Lógica de Programação

Unidade 6.1 – Classe Main

QI ESCOLAS E FACULDADES
Curso Técnico em Informática

SUMÁRIO

CRIAÇÃO DA CLASSE PRINCIPAL

Um projeto em Java pode ser composto por diversas classes. Mas uma delas deve ser responsável pela execução do programa, um ponto de partida. Um início.

Normalmente chamamos essa classe de Programa, Principal, Main, Teste, etc. Essa classe não possui atributos, e normalmente possui apenas um método chamado de ***main*** (principal, em inglês).

Nesta classe inserimos os comandos para o programa "funcionar". Esta é a classe que será "executada". E que nos retornará finalmente algo na tela.

Sintaxe da Classe Main

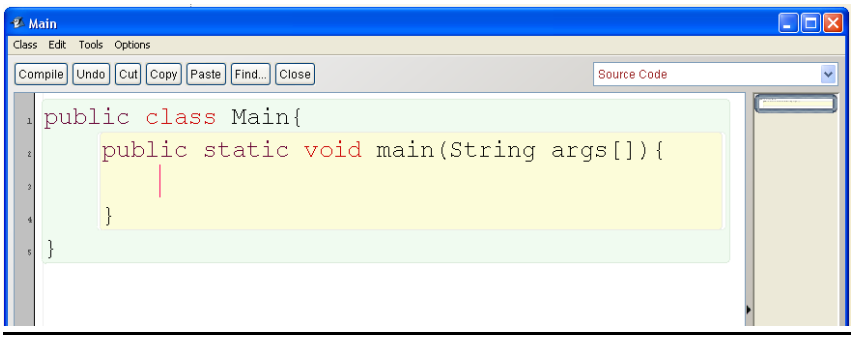


Figura 1

Toda a classe Principal possuirá uma linha com o seguinte comando: **public static void main(String args[])** - Este comando especifica que a classe é estática, sem retorno e principal. Abaixo deste comando devemos escrever todos os passos que devem ser seguidos pelo programa.

Exemplo da Classe Main

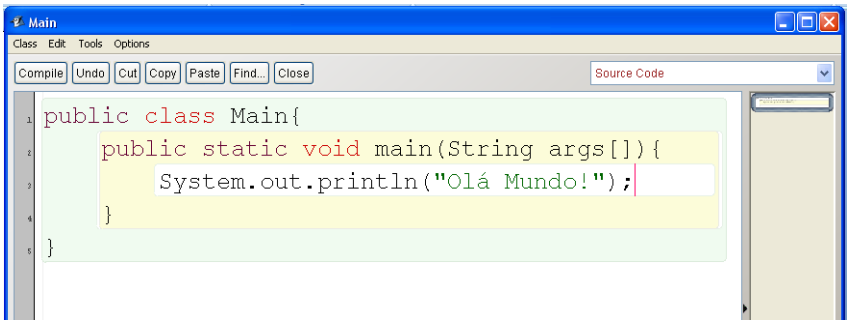


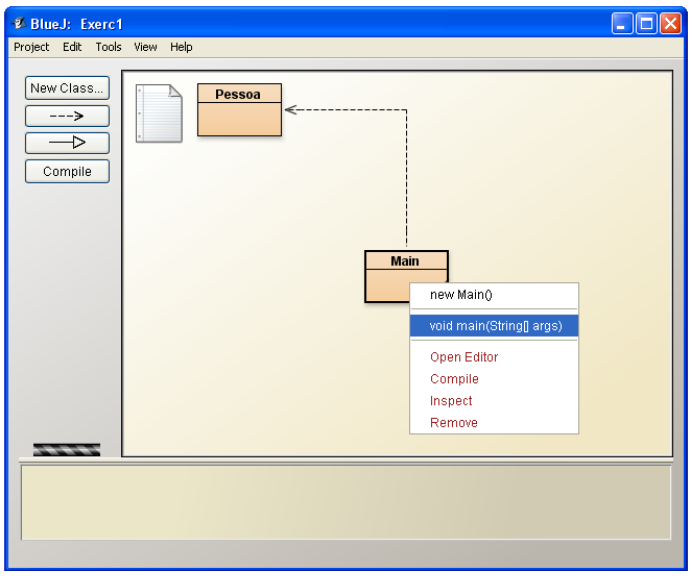
Figura 2

No exemplo da figura 2, nosso programa irá mostrar na tela a mensagem "Olá Mundo", pois é o único comando encontrado no método ***main***.

