBtn.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 19));
                  limparBtn.setBounds(282, 184, 117, 38);
                  limparBtn.addActionListener(new ActionListener() {
                          public void actionPerformed(ActionEvent e) {

                                  //deixando as JTextField sem texto, para limpá-las
                                  valorText.setText("");
                   qtdText.setText("");

                          }
                  });
                  frame.getContentPane().add(limparBtn);
JLabel lblNewLabel = new JLabel("Voc\u00EA \u00E9:");
                  lblNewLabel.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
                  lblNewLabel.setBounds(22, 33, 117, 26);
                  frame.getContentPane().add(lblNewLabel);

                  JLabel lblDigiteOValor = new JLabel("Digite a quantidade de horas/aula:");
                  lblDigiteOValor.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
                  lblDigiteOValor.setBounds(22, 119, 307, 38);
                  frame.getContentPane().add(lblDigiteOValor);

                  JLabel lblNewLabel_1 = new JLabel("Digite o valor da hora/aula:");
                  lblNewLabel_1.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 18));
                  lblNewLabel_1.setBounds(22, 70, 239, 38);
                  frame.getContentPane().add(lblNewLabel_1);

                  voceE.setBounds(202, 28, 209, 33);
                  frame.getContentPane().add(voceE);

                  valorText = new JTextField();
                  valorText.setBounds(325, 77, 86, 26);
                  frame.getContentPane().add(valorText);
                  valorText.setColumns(10);

                  qtdText = new JTextField();
                  qtdText.setBounds(325, 119, 86, 27);
                  frame.getContentPane().add(qtdText);
                  qtdText.setColumns(10);
        }
}
```

Figura 56 – Exibição do exercício 3 por interface gráfica



Fonte: Própria (2021)

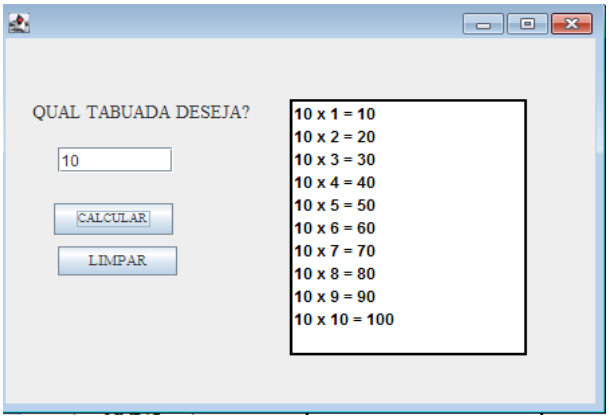
4 – Elabore um exercício que peça ao usuário um número. A partir desse número digita, exiba a sua respectiva tabuada. Insira também um botão com a função de limpar a exibição da tabuada.

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.SwingConstants;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Scanner;
import javax.swing.JList;
import javax.swing.DefaultListModel;
import javax.swing.AbstractListModel;
import java.awt.Color;
import javax.swing.border.BevelBorder;
import java.awt.Font;

	public class EX_04 {

	private JFrame frame;
	private JTextField txt_n;

	Scanner sc1 = new Scanner(System.in);
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					EX_04 window = new EX_04();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	public EX_04() {
		initialize();
	}
```

```
                private void initialize() {
                frame = new JFrame();
                frame.setBounds(100, 100, 456, 326);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.getContentPane().setLayout(null);

                JLabel lblNewLabel = new JLabel("QUAL TABUADA DESEJA?");
                lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                lblNewLabel.setBounds(20, 51, 184, 14);
                frame.getContentPane().add(lblNewLabel);

                JList list = new JList();
                list.setBorder(new BevelBorder(BevelBorder.LOWERED, Color.BLACK, Color.BLACK,
Color.BLACK, Color.BLACK));
                list.setForeground(Color.WHITE);

                list.setBounds(69, 75, 60, 202);
                frame.getContentPane().add(list);

                list.setModel(new AbstractListModel() {
                        String[] values = new String[] {};
                        public int getSize() {
                                return values.length;
                        }
                        public Object getElementAt(int index) {
                                return values[index];
                        }
                });
                txt_n = new JTextField();
                txt_n.setBounds(39, 86, 86, 20);
                frame.getContentPane().add(txt_n);
                txt_n.setColumns(10);

                DefaultListModel model = new DefaultListModel();
                JButton btn_calc = new JButton("CALCULAR ");
                btn_calc.setFont(new Font("Times New Roman", Font.PLAIN, 9));
                btn_calc.setBounds(36, 130, 92, 23);
                frame.getContentPane().add(btn_calc);
                btn_calc.addActionListener(new ActionListener() {

                 public void actionPerformed(ActionEvent e) {

                        String tabuada = txt_n.getText();

                        int tab = Integer.parseInt(tabuada);
                        int resultado;
                        int i;

                        for ( i = 1; i < 11; ++i)
                        {
                           resultado = tab *i;
                           model.addElement(tab + " x " + i + " = " + resultado);
                           list.setModel(model);
                        }
                    }
                });
                btn_calc.setBounds(36, 130, 89, 25);
                frame.getContentPane().add(btn_calc);
                list.setForeground(Color.BLACK);
                list.setBounds(212, 48, 177, 202);
                frame.getContentPane().add(list);
```

```
JButton btn_limpar = new JButton("LIMPAR");
                btn_limpar.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {

                                model.clear();
                                list.setModel(model);

                                JTextField field = (JTextField) txt_n;
            field.setText("");
                        }
                });

                btn_limpar.setFont(new Font("Times New Roman", Font.PLAIN, 11));
                btn_limpar.setBounds(39, 164, 89, 23);
                frame.getContentPane().add(btn_limpar);
        }
}
```

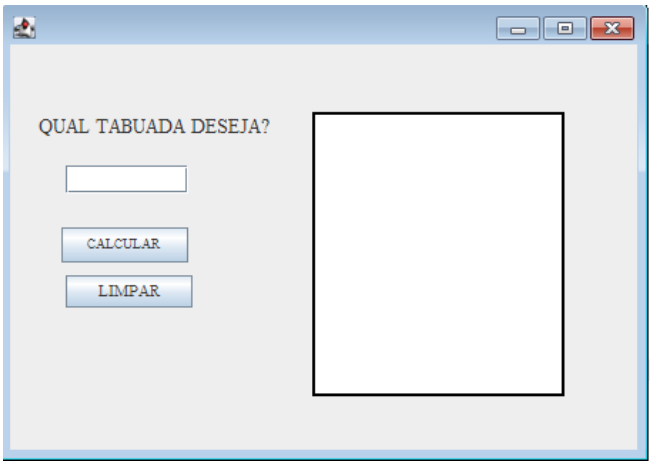
Figura 57 – Exibição do exercício 4 por interface gráfica



Fonte: Própria (2021)

Repare que o usuário informou ao programa o número 10. Após isso, ao clicar em "calcular", foi exibido a tabuada do número digitado. No final, este mesmo usuário clicou no botão limpar, que, por sua vez, limpará a caixa de exibição da tabuada.

Figura 58 – Função limpar ex. 4 por interface gráfica



Fonte: Própria (2021)

