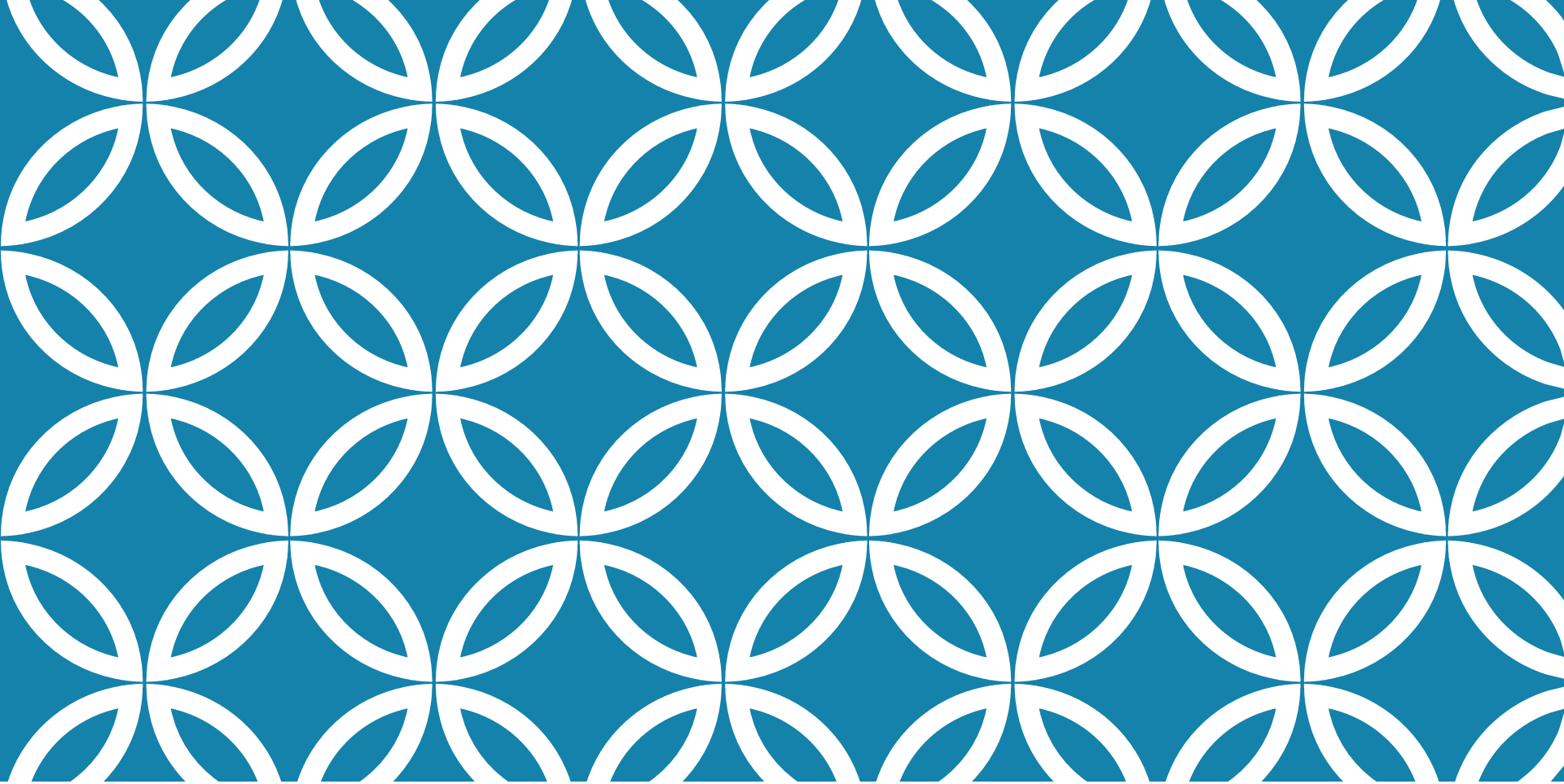


Programação Orientada a Objetos

Rafael Vieira Coelho

O que é um Programa?