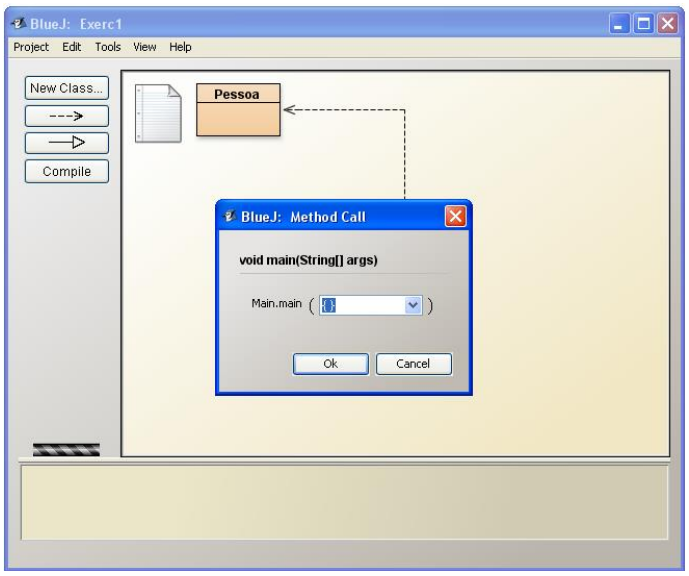
Executando a classe

Para executarmos o programa e poder testá-lo, verificar o que está ocorrendo no nosso projeto, devemos:

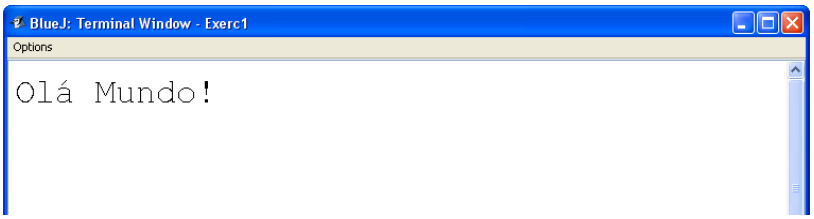
1. Clicar com o botão direito na classe Main – Clicamos em **void main(String[] args)**



2. Após aparecerá uma tela de confirmação, pressionamos Ok.



3. O resultado será uma tela (terminal) com a frase "Olá Mundo!"



Fluxo de Operações do Main

Podemos estabelecer o seguinte fluxo, ordem na classe ***Main***:

→ Criação dos objetos

→ Definição dos dados (entrada)

→ Processamento (execução dos métodos de cálculos, por exemplo)

→ Exibição dos resultados (saída para o usuário)

MÉTODOS DE SAÍDA DE DADOS (IMPRESSÃO NA TELA)

Como podemos perceber na tela acima, temos a possibilidade de mandar mensagens ao usuário, essas mensagens podem conter somente texto, somente respostas, textos e respostas, texto, respostas e mais texto...

Isso é possível através de um método de saída de dados, o método **out**, este método é da classe do sistema, ***class System***, portanto o comando fica: **System.out.print("mensagem");**

Utilizando o Método de Saída de Dados

Para utilizar este método basta incorporá-lo na classe ***Main*** ou nos métodos das nossas classes, como métodos de respostas, por exemplo.

print X println

Podemos utilizar nossos métodos de saída com ***print*** ou ***println***, a diferença é que ao utilizarmos ***print***, o final da mensagem ficará junto com a próxima mensagem.

Exemplo Sintaxe com print:

```
System.out.print("Olá! ");
System.out.print("Sou aluno de Lógica.");
```

Saída no Terminal:

```
Olá! Sou aluno de Lógica.
```

Já ao utilizarmos ***println***, a próxima mensagem ficará abaixo da anterior, ou seja, o ***ln*** represente *line* = linha, uma nova linha no terminal.

Exemplo Sintaxe com println:	**Exemplo no Terminal:**
System.out.println("Olá! ");	Olá!
System.ou.println("Sou aluno de Lógica.");	Sou aluno de Lógica.

**Imprimindo apenas textos*

```
System.out.println("Olá pessoal, sejam bem vindos à lógica de programação");
```

Ao observarmos a linha acima percebemos que o texto está todo entre aspas **“”.** Sempre que representarmos um texto, deverá ser entre aspas.

INSTANCIANDO OBJETOS

A **instância** de um objeto, nada mais do que a criação de um objeto a partir de uma classe, ou seja, **o objeto é a instância de uma classe**.

Até então, nossas classes serviam de moldes para futuros objetos, porém não tínhamos um objeto para ver as características, as ações sofridas ou exercidas. Mas como não podemos ter um objeto sem antes ter definido sua classe, primeiro precisamos aprender como definir uma classe para podermos criar objetos.