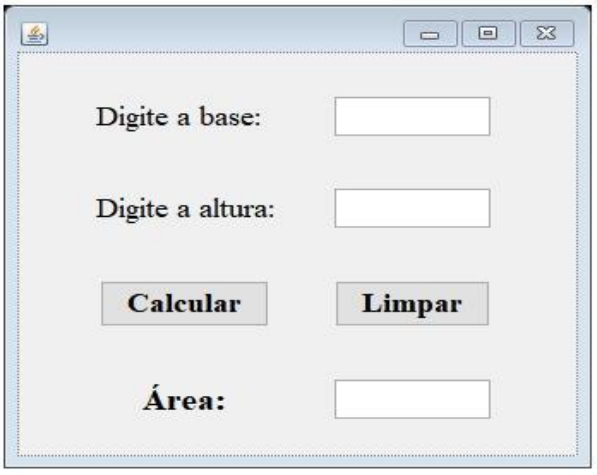
4.2. POO E PARADIGNAS DA ARQUITETURA DE PROGRAMAÇÃO DE MÚLTIPLAS CAMADAS

1 - A partir da digitação da base e altura de um triângulo o programa deverá calcular sua área e exibi-la no monitor.

```
private void
btnCalculoActionPerformed(java.awt.event.ActionEvent
evt) {
String vBase = txtBase.getText();
String vAltura = txtAltura.getText();
ClasseB verificação = new ClasseB();
String mens = verificacao.validaDados(vBase, vAltura);
if (mens == null){
ClasseTriang area = new ClasseTriang(Float.parseFloat(vBase),
Float.parseFloat(vAltura));
lblArea.setText("Área: " + Float.toString(area.calcularArea()));
}
else {
JOptionPane.showMessageDialog(null,mens);
} }
private void
btnLimpaActionPerformed(java.awt.event.ActionEve
nt evt) {
txtBase.setText("");
txtAltura.setText("");
lblArea.setText("Área: ");
}
//Toda a orientação a objetos foi construída através de uma classe construída em Groovy.
Class Triang {
float b;
float a;
ClasseTriangulo (B, A){
this.b = Vb;
this.a = Va;
}
```

```
public float calculoArea(){
return(b * a) / 2;
}
}
Classe de tratamento de erros:
class ClasseB {
public String validaDados(String b, String a)
{
if (b == "" || b.isNumber() == false || (b as
float) <= 0){
return "O prenchimento do valor da base está vazio,
não é numérico ou é menor que zero.";
}
if (a == "" || a.isNumber() == false || (a
as float) <= 0){
return "O prenchimento do valor da altura está
vazio, não é numérico ou é menor que zero.";
    }
  }
}
//Após programado estes passos o programa estará completo para execução.
```

Figura 59 – Área do triângulo por POO



Fonte: Própria (2021)

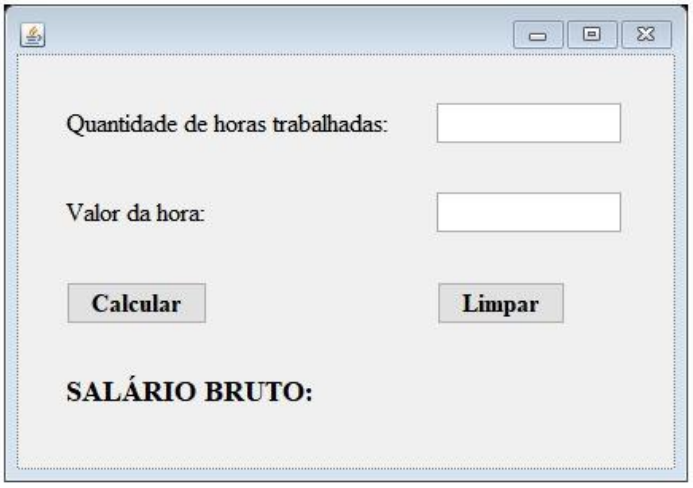
2 – Faça o cálculo do salário bruto de um horista através do valor da hora aula e a quantidade de horas trabalhadas, dados estes que serão informados pelo usuário.

```
//Botão Cálculo:
private void
btnCalculoActionPerformed(java.awt.event.ActionEve
nt evt) {
String quant = txtQtd.getText();
String v = txtValor.getText();
HoristaBLL horista = new HoristaBLL();
String valida = horista.validaDados(quant, v);
if (Erro.getErro()){
JOptionPane.showMessageDialog(null,Erro.getMens());
} else {
if (valida == null) {
Horista 1horista = new Horista();
1horista.setQtd(txtQtd.getText());
1horista.setValor(txtValor.getText());
ClasseG groovy = new
ClasseG(1horista.getSalarioBruto());
String s = groovy.getResposta().toString();
lblS.setText(s);
}
}
}
Botão limpar:
private void
btnLimpaActionPerformed(java.awt.event.ActionEv
ent evt) {
txtQtd.setText("");
txtValor.setText("");
lblSal.setText("");
txtQtd.requestFocus();
txtQtd.setEditable(true);
txtValor.setEditable(true);
}
```

```
txtValor.setEditable(true);

}

Classe groovy para receber valores do formulário:

class ClasseG {

String r;

ClasseGroovy(s){

this.r = "O salário bruto é " +s;

}

}

//Foi desenvolvido as classes "Erro.groovy", "HoristaBLL.groovy" e "Horista.groovy", a fim de contribuir para as requisições finais do programa.

class Erro {
private static boolean erro;

private static String mens;

public static void setErro(boolean _erro) { erro =

_erro; }

public static void setErro(String _mens) { erro = true;

mens = _mens; }

public static boolean getErro() { return erro; }

public static String getMens() { return mens; }

}

Class Horista {

private String quant;

private String v;

public void setquant(String _quant) { quant = _quant; }

public String getquant() { return quant; }

public void setValor(String _valor) { valor = _valor; }

public String getValor() { return v; }

public String getSalarioBruto()

{

return

(Float.parseFloat(getQtd())*Float.parseFloat(getValor())

).toString();

}

}
```

```
class HoristaBLL {
public String validaDados(String qtd, String
val) {
Erro.setErro(false);
if (val == "" || val.isNumber() == false || (val as
float) <= 0){
Erro.setErro("O campo VALOR DA HORA é
preenchimento OBRIGATÓRIO e precisa ser
NUMÉRICO e MAIOR QUE ZERO ");
return;
}
if (qtd == "" || qtd.isNumber() == false ||
(qtd as float) <= 0){
Erro.setErro("O campo QUANTIDADE DE HORAS é
preenchimento OBRIGATÓRIO e precisa ser
NUMÉRICO e MAIOR QUE ZERO ");
return Erro.setErro(true)
}}}
//Após a realização da programação dos dados acima o programa está pronto para execução.
```

Figura 60 – Salário bruto por POO



Fonte: Própria (2021)

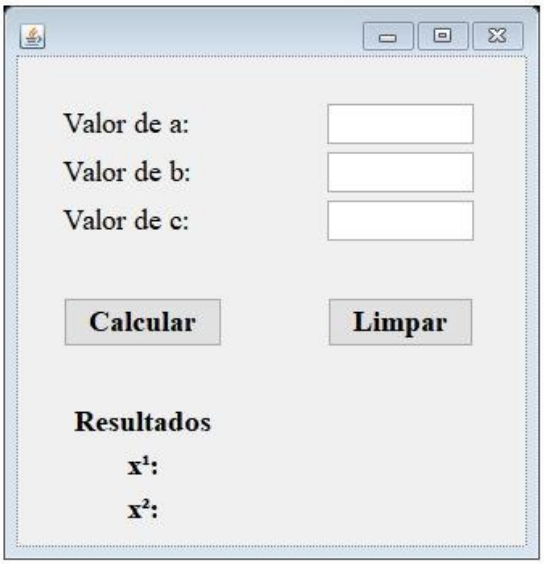
3 – Resolva uma equação do segundo grau.

```
class ExEquaSG {

float a;

float b;

float c;

ExEquaSG (Va,Vb,Vc){

this.a=Va; this.b=Vb; this.c=Vc;

}

public float calculoDelta() {

return (b*b) - (4 * a * c)

}

public float calculoraiz1(){

float d = calculoDelta();

return (-b + Math.sqrt(d)) / (2 * a);

}

public float calculoraiz2(){

float d = calculoDelta();

return (-b - Math.sqrt(d)) / (2 * a);

}

//Acima, há a presença de métodos que realizam o cálculo do delta a das raízes da equação. Em Java,
retornaremos esses valores para exibi-los no form.

class ClasseBLL {

public String vldados(String Va, String Vb,

String Vc) {

if (Va == "" || Va.isNumber() == false){

return "O prenchimento do valor de A está vazio ou

não é numérico.";

}

if (Vb == "" || Vb.isNumber() == false){

return "O prenchimento do valor de B está vazio ou

não é numérico.";

}

if (Vc == "" || Vc.isNumber() == false){

return "O prenchimento do valor de C está vazio ou

não é numérico.";
```

```
if (Vc == "" || Vc.isNumber() == false){
return "O prenchimento do valor de C está vazio ou
não é numérico.";
}
}
}
//Após essa etapa, é necessário retornar ao Java e realizar a programação de retorno dos valores para enfim, exibir ao usuário:
private void
btnCalculoActionPerformed(java.awt.event.ActionEve
nt evt) {
String Va = txtValorA.getText();
String Vb = txtValorB.getText();
String Vc = txtValorC.getText();
ClasseBLL verificacao = new ClasseBLL();
String mens = verificacao.validaDados(Va,
Vb,Vc);
if (mens == null){
ExEquaSG equacao = new
ExEquaSG(Float.parseFloat(Va),
Float.parseFloat(Vb), Float.parseFloat(Vc));
float raiz1 = equacao.calculoraiz1();
float raiz2 = equacao.calculoraiz2();
lblX1.setText("Valor da raiz 1: " + Float.toString(raiz1));
lblX2.setText("Valor da raiz 2: " +Float.toString(raiz2));
}
else {
JOptionPane.showMessageDialog(null,mens);
}
}
private void
btnLimpaActionPerformed(java.awt.event.ActionEven
t evt) {
txtValorA.setText("");
txtValorB.setText("");
```

```
txtValorC.setText("");

lblX1.setText("Valor da raiz 1: ");

lblX2.setText("Valor da raiz 2:");

}

//Após a realização de todas essas etapas o programa está pronto para execução.
```

Figura 61 – Bháskara por POO



Fonte: Própria (2021)

4 – Você é o mais novo funcionário da biblioteca do IFSP – Campus: Cubatão. No primeiro dia, foi pedido o cadastramento de um livro, para isso, construa um programa que permita tal cadastro.