Automatização de processos realizados manualmente por pessoas

- Com a automatização da apuração de votos, o tempo para obter os resultados e a chance de ocorrer uma falha humana diminuíram drasticamente
- Os computadores são capazes de executar instruções matemáticas mais rapidamente do que o homem
- Resolvem problemas complexos de maneira mais eficiente

O que é um Programa?

- Mas eles não possuem a inteligência necessária para definir quais
- instruções devem ser executadas para resolver uma determinada tarefa
- Uma pessoa precisa definir um **roteiro** com a sequência de comandos necessários para realizar uma determinada tarefa e depois passar para um computador executar esse roteiro
- Formalmente, esses roteiros são chamados de **programas**

Linguagem de Máquina

- Os computadores só sabem ler instruções escritas em **linguagem de máquina**

- Uma instrução escrita em linguagem de máquina é uma sequência formada por “0s” e “1s” que representa a ação que um computador deve executar

- Um arquivo contendo as instruções de um programa em Linguagem de Máquina é chamado de executável

```
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000100000000010000000000
00000000000011100001111110111010000011100000000010110100000010 0
11100110100100001101110000000000101001100110011010010000101010 1
00011010000110100101110011001000000111000001110010011011110110 0
11101110010011000010110110100100000011000110110000101101110011 0
11100110111101110100001000000110001001100101001000000111001001 1
```

Linguagem de Programação

- Escrever um programa em linguagem de máquina é totalmente inviável para uma pessoa

- As *linguagens de programação* tentam se aproximar das linguagens humanas

- Um arquivo contendo as instruções de um programa em linguagem de programação é chamado de **arquivo fonte**

```
public class OlaMundo {

  public static void main(String[] args) {

    System.out.println("Olá Mundo!");

  }
}
```

Ling. de Máquina x Ling. de Programação

```
global  _start

        section .text
_start:
        mov     rax, 1
        mov     rdi, 1
        mov     rsi, message
        mov     rdx, 13
        syscall
        ; exit(0)
        mov     eax, 60
        xor     rdi, rdi
        syscall
message:
        db      "Hello, World!", 10
```

```
print("Hello, World!")
```

Compilador

- É necessário traduzir o código escrito em linguagem de programação por uma pessoa para um código em linguagem de máquina para que um computador possa processar

