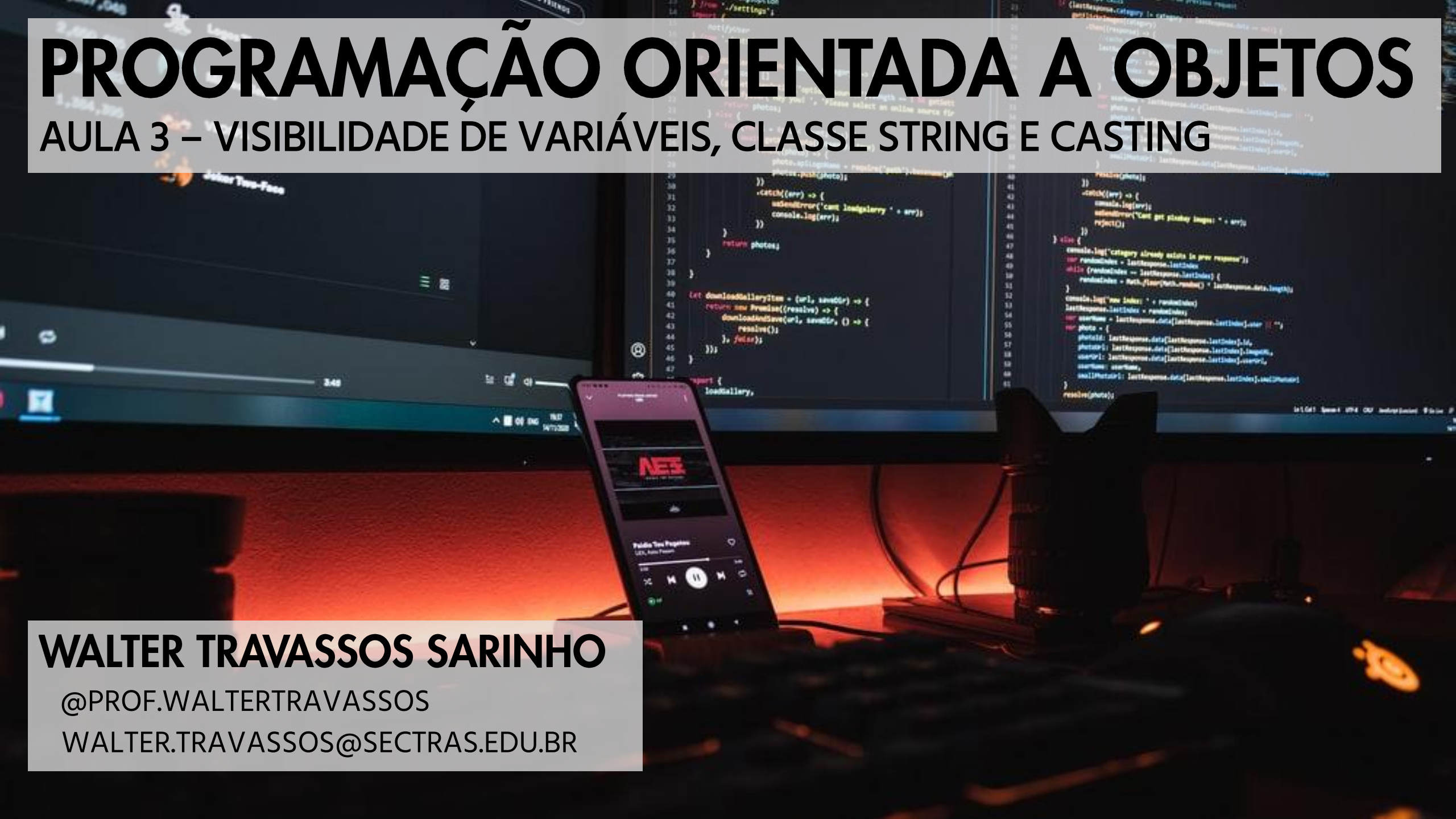
PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 3 – VISIBILIDADE DE VARIÁVEIS, CLASSE STRING E CASTING



WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR

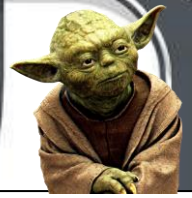
Dúvidas sobre a aula passada

Mapa de Estudos

AULA 01
Paradigmas de Programação e história do Java.
IDEs e Hello Universe

AULA 03
Visibilidade de variáveis, classe String e casting.