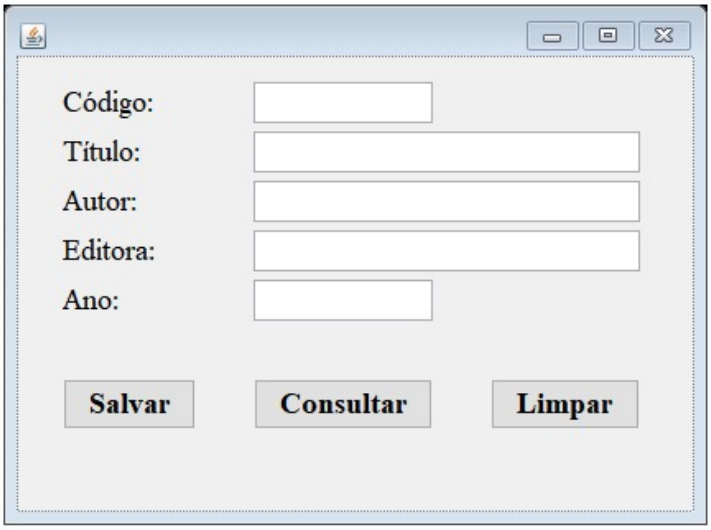
```
//Classe em Groovy para alocação das propriedades do livro:

class Livro {

String cod;

String t;

String a;

String e;

float ano;

Livro (codigo, tit, aut, ed, an){

this.cod = codigo;

this.t = tit;

this.a = aut;

this.e = ed;

this.ano = an;
```

```
}
}
//Agora, é necessário a criação de classes para tratar erros de digitação por parte do usuário.
class ClasseBLL {
public String valDados(String cod, String t,
String a, String e, String ano) {
if (cod == ""){
return "O prenchimento do código está vazio.";
}
if (t == ""){
return "O prenchimento do título está vazio.";
}
if (a == ""){
return "O prenchimento do autor está vazio.";
}
if (e == ""){
return "O prenchimento da editora está vazio.";
}
if (ano == "" || ano.isNumber() == false || (ano as float)
<= 0){
return "O prenchimento do ano está vazio, não é
numérico ou é menor que zero.";
}
}
}
//construção dos botões:
Botão salvar:
private void
bntSalvaActionPerformed(java.awt.event.ActionEvent
evt) {
String cod = txtCodigo.getText();
String t = txtTitulo.getText();
String a = txtAutor.getText();
String e = txtEditora.getText();
String ano = txtAno.getText();
```

```
ClasseBLL validacao = new ClasseBLL();
String mens = validacao.validaDados(cod,
t, a, e, ano);
if (mens == null){
livro = new Livro(cod, t, a, e,
Float.parseFloat(ano));
JOptionPane.showMessageDialog(this, "Livro
cadastrado!");
}
else {
JOptionPane.showMessageDialog(null,mens);
}
}
private void
btnConsultaActionPerformed(java.awt.event.ActionEv
ent evt) {
String cod = txtCodigo.getText();
if (livro.getCodigo().equals(cod)){
txtCodigo.setText(livro.getCodigo());
txtTitulo.setText(livro.getTitulo());
txtAutor.setText(livro.getAutor());
txtEditora.setText(livro.getEditora());
txtAno.setText(Float.toString(livro.getAno()));
} else{
JOptionPane.showMessageDialog(null,"Livro não
encontrado.");
}
}
Botão limpar
private void
btnLimpaActionPerformed(java.awt.event.ActionEvent
evt) {
txtCodigo.setText("");
txtTitulo.setText("");
```

```
txtAutor.setText("");
txtEditora.setText("");
txtAno.setText("");
}
//Após esses passos o programa estará pronto para a execução.
```

Figura 62 – Cadastro do livro por POO



Fonte: Própria (2021)

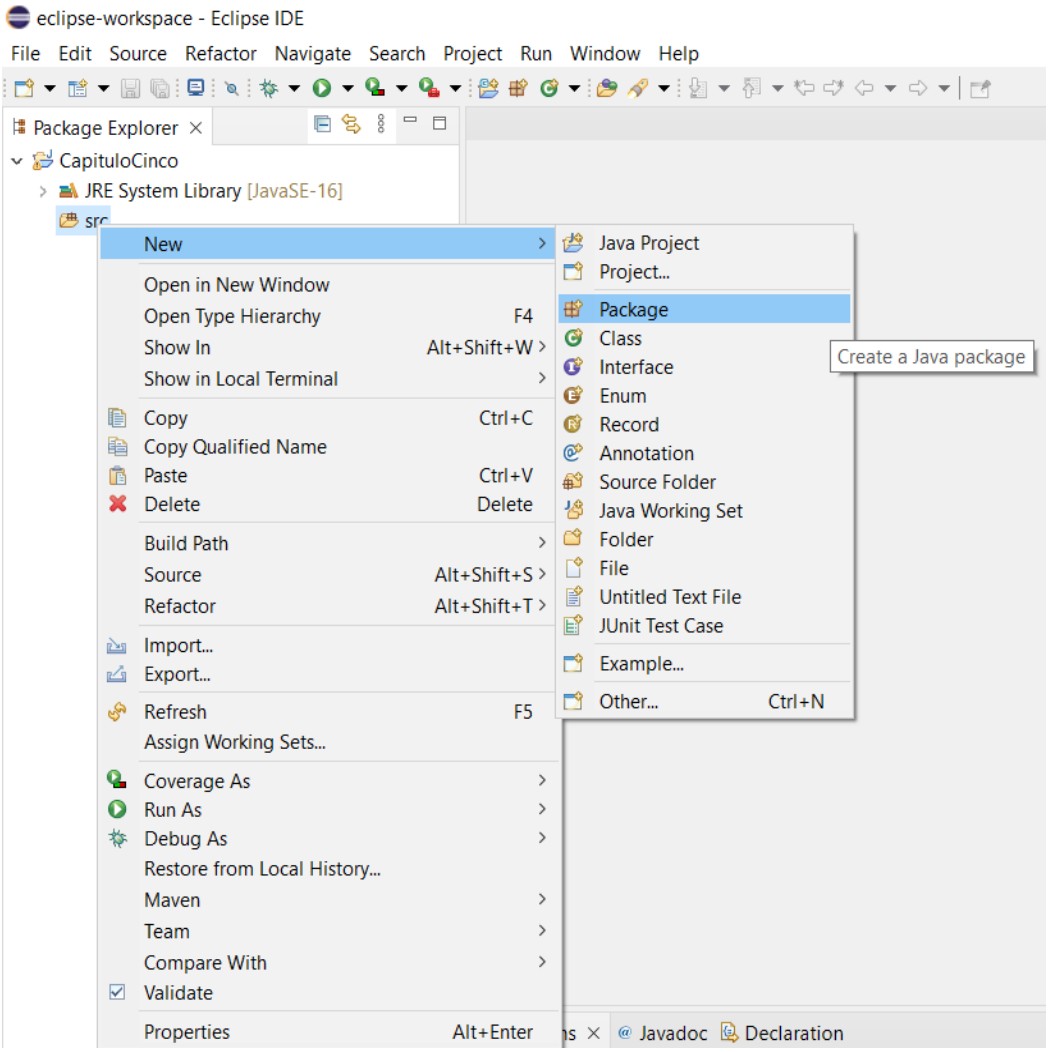
5. ASSOCIAÇÃO AO BANCO DE DADOS

Nesta seção, teremos o objetivo de criar um programa com interface gráfica – as janelas "Windows Forms" vistas no capítulo anterior – capaz de se comunicar com um banco de dados, sendo possível consultar, adicionar, alterar e excluir registros a partir dos componentes visuais. Entretanto, apenas realizar tal implementação não é suficiente: devemos seguir princípios de organização do código para evitar que a aplicação se torne desnecessariamente complexa por compactar diversas funcionalidades em um mesmo arquivo.