- Essa tradução é realizada por programas especiais chamados **compiladores**

Máquinas Virtuais

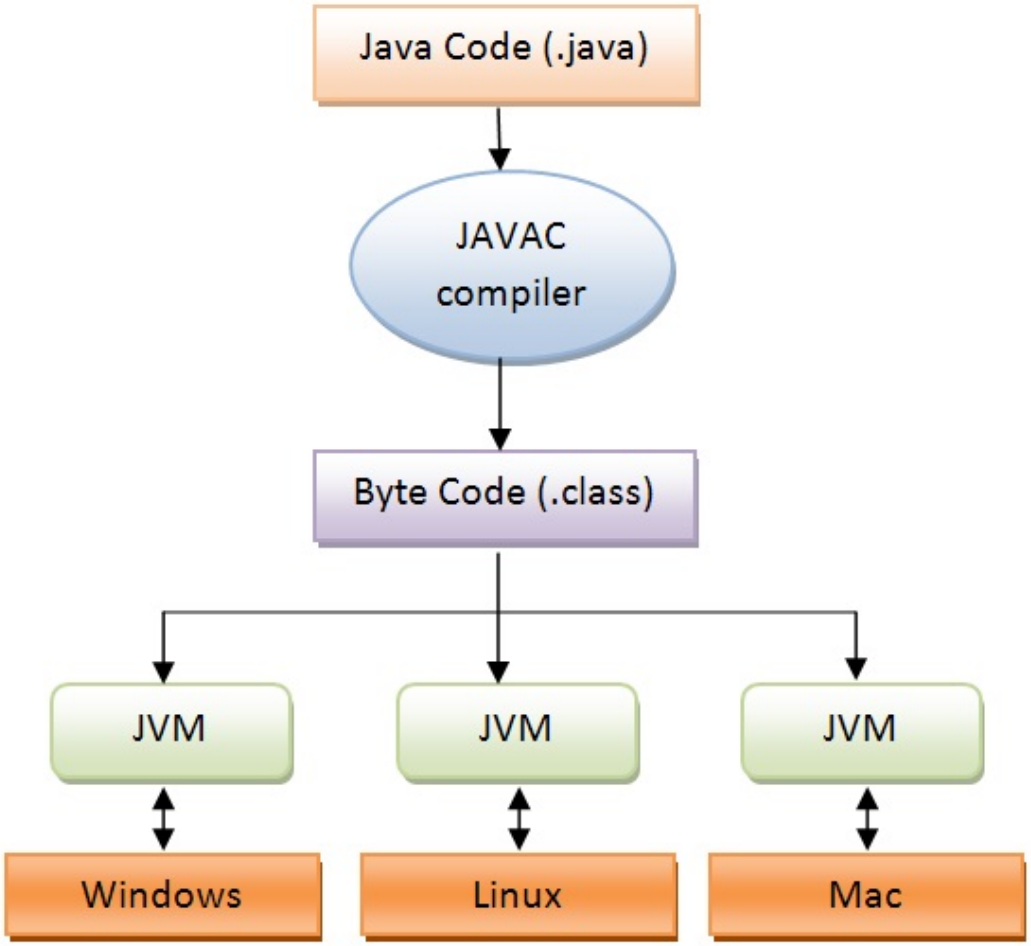
- A linguagem de máquina de um computador é definida pela **arquitetura do processador** desse computador
- Há diversas arquiteturas diferentes (Intel, ARM, PowerPC, etc.) e cada uma delas define uma linguagem de máquina diferente
- Um programa também pode não executar em computadores com **sistemas operacionais** diferentes

Máquinas Virtuais

- Para determinar se um código em linguagem de máquina pode ou não ser executada por um computador, devemos considerar a **arquitetura do processador** e o **sistema operacional** desse computador

- Para tentar resolver o problema do desenvolvimento de aplicações multiplataforma, surgiu o conceito de *máquina virtual*

JVM - Java Virtual Machine



Máquinas Virtuais

- Uma desvantagem em utilizar uma máquina virtual para executar um programa é a diminuição de performance, já que a própria máquina virtual consome recursos do computador
- Além disso, as instruções do programa são processadas primeiro pela máquina virtual e depois pelo computador

1) Exemplo de Programa Java

- O código fonte Java deve ser colocado em arquivos com a extensão **.java**

- Toda aplicação Java precisa ter um método especial chamado **main** para executar

- O código gerado pelo compilador Java é armazenado em arquivos com a extensão **.class**

```
class OlaMundo {
  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  }
}
```

Método Main: Ponto de Entrada

- Para um programa Java executar, é necessário definir um método especial para ser o ponto de entrada do programa, ou seja, para ser o primeiro método a ser chamado quando o programa for executado

- O método main precisa ser **public**, **static**, **void** e receber um **array** de **String** como argumento