AULA 02
Manipulação de datas.
Sintaxe da Linguagem Java e variáveis primitivas e de referência.



AULA 04
Classes Wrapper.
Operadores e métodos de conversão.

AULA 05
Formatar a saída de dados numérica.
Operador condicional if.
Métodos para comparar Strings

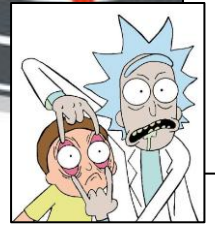
Avaliação 1

Java

Mapa de Estudos

AULA 06
Manipulação de Strings. Operador condicional switch.

AULA 07
Operadores de atribuição compostos. Estruturas de repetição: while, do while e for.



AULA 08
Vetores (for each) e matrizes. Classe ArrayList

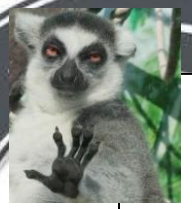
AULA 09
O paradigma Orientado a Objetos

AULA 10
Construindo métodos e Métodos construtores

Avaliação 2

Java

Mapa de Estudos



AULA 12
Classes Abstratas e Interfaces

AULA 11
Pilares da POO: Abstração, Encapsulamento, Herança e Polimorfismo

Avaliação 3

Java

Exercício 1

Faça um programa que pergunte o peso e a altura do usuário em seguida apresente os valores perguntados. As variáveis **peso** e **altura** devem ser do tipo **double**.

```
peso:
93,5
altura:
1,82
Peso: 93.5 | Altura: 1.82
```