> *O objeto é quem armazena os dados e executa as ações, conforme elas estiverem definidas na classe.*

Sintaxe para criar uma nova instância

NomeDaClasse nomeDoObjeto = new NomeDaClasse();

Exemplo

```
Pessoa p1 = new Pessoa();
```

O exemplo acima é uma instância da classe Pessoa, o objeto irá se chamar **p1**.

O nome do objeto é determinado pelo programador, pode ser pessoa1, p1, pe1, primeiroObjeto. Só temos que pensar que o nome do objeto irá influenciar no desenvolvimento de todo o projeto, pois os acessos aos atributos, métodos, são feitos por ele, quanto maior o nome dele maior será o trabalho.

Acessando atributos dos objetos

Quando os atributos são públicos, temos acesso direto a eles. Logo, se quisermos definir os atributos do objeto p1 (nome, idade, salario), utilizamos a seguinte sintaxe:

nomedoobjeto.nomeDoAtributo = valor;

Exemplos

```
p1.nome = "Maria";
p1.idade = 25;
p1.salario=750.60;
```

Invocando métodos através do objeto

Invocar um método é chamá-lo, fazê-lo executar. Como temos métodos com retorno, sem retorno, com argumento e sem argumento, vejamos como invocar cada um deles. Para isso, vamos utilizar no exemplo o diagrama de classe abaixo:

Funcionario
+nome:String +salarioBase:double
+dobrarSalario():void +calcularFerias():double +descontarAdiantamento(valor:double):void +calcularHorasExtras(totalDeHoras:double):double

Invocando métodos sem argumento e sem retorno

O nosso primeiro método, dobrarSalario(), não tem nem argumento, nem retorno. Para invocá-lo, após instanciar o objeto e definir o valor do atributo salário base, basta escrever o nome do objeto, seguido de ponto, seguido do nome do método. Observe que os parênteses são obrigatórios.

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
f1.nome = "Pedro";
f1.dobrarSalario();
```

Invocando métodos sem argumento, mas com retorno

O nosso segundo método, calcularFerias(), não tem argumento, mas tem retorno. Para invocá-lo, precisamos de uma variável para armazenar seu resultado, já que como não é um método **void**, a chamada de sua execução implicará no retorno de uma nova informação, que pode ser exibida na tela.

Primeira forma:

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
double ferias = f1.calcularFerias();
```

Segunda forma:

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
double ferias; //declara a variável
ferias = f1.calcularFerias();
```

Invocando métodos com argumento, mas sem retorno

O nosso terceiro método, descontarAdiantamento(valor:double) não tem retorno, mas precisa saber o valor do adiantamento para poder fazer o desconto no salário. Portanto, antes de invocar o método, precisamos ter esta informação.

Primeira forma:

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
double valor = 100; //salvando o valor na variável
f1.descontarAdiantamento(valor);
```

Segunda forma:

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
f1.descontarAdiantamento(100);//passando o valor diretamente
```

Invocando métodos com argumento e com retorno

O nosso quarto método, calcularHorasExtras(totalDeHoras:double):double tem retorno e também precisa do argumento que corresponde ao total de horas para calcular o valor das horas extras. Portanto, antes de invocar o método, precisamos ter esta informação e precisamos de uma variável para armazenar o resultado do método.

```
Funcionario f1 = new Funcionario();
f1.salarioBase = 600;
double totalDeHoras = 15; //salvando o total de horas na variável
double horasExtras = f1.calcularHorasExtras(totalDeHoras);
```

Importante: observe que a variável criada para receber o resultado do método deve ser igual ao seu tipo de retorno!

EXEMPLOS DE UTILIZAÇÃO DE COMANDOS DE SAÍDA COM OBJETOS

**Imprimindo textos e dados de objetos*

```
System.out.println("O valor do salário é R$ " + f1.salarioBase);
```

Ao observarmos a linha acima vemos que será exibido na tela um texto antes do salário.

**Imprimindo variáveis*

```
System.out.println(horasExtras);
```

Na linha acima observamos que não temos aspas e sim o nome de uma variável. No terminal irá aparecer o valor que a variável "horasExtras" contém. Observe que não temos o "f1." na frente, pois "horasExtras" é uma variável e não pertence ao objeto.

**Imprimindo texto e resultado de método (sem variável)*

```
System.out.println("O valor das férias resultou em: R$ "
+f1.calcularFerias());
```

Na linha acima observamos que temos aspas no texto "O valor das férias resultou em: R$ " e um sinal de mais (+) antes do método, isso ocorre pois iremos mostrar uma mensagem na tela (texto próprio) e executar o método calcularFerias() para que o mesmo retorne o resultado.

**Imprimindo várias informações juntas*