- Variações da assinatura do método **main**

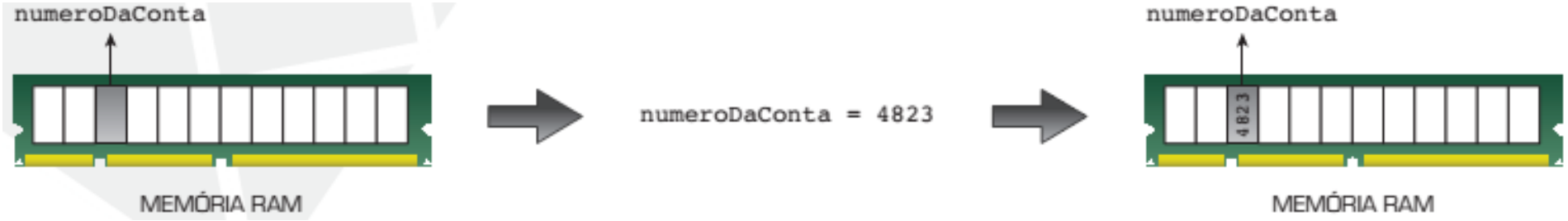
```
static public void main(String[] args)
public static void main(String[] args)
public static void main(String args[])
public static void main(String[] parametros)
```

2) Variáveis

- Basicamente, o que um programa faz é manipular dados
- Em geral, esses dados são armazenados em **variáveis** localizadas na memória RAM do computador
- Uma variável pode guardar dados de vários tipos
  - números;
  - textos;
  - booleanos (verdadeiro ou falso);
  - referências de objetos.

Variáveis

Além disso, toda variável possui um nome que é utilizado quando a informação dentro da variável precisa ser manipulada pelo programa



Declaração

- Na linguagem de programação Java, as variáveis devem ser declaradas para que possam ser utilizadas

- A declaração de uma variável envolve definir um nome único (identificador) dentro de um escopo e um tipo de valor

- As variáveis são acessadas pelos nomes e armazenam valores compatíveis com o seu tipo

```
// Uma variável do tipo int chamada numeroDaConta.
int numeroDaConta;

// Uma variável do tipo double chamada precoDoProduto.
double precoDoProduto;
```

Declaração

Na convenção de nomes da linguagem Java, os nomes das variáveis devem seguir o padrão **camel case** com a primeira letra minúscula (**lower camel case**)

- nomeDoCliente
- numeroDeAprovados

Declaração

- A declaração de uma variável pode ser realizada em qualquer linha de um bloco

- Não é necessário declarar todas as variáveis no começo do bloco como acontece em algumas linguagens de programação

```
// Declaração com Inicialização
int numero = 10;

// Uso da variável
System.out.println(numero);

// Outra Declaração com Inicialização
double preco = 137.6;

// Uso da variável
System.out.println(preco);
```

Declaração

Não podemos declarar duas variáveis com o mesmo nome em um único bloco ou escopo pois ocorrerá um erro de compilação.

```
// Declaração com Inicialização
int numero = 10;

//Erro de Compilação
int numero = 15;
```

Inicialização

- Toda variável deve ser inicializada antes de ser utilizada pela primeira vez

- Se isso não for realizado, ocorrerá um erro de compilação ou um aviso (warning) de possível erro

```
// Declarações
int numero;
double preco;

// Inicialização
numero = 10;

// Uso Correto
System.out.println(numero);

// Erro de compilação
System.out.println(preco);
```

3) Tipos Primitivos

A linguagem Java define um conjunto de tipos básicos de dados que são chamados **tipos primitivos**

Tipo	Descrição	Tamanho
byte	Valor inteiro entre -128 e 127 (inclusivo)	1 byte
short	Valor inteiro entre -32.768 e 32.767 (inclusivo)	2 bytes
int	Valor inteiro entre -2.147.483.648 e 2.147.483.647 (inclusivo)	4 bytes
long	Valor inteiro entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807 (inclusivo)	8 bytes

Tipo	Descrição	Tamanho
float	Valor com ponto flutuante entre 1,4012984643248170707 × 10−45 e 3,4028234663852886 × 1038 (positivo ou negativo)	4 bytes
double	Valor com ponto flutuante entre 4,94065645841246544 × 10^{-324} e 1,79769313486231570 × 10^{308} (positivo ou negativo)	8 bytes
boolean	true ou false	1 bit
char	Um único caractere Unicode de 16 bits. Valor inteiro e positivo entre 0 (ou '\u0000') e 65.535 (ou '\uffff')	2 bytes