Inicialmente, iremos realizar a separação da parte visual do programa da seguinte forma:

Passo 1: crie um novo package na pasta src chamado view.

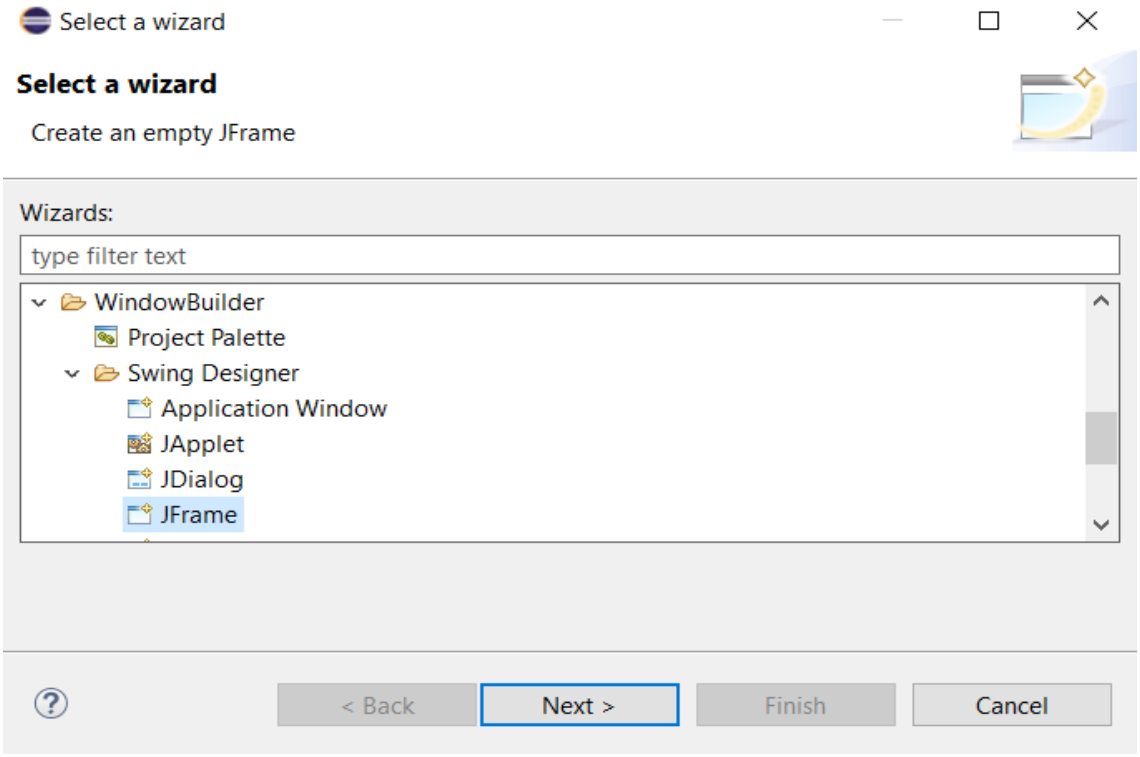
Figura 63 – Passo 1 para criação do package



Fonte: Própria (2021)

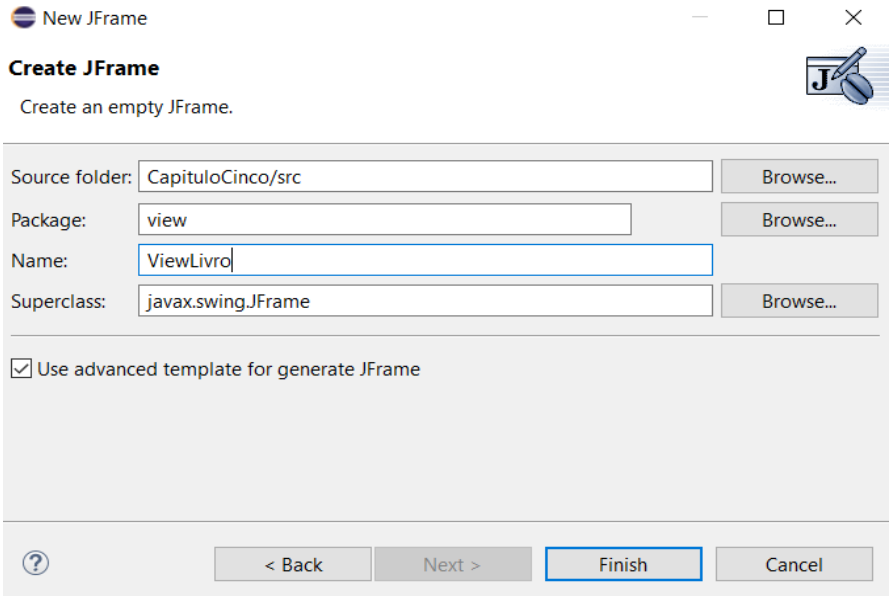
Passo 2: com o package "view" criado, clique sobre ele; pressione CTRL + N; Clique em Window Builder > Swing Designer, selecione JFrame e clique em Next.

Figura 64 – Passo 2 para criação do package



Fonte: Própria (2021)

Figura 65 – Passo 3 para criação do package



Fonte: Própria (2021)

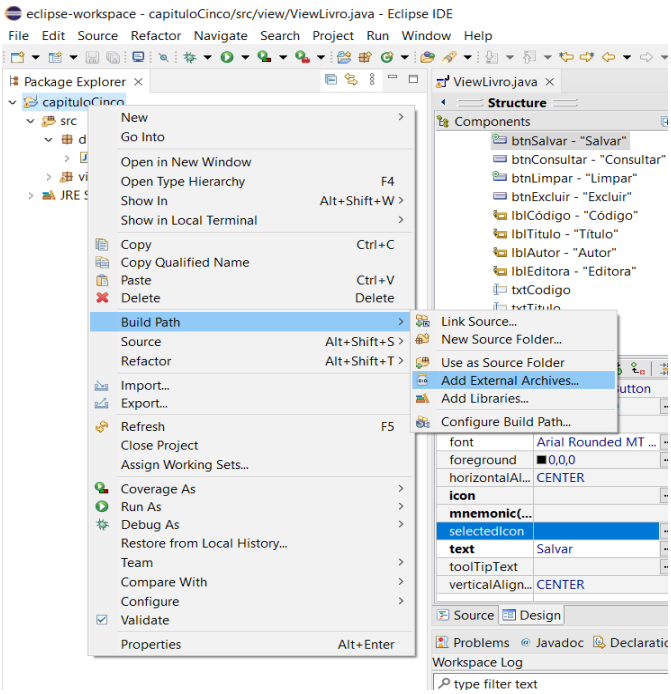
Agora, basta criar na aba “design” o formulário conforme visto anteriormente, e dar procedência na criação de um novo package chamado “bd”, responsável pelos códigos que possuem alguma relação com o banco de dados.

Antes de desenvolvermos os códigos que irão compor esse novo pacote, é necessário instalar a dependência do JDBC, que, em tradução literal, é uma sigla para “Conectividade Java com Banco de Dados”. Logo, é uma ferramenta que reúne classes e interfaces escritas em Java que possibilita a conexão com um banco de dados. Para implementá-lo, antes é preciso saber com qual banco de dados sua aplicação irá interagir. Nesse caso, utilizaremos o MySQL e, portanto, devemos baixar um conector JDBC compatível com o MySQL, que você poderá acessar clicando aqui.

Feito o download, basta descompactar a pasta e seguir os passos a seguir:

Passo 1: clique com o botão direito na pasta do projeto e vá em Build Path > Add External Archives.