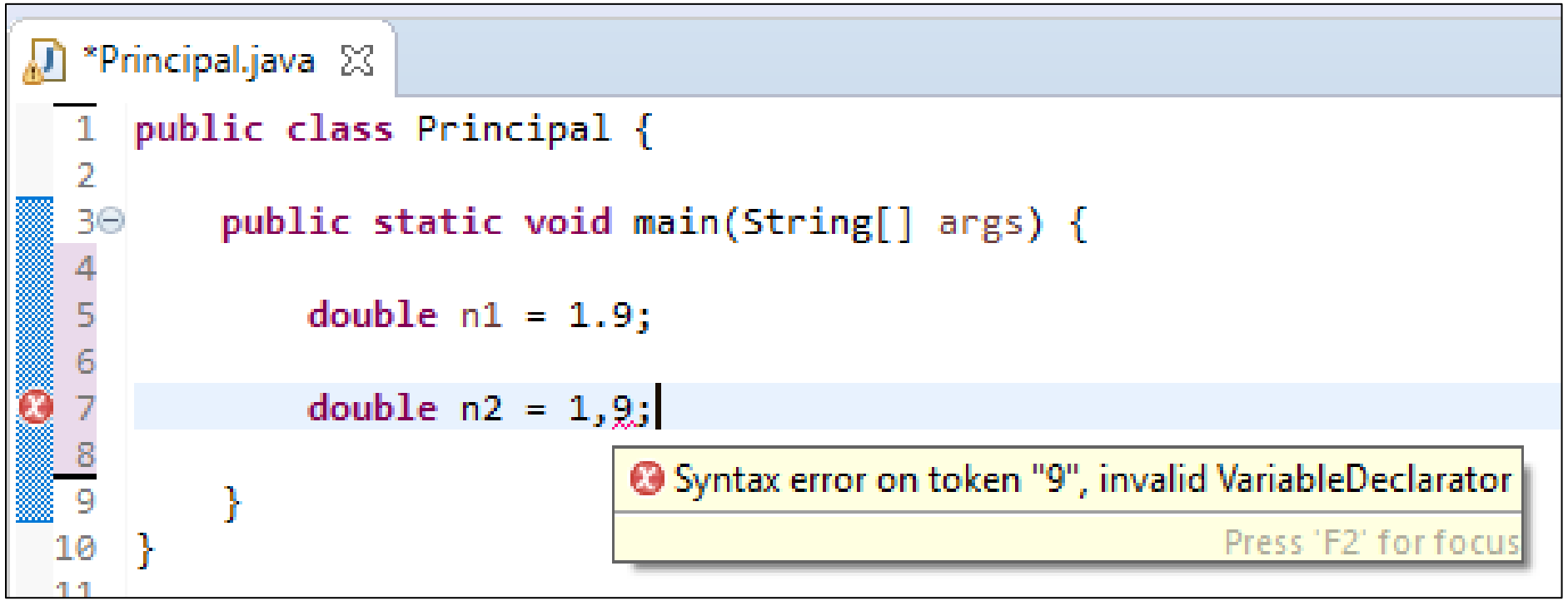
Atenção

- Ao inserir no console um valor em **ponto flutuante**, o separador padrão (americano) de valores numéricos **é o ponto**. No entanto, em versões mais novas do eclipse, quando inserimos numa variável valores separados com ponto, é apresentado um erro de **InputMismatchException**.
- Este erro está relacionado com a entrada de dados.

```
<terminated> Principal (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe
digite um numero:
1.2
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextDouble(Scanner.java:2564)
        at Principal.main(Principal.java:8)
```

No entanto...

Dentro do código, podemos colocar valores do tipo *float* e *double* apenas separados por ponto.



Exercício 1 - Resposta

Para resolver, **insira os dados em ponto flutuante separados por vírgula**. Mas observe que a saída do dado é apresentada com um ponto como separador.

```java
import java.util.Scanner;

public class Pokemon {

    public static void main(String[] args) {
        double peso, altura;
        Scanner teclado = new Scanner(System.in);

        System.out.println("peso: ");
        peso = teclado.nextDouble();
        System.out.println("altura: ");
        altura = teclado.nextDouble();
        System.out.println("Peso: "+peso+" | Altura: "+altura);
    }

}
```

```
peso:
93,5
altura:
1,82
Peso: 93.5 | Altura: 1.82
```

Locale.setDefault(Locale.US);

- Outra possibilidade é configurar a entrada de dados no console para o padrão americano com o método *Locale.setDefault(Locale.US);*
- Assim, os valores de entrada serão todos separados com o ponto.

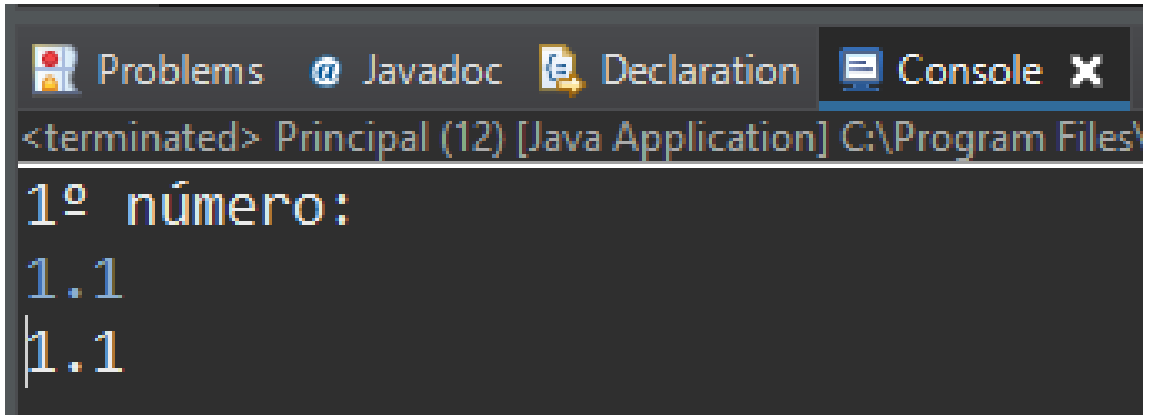
Principal.java

```java
import java.util.Locale;
import java.util.Scanner;

public class Principal {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner teclado = new Scanner(System.in);
        System.out.println("1º número: ");
        double num1 = teclado.nextDouble();
        System.out.println(num1);
        teclado.close();

    }
}
```



Recomendações – Boas Práticas

É recomendado que ao capturar dados do usuário, sempre peguemos esse valor no formato **String** para, em seguida, converter para o tipo desejado.