4) Operadores

Para manipular os valores das variáveis de um programa, devemos utilizar os operadores oferecidos pela linguagem de programação adotada

A linguagem Java possui diversos operadores e os principais são categorizados da seguinte forma

Aritmético (+, -, *, /, %)

Atribuição (=, +=, -=, *=, /=, %=)

Relacional (==, !=, <, <=, >, >=)

Lógico (&&, ||)

Operadores - Aritmético

Funcionam de forma muito semelhante aos operadores na matemática

Os operadores aritméticos são

- Soma +
- Subtração –
- Multiplicação *
- Divisão /
- Módulo %

```
int umMaisUm = 1 + 1; // umMaisUm = 2
int tresVezesDois = 3 * 2; // tresVezesDois = 6
int quatroDivididoPor2 = 4 / 2; //
quatroDivididoPor2 = 2
int seisModuloCinco = 6 % 5; // seisModuloCinco
= 1
int x = 7;
x = x + 1 * 2; // x = 9
x = x - 3; // x = 6
x = x / (6 - 2 + (3 * 5) / (16 - 1)); // x = 2
```

Operadores - Atribuição

Sabemos que usamos como um dos operadores de atribuição, o operador = (igual)

Os operadores de atribuição são:

Simples =
Incremental +=
Decremental -=
Multiplicativa *=
Divisória /=
Modular %=

```
int valor = 1; // valor = 1
valor = valor + 2; // valor = 3
valor = valor - 1; // valor = 2
valor = valor * 6; // valor = 12
valor = valor / 3; // valor = 4
valor = valor % 3; // valor = 1
```

```
int valor = 1; //
valor = 1
valor += 2; // valor =
3
valor -= 1; // valor =
2
valor *= 6; // valor =
12
valor /= 3; // valor =
4
valor %= 3; // valor =
1
```

Operadores - Relacional

- Muitas vezes precisamos determinar a relação entre uma variável ou valor e outra variável ou valor
- Nessas situações, utilizamos os operadores relacionais
- As operações realizadas com os operadores relacionais devolvem valores do tipo primitivo **boolean**

- Os operadores relacionais são
  - Igualdade ==
  - Diferença !=
  - Menor <
  - Menor ou igual <=
  - Maior >
  - Maior ou igual >=

```
int valor = 2;
boolean t = false;
t = (valor == 2); // t =
true
t = (valor != 2); // t =
false
t = (valor < 2); // t =
false
t = (valor <= 2); // t =
true
t = (valor > 1); // t =
true
t = (valor >= 1); // t =
true
```

Operadores - Lógico

- A linguagem Java permite verificar duas ou mais condições através de operadores lógicos

- Os operadores lógicos são
  - "E" lógico &&
  - "OU" lógico ||

```
int valor = 30;
boolean teste = false;
teste = valor < 40 && valor > 20; // teste
= true
teste = valor < 40 && valor > 30; // teste
= false
teste = valor > 30 || valor > 20; // teste
= true
teste = valor > 30 || valor < 20; // teste
= false
teste = valor < 50 && valor == 30; // teste
= true
```

5) Comandos Condicionais

- O comportamento de uma aplicação pode ser influenciado por valores definidos pelos usuários

- Para verificar uma determinada condição e decidir qual bloco de instruções deve ser executado, devemos aplicar o comando **if**

```
if (preco < 0) {
        System.out.println("O preço do produto não pode ser negativo");
} else {
        System.out.println("Produto cadastrado com sucesso");
}
```

If-else

```java
//Exemplo de SE-ENTÃO
if (x >= 10) {
    System.out.println("O valor " + x + " é maior ou igual a dez.");
} else {
    System.out.println("Valor menor que dez.");
}
```

Switch-case

```
//Exemplo de ESCOLHA-CASO
switch (x) {
    case 10: {
        System.out.println("valor igual a dez");
        break;
    }
    case 11: System.out.println("valor igual a onze"); break;
    case 13: System.out.println("valor igual a treze"); break;
    default: System.out.println("outro valor");
}
```