```
System.out.println("O salário é " + f1.salarioBase+ " reais e o total das
férias será " +f1.calcularFerias()+ " reais. ");
```

Na linha acima observamos que temos aspas nos textos e um sinal de + para concatenar respostas e textos.

EXEMPLO COMPLETO

Funcionario
+nome:String +salarioBase:double
+dobrarSalario():void +calcularFerias():double +descontarAdiantamento(valor:double):void +calcularHorasExtras(totalDeHoras:double):double

A classe Funcionario

Vamos observar na figura 3 o código da classe. Analisando os métodos, vemos que para dobrar o salário, multiplicamos o mesmo por 2, para calcular as férias, acrescentamos um terço sobre o valor do salário base. No cálculo do desconto de adiantamento, basta subtrair o valor do salário, e no cálculo das horas extras, assumimos que cada hora extra trabalhada pelo funcionário vale o dobro[1] da sua hora normal. Por exemplo, se o salário do funcionário é 500 reais, e sua jornada mensal de trabalho totaliza 220 horas (padrão), seu valor hora seria 2,72 (resultado da divisão de 500 por 220) e, portanto, cada hora extra vale o dobro, ou seja, 5,44. Se trabalhar 5 horas extras, receberá 27,20 (resultado da multiplicação de 5,44 por 5).

```
1  public class Funcionario{
2      public String nome;
3      public double salarioBase;
4
5      public void dobrarSalario(){
6          this.salarioBase = this.salarioBase * 2;
7      }
8      public double calcularFerias(){
9          //salário acrescido de um terço do salário
10         return this.salarioBase * 1.33;
11     }
12     public void descontarAdiantamento(double valor){
13         this.salarioBase = this.salarioBase - valor;
14     }
15     public double calcularHorasExtras(double totalDeHoras){
16         //horas extras valem o dobro do valor hora normal
17         return this.salarioBase/220 * 2 * totalDeHoras;
18     }
19 }
```

Figura 3

Observando a classe da figura 3, vamos analisar:

1. **Quais métodos alterariam o estado do objeto?**
2. **Quais métodos geram informações que não temos nos atributos da classe?**

Respostas:

1. dobrarSalario e descontarAdiantamento, pois modificam o salário.
2. calcularFerias e calcularHorasExtras, pois retornam um dado que precisa ser calculado

[1] Aqui desconsideramos as horas extras normais que valem 50% a mais do valor hora.

A classe Main

Agora que já temos a classe pronta, podemos criar a classe Main. Nela, criamos (instanciamos) os objetos, determinamos que operações serão realizadas.

> *Não somos obrigados a utilizar todos os métodos da classe. Mas, se não forem invocados, eles não se executam sozinhos.*

```
public class Main{
    public static void main(String args[]){
        //instanciando o objeto
        Funcionario f1 = new Funcionario();

        //definindo seus atributos
        f1.nome = "Pedro";
        f1.salarioBase = 800;

        //invocando seus métodos sem retorno e sem argumento
        f1.dobrarSalario();
        //invocando seus métodos sem retorno e com argumento
        f1.descontarAdiantamento(300);
        //invocando métodos sem argumento mas com retorno:
        double ferias = f1.calcularFerias();
        //invocando métodos com argumento e com retorno
        double horasExtras = f1.calcularHorasExtras(10);

        //Exibindo informações para o usuário
        System.out.println("Olá, " + f1.nome + ", seu salário é: " + f1.salarioBase);
        System.out.println("O valor de suas férias será: " + ferias);
        System.out.println("O valor de suas horas extras é: " + horasExtras);
    }
}
```

Figura 4

> *Se escrevermos um atributo ou método diferente de como está escrito na classe, o compilador do Java não vai reconhecer o comando, acusando erro de sintaxe.*

O resultado da sequência de passos contidos na Main pode ser visto na figura 5, que mostra o terminal após a execução da classe Main.

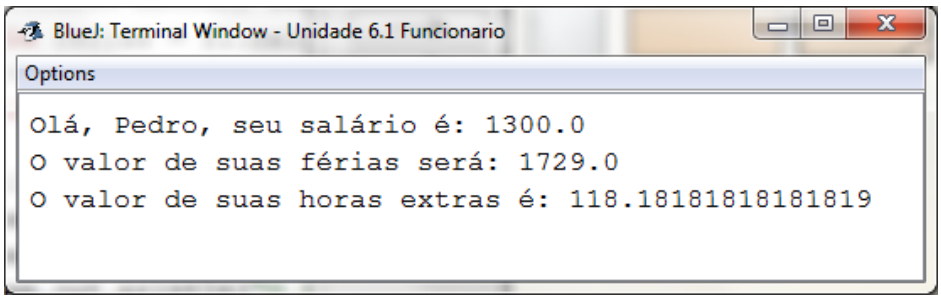


Figura 5