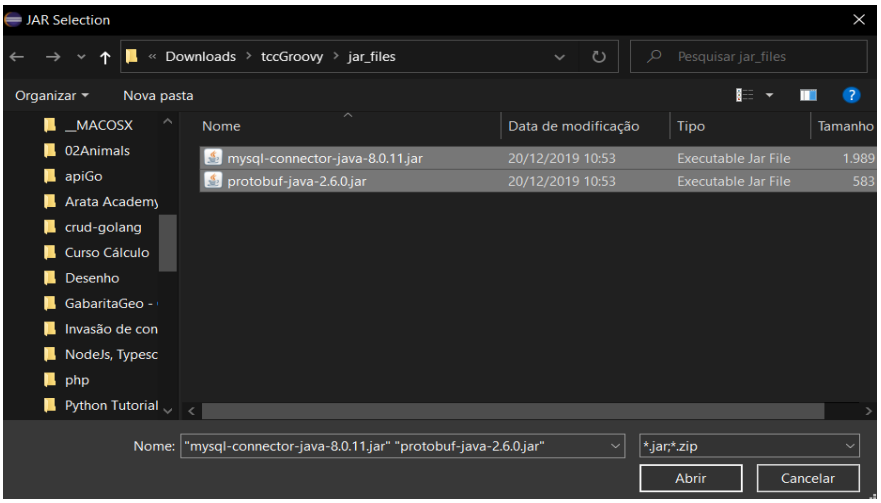
Figura 66 – Passo 1 para inserção do conector



Fonte: Própria (2021)

Passo 2: procure pelos arquivos do conector MySQL com a extensão .jar e selecione-os; clique em abrir.

Figura 67 – Passo 2 para inserção do conector



Fonte: Fonte: Própria (2021)

Feita a inserção do conector, é preciso criar uma classe “ConexaoBanco” no package “bd” responsável por criar uma conexão com o banco.

```
package db;

//Importa todas as classes do java que lidam com SQL

import java.sql.*;

import javax.swing.JOptionPane;

import erro.Erro;

public class ConexaoBanco {

	//Caminho do banco de dados
	private static final String DB_PATH =
"jdbc:mysql://localhost:3306/livraria?serverTimezone=UTC&useSSL=false";

	//Nome do usuário mysql
	private static final String USUARIO = "root";

	//Senha do banco
	private static final String SENHA = "root";


	/*

	 * Retorna uma conexão com o banco de dados

	 */

	public static Connection getConnection() throws Exception {
				Connection con = DriverManager.getConnection(
						DB_PATH,
						USUARIO,
						SENHA
				);
			return con;
	}
}
```

Em seguida, utilizaremos novamente a classe erro, dentro de um package chamado “erro”:

```
package erro;

public class Erro {
	private static String msg;
	private static boolean erro;

	public static String getMsg() {
		return msg;
	}
	public static void setMsg(String msg) {
		erro = true;
		Erro.msg = msg;
	}
	public static boolean getErro() {
		return erro;
	}
	public static void setErro(boolean erro) {
		Erro.erro = erro;
	}

}
```

Por fim, criaremos três classes dentro de um package chamado “livro”:

```
package livro;

public class Livro {
	private String codigo;
	private String titulo;
	private String autor;
	private String editora;
	private String ano;

	public String getCodigo() {
		return codigo;
	}
	public void setCodigo(String codigo) {
		this.codigo = codigo;
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public String getAutor() {
		return autor;
	}
	public void setAutor(String autor) {
		this.autor = autor;
	}
	public String getEditora() {
		return editora;
	}
	public void setEditora(String editora) {
		this.editora = editora;
	}
	public String getAno() {
		return ano;
	}
	public void setAno(String ano) {
		this.ano = ano;
	}
}
```

Livro.java (é uma classe que “reflete” como os registros estão no banco de dados, com cada campo condizendo a uma coluna existente).

LivroBLL.java

```
package livro;

import erro.Erro;

public class LivroBLL {

	public static void conecta()
	{
		LivroDAL.conecta();
	}

	public static void validaCodigo(Livro livro, String op)
	{
		Erro.setErro(false);
		if(livro.getCodigo() == null) {
			Erro.setMsg("O código é de preenchimento obrigatório!");
			return;
		}
		if (op.equals("c"))
			LivroDAL.consultaUmLivro(livro);
		else
			LivroDAL.excluiUmLivro(livro);
	}

	public static void validaDados(Livro livro, String op) {
		if(livro.getCodigo().equals("")) {
			Erro.setMsg("O código é de preenchimento obrigatório!");
			return;
		}
		if(livro.getTitulo().equals("")) {
			Erro.setMsg("O título é de preenchimento obrigatório!");
			return;
		}
		if(livro.getAutor().equals("")) {
			Erro.setMsg("O autor é de preenchimento obrigatório!");
			return;
		}
		if(livro.getEditora().equals("")) {
			Erro.setMsg("A editora é de preenchimento obrigatório!");
			return;
		}
		if(livro.getAno().equals("")) {
			Erro.setMsg("O ano é de preenchimento obrigatório!");
			return;
		}

		try {
			Integer.parseInt(livro.getAno());
		} catch (Exception e)
		{
			Erro.setMsg("O valor do ano deve ser numérico!");
			return;
		}

		if(Integer.parseInt(livro.getAno()) <= 0) {
			Erro.setMsg("O valor do ano deve ser numérico e positivo!");
		}
		if(op.equals("i"))
			LivroDAL.inseriUmLivro(livro);
		else
			LivroDAL.atualizaUmLivro(livro);
	}
}
```

LivroDAL.java

```
package livro;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import db.ConexaoBanco;
import erro.Erro;

public class LivroDAL {
	private static Connection con;

	public static void conecta() {
		try {
			con = ConexaoBanco.getConnection();
		} catch(Exception e) {
			Erro.setMsg("Problemas ao conectar-se com o banco de dados!");
		}
	}

	public static void inseriUmLivro(Livro livro) {
		String sql = "INSERT INTO livro(codigo, titulo, autor, editora, ano) VALUES (?, ?, ?, ?, ?)";

		try {
			PreparedStatement stmt = con.prepareStatement(sql);
			stmt.setString(1, livro.getCodigo());
			stmt.setString(2, livro.getTitulo());
			stmt.setString(3, livro.getAutor());
			stmt.setString(4, livro.getEditora());
			stmt.setInt(5, Integer.parseInt(livro.getAno()));
			stmt.execute();
			con.close();
		} catch (Exception e) {
			Erro.setMsg("Erro ao inserir registro no banco de dados!");
		}
	}

	public static void excluiUmLivro(Livro livro) {
		String sql = "DELETE FROM livro WHERE codigo = ?";
		try {
			PreparedStatement stmt = con.prepareStatement(sql);
			stmt.setString(1, livro.getCodigo());
			stmt.execute();
			con.close();
		} catch (Exception e) {
			Erro.setMsg("Erro ao excluir o registro no banco de dados!");
		}
	}

	public static void atualizaUmLivro(Livro livro) {
		String sql = "UPDATE livro SET titulo = ?, autor = ?, editora = ?, ano = ? WHERE codigo = ?";

		try {
			PreparedStatement stmt = con.prepareStatement(sql);
			stmt.setString(1, livro.getTitulo());
			stmt.setString(2, livro.getAutor());
			stmt.setString(3, livro.getEditora());
			stmt.setInt(4, Integer.parseInt(livro.getAno()));
			stmt.setString(5, livro.getCodigo());
			stmt.execute();
			con.close();
		} catch (Exception e) {
			Erro.setMsg("Erro ao atualizar o registro no banco de dados!");
		}
	}

	public static void consultaUmLivro(Livro livro) {
		String sql = "SELECT * FROM livro WHERE codigo = ?";

		try {
			PreparedStatement stmt = con.prepareStatement(sql);
			stmt.setString(1, livro.getCodigo());
			ResultSet resultado = stmt.executeQuery();
			if(resultado.first()) {
```

```
livro.setTitulo(resultado.getString(2));
                                        livro.setAutor(resultado.getString(3));
                                        livro.setEditora(resultado.getString(4));
                                        livro.setAno(resultado.getString(5));
                                }
        }

}
```