Sistema "apaga" o zero do CPF no cadastro e dificulta pedido de R$ 600

Thâmara Kaoru
Do UOL, em São Paulo
27/04/2020 15h41

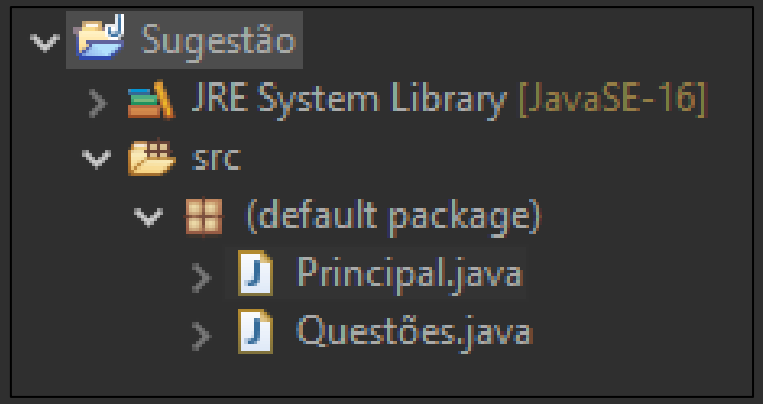
Trabalhadores que possuem algum familiar com número de CPF que começa com zero estão enfrentando dificuldades para receber o auxílio emergencial de R$ 600.

https://economia.uol.com.br/noticias/redacao/2020/04/27/auxilio-r-600-zero-cpf-some-confirmacao-do-pedido.htm

Sugestão para organização dos exercícios no projeto

Sugestão

- Crie um projeto java com 2 classes.
- A primeira classe será a **Principal** contendo o método **main**.
- A segunda classe pode se chamar **Questões**. Vamos criar um método para cada exercício que formos fazendo durante a aula. Cada um dos exercícios vai ficar dentro de um método separado.
- Faremos a chamada para cada um dos métodos dos exercícios na classe **Principal**, dentro do método **main**.
- Por exemplo, se na classe **Questões** criarmos o método **exercício1()**, dentro do método **main**, chamaremos assim: **Questões.exercício1();**



Principal.java

```java

public class Principal {

    public static void main(String[] args) {
        Questões.exercício1();
    }

}
```