6) Comandos de Repetição

Em alguns casos, é necessário repetir um trecho de código diversas vezes

Suponha que seja necessário imprimir 10 vezes na tela a mensagem: “Bom Dia”

Isso poderia ser realizado colocando 10 linhas iguais a essa no código fonte

```
System.out.println("Bom Dia");
```

Se ao invés de 10 vezes fosse necessário imprimir 100 vezes, já seriam 100 linhas iguais no código fonte

É muito trabalhoso utilizar essa abordagem para solucionar esse problema

While

Através do comando **while**, é possível definir quantas vezes um determinado trecho de código deve ser executado pelo computador

```
int contador = 0;

while (contador < 100) {
        System.out.println("Bom Dia");
        contador++;
}
```

O parâmetro do comando **while** tem que ser um valor booleano

- Caso contrário, ocorrerá um erro de compilação

Do-While

```
int contador = 0;

do {
        System.out.println("Bom Dia");
        contador++;
} while (contador < 100);
```

For

O comando **for** é análogo ao **while**

A diferença entre esses dois comandos é que o for recebe três argumentos

```
for(int contador = 0; contador < 100; contador++) {
        System.out.println("Bom Dia");
}
```

```java
public class LaçosRepetição {
    public static void main(String[] args) {
        //Exemplo de For
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " x 10 = " + (i*10));
        }

        //Exemplo de While
        int i = 0;

        while (i < 10) {
            System.out.println(i + " x 10 = " + (i*10));
             i++;
        }

        //Exemplo de Do-While
        i = 0;

        do {
            System.out.println(i + " x 10 = " + (i*10));
            i++;
        } while (i < 10);
    }
}
```

7) Escrita de Dados

```
int x = 10;
System.out.println("O valor é " + x);

int y = 20;
System.out.printf("\n sim, igual a %d", y);
```

8) Leitura de Dados

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String texto = br.readLine();
int inteiro = Integer.parseInt(br.readLine());
long inteiro_duplo = Long.parseLong(br.readLine());
float real = Float.parseFloat(br.readLine());
double real_duplo = Double.parseDouble(br.readLine());
char caractere = br.readLine().charAt(0);
```

Exceção

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class LeituraEscrita {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Informe um texto: ");
        String texto = br.readLine();
        System.out.println("Informe um número inteiro: ");
        int inteiro = Integer.parseInt(br.readLine());
        System.out.println("Informe um número inteiro de dupla precisão: ");
        long inteiro_duplo = Long.parseLong(br.readLine());
        System.out.println("Informe um número em ponto flutuante: ");
        float real = Float.parseFloat(br.readLine());
        System.out.println("Informe um número em ponto flutuante de dupla precisão: ");
        double real_duplo = Double.parseDouble(br.readLine());
        System.out.println("Informe um caractere:");
        char caractere = br.readLine().charAt(0);

        System.out.println(" Int: " + valorInteiro
                           + " Long: " + valorLong
                           + " Float: " + valorReal
                           + " Double: " + valorDouble
                           + " Frase: " + frase
                           + " Palavra: " + palavra
                           + " Caractere: " + caractere);
    }
}
```

O que chama a atenção neste código?

EXERCÍCIOS

1. Acessar o link:https://www.urionlinejudge.com.br

2. Realizar em duplas os problemas (em Java):

https://www.urionlinejudge.com.br/judge/pt/problems/view/1001

https://www.urionlinejudge.com.br/judge/pt/problems/view/1002

https://www.urionlinejudge.com.br/judge/pt/problems/view/1003

https://www.urionlinejudge.com.br/judge/pt/problems/view/1004

https://www.urionlinejudge.com.br/judge/pt/problems/view/2313

Para submeter uma solução:

https://www.urionlinejudge.com.br/judge/pt/runs/add