Com a lógica da aplicação pronta e descentralizada, basta adicionar as funcionalidades aos botões do formulário, que irá ficar da seguinte forma:

```
package view;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import db.ConexaoBanco;
import erro.Erro;
import livro.Livro;
import livro.LivroBLL;

import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class ViewLivro extends JFrame {

        private JPanel contentPane;
        private JTextField txtCodigo;
        private JTextField txtTitulo;
        private JTextField txtAutor;
        private JTextField txtEditora;
        private JTextField txtAno;


        public static void main(String[] args) {
                EventQueue.invokeLater(new Runnable() {
                        public void run() {
                                try {
                                        ViewLivro frame = new ViewLivro();
                                        frame.setVisible(true);

                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }
```

```
public ViewLivro() {

		LivroBLL.conecta();
		Livro livro = new Livro();
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 655, 361);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);

		contentPane.setLayout(null);
		txtCodigo = new JTextField();
		txtCodigo.setFont(new Font("Arial", Font.PLAIN, 14));
		txtCodigo.setBounds(274, 16, 227, 30);
		contentPane.add(txtCodigo);
		txtCodigo.setColumns(10);

		txtTitulo = new JTextField();
		txtTitulo.setFont(new Font("Arial", Font.PLAIN, 14));
		txtTitulo.setColumns(10);
		txtTitulo.setBounds(274, 60, 227, 30);
		contentPane.add(txtTitulo);

		txtAutor = new JTextField();
		txtAutor.setFont(new Font("Arial", Font.PLAIN, 14));
		txtAutor.setColumns(10);
		txtAutor.setBounds(274, 109, 227, 30);
		contentPane.add(txtAutor);

		txtEditora = new JTextField();
		txtEditora.setFont(new Font("Arial", Font.PLAIN, 14));
		txtEditora.setColumns(10);
		txtEditora.setBounds(274, 153, 227, 30);
		contentPane.add(txtEditora);

		JButton btnSalvar = new JButton("Salvar");
		btnSalvar.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 13));
		btnSalvar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				livro.setCodigo(txtCodigo.getText());
				livro.setTitulo(txtTitulo.getText());
				livro.setAutor(txtAutor.getText());
				livro.setEditora(txtEditora.getText());
				livro.setAno(txtAno.getText());
				LivroBLL.validaDados(livro, "i");
				if(Erro.getErro())
					JOptionPane.showMessageDialog(null, Erro.getMsg());
				else
					JOptionPane.showMessageDialog(null, "Livro inserido!");
			}
		});
```

```
btnSalvar.setBounds(24, 268, 102, 34);
		contentPane.add(btnSalvar);

		JButton btnConsultar = new JButton("Consultar");
		btnConsultar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				livro.setCodigo(txtCodigo.getText());
				LivroBLL.validaCodigo(livro, "c");
				if(Erro.getErro())
					JOptionPane.showMessageDialog(null, Erro.getMsg());
				else {
					txtTitulo.setText(livro.getTitulo());
					txtAutor.setText(livro.getAutor());
					txtEditora.setText(livro.getEditora());
					txtAno.setText(livro.getAno());
				}

			}
		});
		btnConsultar.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 13));
		btnConsultar.setBounds(147, 268, 102, 34);
		contentPane.add(btnConsultar);

		JButton btnLimpar = new JButton("Limpar");
		btnLimpar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				txtCodigo.setText("");
				txtTitulo.setText("");
				txtAutor.setText("");
				txtEditora.setText("");
				txtAno.setText("");
				txtCodigo.requestFocus();
			}
		});
		btnLimpar.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 13));
		btnLimpar.setBounds(271, 268, 102, 34);
		contentPane.add(btnLimpar);

		JButton btnExcluir = new JButton("Excluir");
		btnExcluir.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				livro.setCodigo(txtCodigo.getText());
				LivroBLL.validaCodigo(livro,"e");
				if(Erro.getErro())
					JOptionPane.showMessageDialog(null, Erro.getMsg());
				else
					JOptionPane.showMessageDialog(null, "Livro excluído!");
			}
		});
```

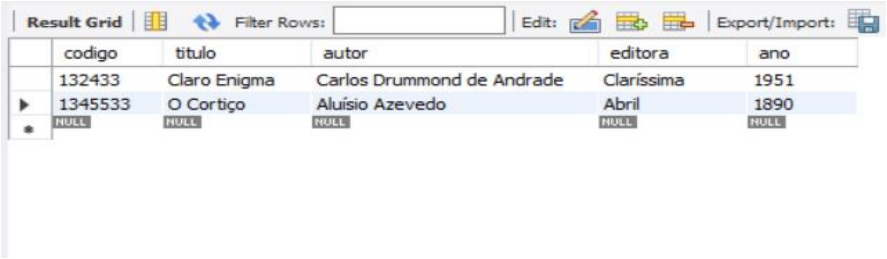
```
btnExcluir.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 13));
                btnExcluir.setBounds(529, 268, 102, 34);
                contentPane.add(btnExcluir);

                JLabel lblCódigo = new JLabel("C\u00F3digo:");
                lblCódigo.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 17));
                lblCódigo.setBounds(152, 10, 102, 34);
                contentPane.add(lblCódigo);

                JLabel lblTitulo = new JLabel("T\u00EDtulo:");
                lblTitulo.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 17));
                lblTitulo.setBounds(152, 54, 102, 34);
                contentPane.add(lblTitulo);

                JLabel lblAutor = new JLabel("Autor:");
                lblAutor.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 17));
                lblAutor.setBounds(152, 103, 102, 34);
                contentPane.add(lblAutor);

                JLabel lblEditora = new JLabel("Editora:");
                lblEditora.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 17));
                lblEditora.setBounds(152, 149, 102, 34);
                contentPane.add(lblEditora);

                JLabel lblAno = new JLabel("Ano:");
                lblAno.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 17));
                lblAno.setBounds(152, 200, 102, 34);
                contentPane.add(lblAno);

                txtAno = new JTextField();
                txtAno.setFont(new Font("Arial", Font.PLAIN, 14));
                txtAno.setColumns(10);
                txtAno.setBounds(274, 204, 227, 30);
                contentPane.add(txtAno);

                JButton btnAlterar = new JButton("Alterar");
                btnAlterar.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                livro.setCodigo(txtCodigo.getText());
                                livro.setTitulo(txtTitulo.getText());
                                livro.setAutor(txtAutor.getText());
                                livro.setEditora(txtEditora.getText());
                                livro.setAno(txtAno.getText());
                                LivroBLL.validaDados(livro, "a");
                                if(Erro.getErro())
                                        JOptionPane.showMessageDialog(null, Erro.getMsg());
                                else
                                        JOptionPane.showMessageDialog(null, "Livro alterado!");
                        }
                });
                btnAlterar.setFont(new Font("Arial Rounded MT Bold", Font.PLAIN, 13));
                btnAlterar.setBounds(399, 268, 102, 34);
                contentPane.add(btnAlterar);

        }
}
```

Observe o funcionamento do programa:

Figura 68 – Programa “Livraria”

Fonte: Própria (2021)

Figura 69 – Dados inseridos no banco “livraria”



Fonte: Própria (2021)

5.1. APIs

Para finalizarmos o conteúdo da apostila, falta abordar um assunto fundamental: APIs. É através desse acrônimo para Application Programming Interface (interface de programação de aplicação) que podemos realizar a comunicação entre softwares distintos.

Em programação orientada à objetos, o conceito de interface é exemplificado como sendo um “contrato”, que fornece as especificações que uma determinada classe deve seguir para ser implementada. Nas APIs, isso não é diferente: elas contemplam definições e protocolos utilizados no desenvolvimento e na integração de software de aplicações. Com isso, é possível criar programas sem saber como outros produtos e serviços foram implementados, bastando seguir o que foi definido pela API que permite a comunicação com ele.