Questões.java

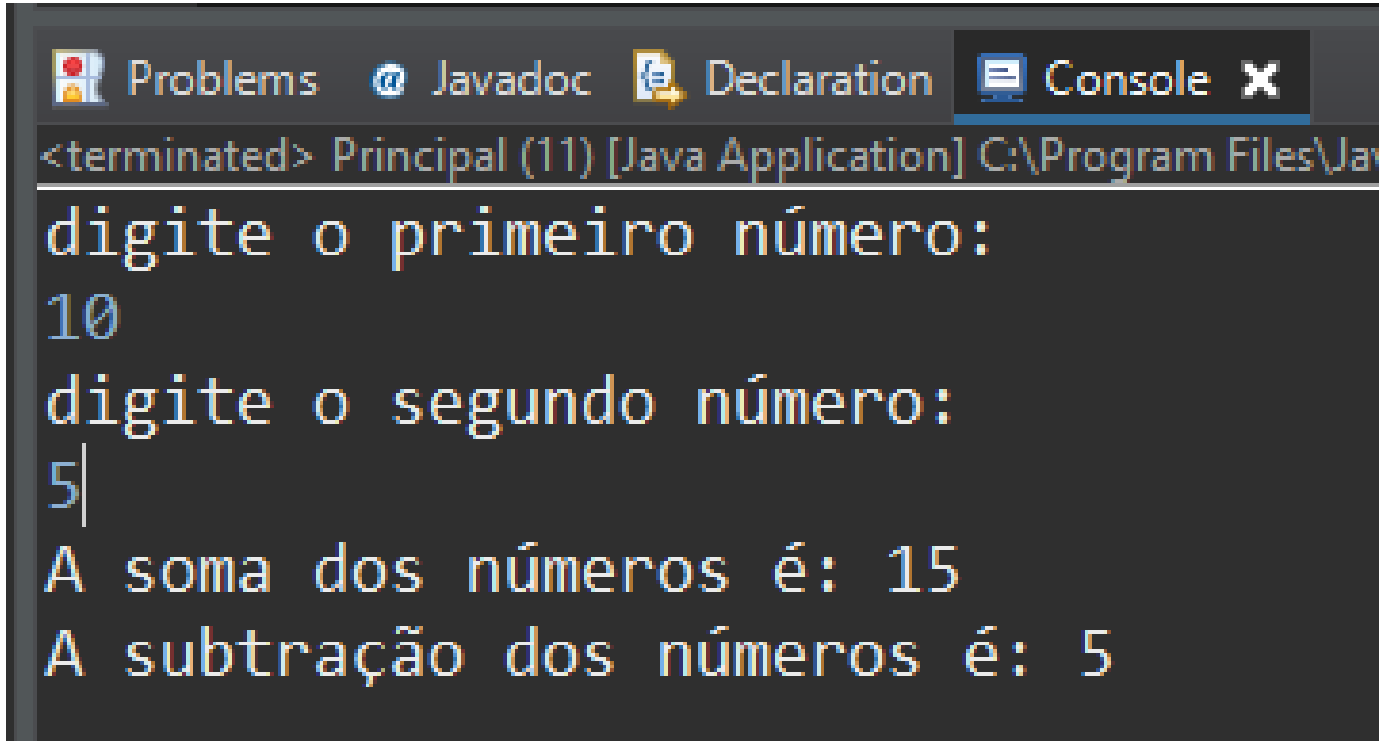
```java

public class Questões {
    public static void exercício1() {
        System.out.println();
    }
}

```

Exercício 2 - Revisão

Faça um programa que receba dois números (por exemplo 10 e 5) e em seguida apresente a soma e a diferença desses números.



Exercício 2 - Resposta

*Principal.java

```java
import java.util.Scanner;
public class Principal {
    public static void main(String[] args) {

        Scanner teclado = new Scanner(System.in);
        System.out.println("digite o primeiro número: ");
        int n1 = teclado.nextInt();
        System.out.println("digite o segundo número: ");
        int n2 = teclado.nextInt();
        int soma = n1+n2;
        System.out.println("A soma dos números é: " + soma );
        System.out.println("A subtração dos números é: "+ (n1-n2));
        teclado.close();
    }
}
```

Problems Javadoc Declaration Console

<terminated> Principal (11) [Java Application] C:\Program Files\Ja

```
digite o primeiro número:
10
digite o segundo número:
5
A soma dos números é: 15
A subtração dos números é: 5
```

Visibilidade de Variáveis

Visibilidade de Variáveis

- Uma variável é válida somente dentro do bloco que é declarada.
- O bloco é delimitado pelas chaves **{ }.**
- Note que o escopo de uma classe é delimitado pelas chaves. O bloco de atuação de um método também!