No exemplo a seguir, iremos utilizar a API “ViaCEP”, um serviço que nos fornece dados de localização a partir de um CPF informado da seguinte forma:

viacep.com.br/ws/ 01001-000/json/

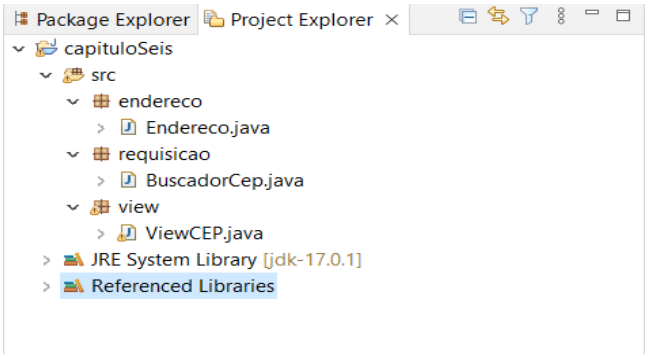
Ao digitar essa URL, iremos obter os dados referentes ao CEP “01001-000” em um JSON (JavaScript Object Notation), um formato de texto que estrutura as informações de forma a facilitar a transmissão de um sistema para o outro:

```
{
    "cep": "01001-000",
    "logradouro": "Praça da Sé",
    "complemento": "lado ímpar",
    "bairro": "Sé",
    "localidade": "São Paulo",
    "uf": "SP",
    "ibge": "3550308",
    "gia": "1004",
    "ddd": "11",
    "siafi": "7107"
}
```


Além disso, também é possível utilizar uma biblioteca desenvolvida pelo Google chamada gson, capaz de converter uma estrutura json diretamente para um objeto, tendo seus campos condizentes com os nomes fornecidos pelo json. Ex: objeto “Endereco” com os campos logradouro, complemento e bairro, recebendo os dados “Praça da Sé”, “lado ímpar” e “bairro” através do código da biblioteca gson.

Para começar, iremos criar o projeto “capituloSeis”, contendo os packages “endereco”, “requisicao” e “view” dentro da pasta “src”, seguindo os mesmos princípios de organização do código apresentados no capítulo anterior, e as classes “Endereco”, “BuscadorCep” e “ViewCep”, ficando da seguinte forma:

Figura 70 – Exibição da estrutura de pastas do projeto “capituloSeis”



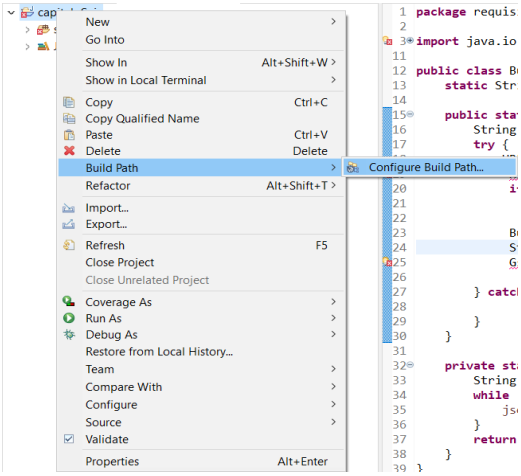
Fonte: Própria (2021)

Agora, deve-se importar a classe gson dentro do projeto, sendo necessário fazer o download do arquivo por este link.

Feito o download, siga os passos a seguir:

Passo 1: clique com o botão direito na pasta do projeto e selecione Build Path > Configure Build Path.

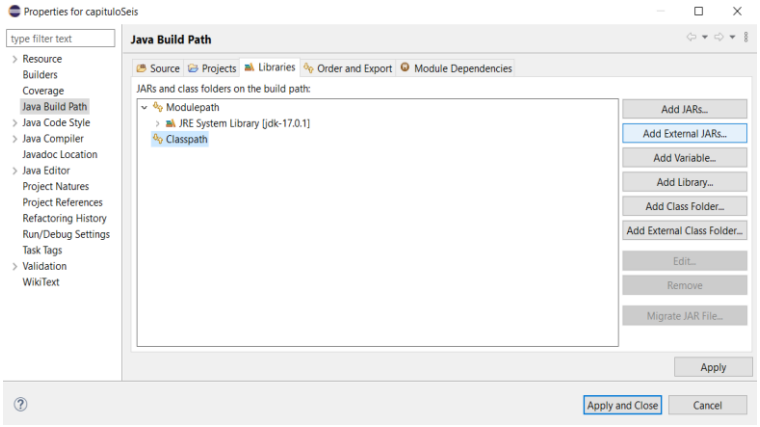
Figura 71 – Exibição do Passo 1 para importar a classe gson



Fonte: Própria (2021)

Passo 2: selecione “Classpath” e clique em “Add External JARs”.

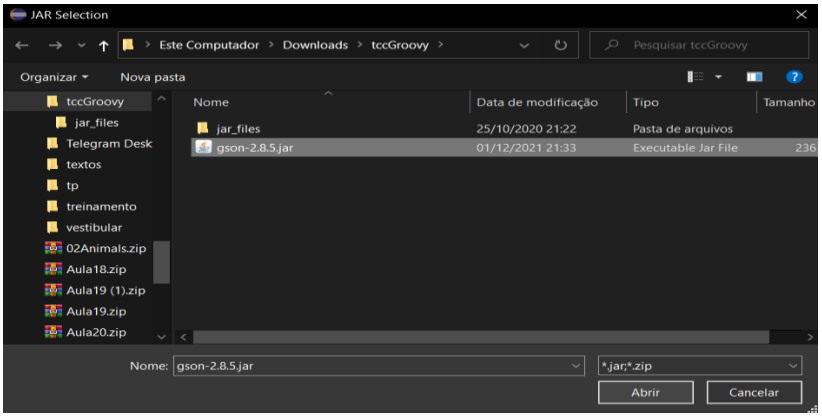
Figura 72 – Exibição do Passo 2 para importar a classe gson



Fonte: Própria (2021)

Passo 3: selecione o arquivo “gson-2.8.5.jar”, clique em “Abrir” e em “Apply and Close”.

Figura 73 – Exibição do Passo 3 para importar a classe gson



Fonte: Própria (2021)

Após a importação, basta inserir os seguintes códigos:

Endereco.java

```
package endereco;


public class Endereco {
            String logradouro;
            String bairro;
            String localidade;
            String uf;

    public String getLogradouro() {
        return logradouro;
    }

    public String getBairro() {
        return bairro;
    }

    public String getLocalidade() {
        return localidade;
    }

    public String getUf() {
        return uf;
    }
}
```

BuscadorCep.java

```
package requisicao;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

import com.google.gson.Gson;


import endereco.Endereco;


public class BuscadorCep {

	static String urlServico = "http://viacep.com.br/ws/";


	public static Endereco buscarEndereco(String cep)  {

		// Caso o CEP informado não esteja no formato correto, retornará nulo

		if (!validarCEP(cep))

			return null;

		// formato "http://viacep.com.br/ws/numeroCEP/json/unicode

		String urlRequisicao = urlServico + cep + "/json/unicode";

		try {

			URL url = new URL(urlRequisicao);

			HttpURLConnection conexao = (HttpURLConnection) url.openConnection();

			// Caso a requisição HTTP não seja bem-sucedida

			if (conexao.getResponseCode() != 200)

				throw new RuntimeException("Erro na conexão HTTP: " +
conexao.getResponseCode());

			// Leitura dos dados recebidos pela requisição

			BufferedReader resposta = new BufferedReader(new
InputStreamReader((conexao.getInputStream())));

			String jsonEmString = converterJsonParaString(resposta);

			// Uso da biblioteca gson para mapear os dados json com os atributos do
objeto endereco

			Gson gson = new Gson();

		Endereco endereco = gson.fromJson(jsonEmString, Endereco.class);
```

```
        // Fechamento de conexões
                                resposta.close();
                                conexao.disconnect();


                                return endereco;
                        } catch (Exception e) {
                                return null;
                        }
                }
                private static boolean validarCEP(String cep) {
                        /*
                         * Valida o CEP através do regex, uma espécie de "padrão de string" que permite
                        avaliar se uma sequência de caracteres possui determinadas características.
                         */
                        return cep.matches("(^[0-9]{5})-?([0-9]{3}$)");
                }
                private static String converterJsonParaString(BufferedReader leitor) throws IOException {
                        String resposta, jsonEmString = "";
                        while ((resposta = leitor.readLine()) != null) {
                                jsonEmString += resposta;
                        }
                        return jsonEmString;
                }
        }
```

ViewCEP.java

```
package view;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import endereco.Endereco;

import requisicao.BuscadorCep;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import javax.swing.JTextField;

import javax.swing.JFormattedTextField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.Color;


public class ViewCEP extends JFrame {

	private JPanel contentPane;

	private JTextField txtCep;

	private JTextField txtLogradouro;

	private JTextField txtBairro;

	private JTextField txtLocalidade;

	private JTextField txtUF;


	public static void main(String[] args) {

		EventQueue.invokeLater(new Runnable() {
```

```
	public void run() {
					try {
						ViewCEP frame = new ViewCEP();
						frame.setVisible(true);
					} catch (Exception e) {
						e.printStackTrace();
					}
				}
			});
		}
		public ViewCEP() {
			setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			setBounds(100, 100, 588, 524);
			contentPane = new JPanel();
			contentPane.setForeground(Color.GRAY);
			contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
			setContentPane(contentPane);
			contentPane.setLayout(null);

			JLabel lblTitulo = new JLabel("Servi\u00E7o de CEP");
			lblTitulo.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 50));
			lblTitulo.setBounds(151, 10, 315, 81);
			contentPane.add(lblTitulo);

			JLabel lblCEP = new JLabel("Digite um CEP:");
			lblCEP.setFont(new Font("Tahoma", Font.PLAIN, 24));
			lblCEP.setBounds(72, 101, 174, 43);
			contentPane.add(lblCEP);

			txtCep = new JTextField();
			txtCep.setFont(new Font("Tahoma", Font.PLAIN, 14));
			txtCep.setBounds(256, 101, 232, 37);
			contentPane.add(txtCep);
```

```
		txtCep.setColumns(10);

                        JButton btnPesquisar = new JButton("Pesquisar");

                        btnPesquisar.setFont(new Font("Tw Cen MT Condensed", Font.PLAIN, 32));

                        btnPesquisar.addActionListener(new ActionListener() {

                                public void actionPerformed(ActionEvent e) {

                                        Endereco endereco =
BuscadorCep.buscarEndereco(txtCep.getText());

                                        if(endereco == null)

                                                JOptionPane.showMessageDialog(null, "CEP inválido ou não
foi possível conectar-se com a API!");

                                        else {

                                                txtLogradouro.setText(endereco.getLogradouro());

                                                txtBairro.setText(endereco.getBairro());

                                                txtLocalidade.setText(endereco.getLocalidade());

                                                txtUF.setText(endereco.getUf());

                                        }

                                }

                        });

                        btnPesquisar.setBounds(66, 384, 183, 60);

                        contentPane.add(btnPesquisar);


                        JLabel lblLogradouro = new JLabel("Logradouro:");

                        lblLogradouro.setFont(new Font("Tahoma", Font.PLAIN, 24));

                        lblLogradouro.setBounds(72, 152, 174, 43);

                        contentPane.add(lblLogradouro);


                        JLabel lblBairro = new JLabel("Bairro:");

                        lblBairro.setFont(new Font("Tahoma", Font.PLAIN, 24));

                        lblBairro.setBounds(72, 205, 174, 43);

                        contentPane.add(lblBairro);


                        JLabel lblLocalidade = new JLabel("Localidade:");

                        lblLocalidade.setFont(new Font("Tahoma", Font.PLAIN, 24));

                        lblLocalidade.setBounds(72, 258, 174, 43);

                        contentPane.add(lblLocalidade);
```

```
JLabel lblUF = new JLabel("UF:");
                        lblUF.setFont(new Font("Tahoma", Font.PLAIN, 24));
                        lblUF.setBounds(72, 311, 174, 43);
                        contentPane.add(lblUF);

                        txtLogradouro = new JTextField();
                        txtLogradouro.setFont(new Font("Tahoma", Font.PLAIN, 14));
                        txtLogradouro.setColumns(10);
                        txtLogradouro.setBounds(256, 158, 232, 37);
                        contentPane.add(txtLogradouro);

                        txtBairro = new JTextField();
                        txtBairro.setFont(new Font("Tahoma", Font.PLAIN, 14));
                        txtBairro.setColumns(10);
                        txtBairro.setBounds(256, 211, 232, 37);
                        contentPane.add(txtBairro);

                        txtLocalidade = new JTextField();
                        txtLocalidade.setFont(new Font("Tahoma", Font.PLAIN, 14));
                        txtLocalidade.setColumns(10);
                        txtLocalidade.setBounds(256, 264, 232, 37);
                        contentPane.add(txtLocalidade);

                        txtUF = new JTextField();
                        txtUF.setFont(new Font("Tahoma", Font.PLAIN, 14));
                        txtUF.setColumns(10);
                        txtUF.setBounds(256, 317, 232, 37);
                        contentPane.add(txtUF);

                        JButton btnLimpar = new JButton("Limpar");
                        btnLimpar.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        txtCep.setText("");
                                        txtLogradouro.setText("");
                                        txtBairro.setText("");
```

```
txtLocalidade.setText("");
                                        txtUF.setText("");
                                        txtCep.requestFocus();
                                }
                        });
                        btnLimpar.setFont(new Font("Tw Cen MT Condensed", Font.PLAIN, 32));
                        btnLimpar.setBounds(305, 384, 183, 60);
                        contentPane.add(btnLimpar);
                }
}
```

Figura 74 – Demonstração do programa “Serviço de CEP”

Fonte: Própria (2021)

REFERÊNCIAS

O que é Groovy?. Disponível em: <https://www.devmedia.com.br/linguagem-de-programacao-groovy-introducao/34099/>. Acesso em: 09 mai. 2021.
DEVMEDIA

Mazzega Menegucci, Lucas; Gobbi de Angeli, Rodolfo; Natale, Ricardo. **Desenvolvimento ágil para plataforma Java**. Disponível em: <http://www.inf.ufes.br/~vitorsouza/archive/2020/wp-content/uploads/teaching-lp-20132-seminario-groovy.pdf/>. Acesso em: 10 mai. 2021.

PAULA ROSA, de Renan; VICENTE RONDON MACHADO, Robinson. **SISTEMA PARA CLASSIFICAÇÃO DE ESTUDANTES À SEREM CONTEMPLADOS NO PROGRAMA BOLSA-PERMANÊNCIA DA UTFPR UTILIZANDO FRAMEWORK GRAILS E IREPORT: DESENVOLVIMENTO E PROPOSTA DE IMPLANTAÇÃO.** Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6433/1/PG_COADS_2012_2_15.pdf/>. Acesso em: 10 mai. 2021.

Assistência de código no editor Java do NetBeans IDE: um guia de referência. Disponível em: <https://NetBeans.apache.org/kb/docs/java/editor-codereference.html/>. Acesso em: 10 mai. 2021.
APACHE NETBEANS

Restruturação. Disponível em: <https://refactoring.guru/pt-br/refactoring/>. Acesso em: 11 mai. 2021.
REFACTPRING GURU

FOWLER, Martin. **Refactoring.com.** Disponível: <https://refactoring.com/>. Acesso em: 15 mai. 2021.

PRACIANO, Elias. **Como destacar a sintaxe do seu código em um blog wordpress.** Disponível: <https://elias.praciano.com/2016/07/como-aplicar-destacar-a-sintaxe-do-codigo-em-um-blog-wordpress/#:~:text=O%20uso%20de%20realce%20de,programas%20ou%20comandos%20mais%20f%C3%A1cil/> Acesso em: 15 mai. 2021.

13 Best Groovy IDE and Editor. Disponível em: <https://www.dunebook.com/best-groovy-ide-and-editor/> Acesso em: 15 mai. 2021.
DUNEBOOK

CANTANZARO, Carlos Sabo. **Visual Studio Code o novo editor de texto queridinho dos Desenvolvedores.** Disponível em: <https://medium.com/@wtricks/visual-studio-code-o-novo-editor-de-texto-queridinho-dos-desenvolvedores-2fb3b59182c#:~:text=–%20O%20Visual%20Studio%20Code%20é,entre%20outras%20linguagem%20mais%20robustas/> Acesso em: 10 jun. 2021.

Rogério Carvalho Filho, Cláudio. **Estrutura de repetição for em Java**. Disponível em: <http://excript.com/java/estrutura-repeticao-for-java.html/>. Acesso em: 20 jun. 2021.

Furtado de Oliveira Alves, Gustavo. **O que são vetores e matrizes (arrays)**. Disponível em: <https://dicasdeprogramacao.com.br/o-que-sao-vetores-e-matrizes-arrays/>. Acesso em: 20 jun. 2021.

O que é API?. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces/>. Acesso em: 12 nov. 2021.
RED HAT

Acessando o webservice de CEP. Disponível em: <https://viacep.com.br/>. Acesso em: 20 nov. 2021.
VIACEP

ALMEIDA, Arthur dos Santos. **Consumindo uma API de maneira simples com Java.** Disponível em: <https://arthur-almeida.medium.com/consumindo-uma-api-de-maneira-simples-com-java-2a386010e4b9/>. Acesso em: 20 nov. 2021.