Onde começam os blocos de código?

```
public class Miau {
        public static void main (String[] args) {
                int a=2;
                if (a>0) {
                        int b=3;
                        System.out.println(a + b);
                }
                System.out.println( a );
                System.out.println( b );
                int a=4;
        }
}
```

Onde começam os blocos de código?

```
public class Miau {
        public static void main (String[] args) {
                int a=2;
                if (a>0) {
                        int b=3;
                        System.out.println(a + b);
                }
                System.out.println( a );
                System.out.println( b );
                int a=4;
        }
}
```

Bloco 1 - Início

Bloco 2 - Início

Bloco 3 - Início

Bloco 3 - fim

Bloco 2 - fim

Bloco 1 - fim

Esse programa compila? Sim ou não?

```
public class Miau {
        public static void main (String[] args) {
                int a=2;
                if (a>0) {
                        int b=3;
                        System.out.println(a + b);
                }
                System.out.println( a );
                System.out.println( b );
                int a=4;
        }
}
```

Esse programa compila? Sim ou não?

```
public class Miau {
    public static void main (String[] args) {
        int a=2;
        if (a>0) {
            int b=3;
            System.out.println(a + b); //5
        }
        System.out.println( a ); //2
        System.out.println( b );//erro! Inválido
        int a=4; //erro! Declaração duplicada
    }
}
```

O programa não compila!

Inicialização de Variáveis

Criação e Inicialização de Variáveis

- Uma coisa é a **criação** de uma variável: **int x;**
- Outra coisa é sua **inicialização** (ou atribuição): **x = 2;**
- Outra coisa é **criar inicializando** a variável: **int x = 2;**

Declaração de Variáveis e Atribuição

<tipo> <identificador> = <valor> ;

```
int a = 2;
String nome = “joao”;
double x = 0.59;
boolean resultado = true;
```

As casas decimais dentro do código Java são separadas por um **ponto**, ou seja, **não utilize vírgula** para representação de números decimais dentro do programa. Use vírgula apenas quando for capturar o valor do usuário no console.

```
int a;
String nome;
double x;
boolean resultado;

a = 2;
nome = “João”;
x = 0.59;
resultado = true;
```

Esse programa compila?

Não! As variáveis locais devem ser inicializadas antes de utilizá-las.

```
public class Miau {
        public static void main (String[] args) {
                int a;
                String nome;
                if ( a > 0 ) {
                        a = 5;
                        nome = “Maria”;
                }
                System.out.println( a );
                System.out.println( nome );
} }
```

Solução
int a = 0; **//zero**
String nome = null;

Erro de compilação nas duas linhas.
Por quê?

A Classe String

Vocês já sabem que...

Um objeto da classe String é equivalente a uma variável do tipo literal em algoritmos (uma cadeia de caracteres).

Podemos concatenar Strings com o **+**.

*Principal.java

```
1  public class Principal {
2
3      public static void main(String[] args) {
4          String nome = "Kouga";
5          String constelação = "Pegasus";
6          System.out.println("Eu sou: " + nome);
7          System.out.println("Cavaleiro da constelação de: " + constelação);
8          System.out.println("Pegasus... Ryu Sei Ken");
9      }
10 }
```

Problems @ Javadoc Declaration Console

```
<terminated> Principal (1) [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (12 de fev de 2
Eu sou: Kouga
Cavaleiro da constelação de: Pegasus
Pegasus... Ryu Sei Ken
```

Mas não sabem que podemos utilizar caracteres especiais para o tipo String!

Caracteres especiais - que podem ser inseridos entre as aspas de um valor do tipo String

\n ou **\r -** nova linha

\' (apóstrofo) ou **\"** (aspas)

\t - Tabulação horizontal (tab)

\b - Backspace

\f - Avanço de página (pouco utilizado)

\\ - Barra invertida (para representar uma barra invertida dentro da String)

\uXXXX - Representação de caracteres Unicode (útil para inserir símbolos e caracteres especiais) - https://symbl.cc/pt/unicode-table/

Vamos testar alguns?

```java
public class CaracteresEspeciais {
    public static void main(String[] args) {
        System.out.println("Primeira linha\nSegunda linha");
        System.out.println("Tabulação:\tColuna 1\tColuna 2");
        System.out.println("Barra invertida: \\");
        System.out.println("Emoji Unicode: \u2764"); // Coração ❤
    }
}
```

Intervalo: 15 minutos

Coerção implícita e explícita

Casting (cast), Parser ou type casting

Valores Numéricos Aceitáveis pelo Compilador

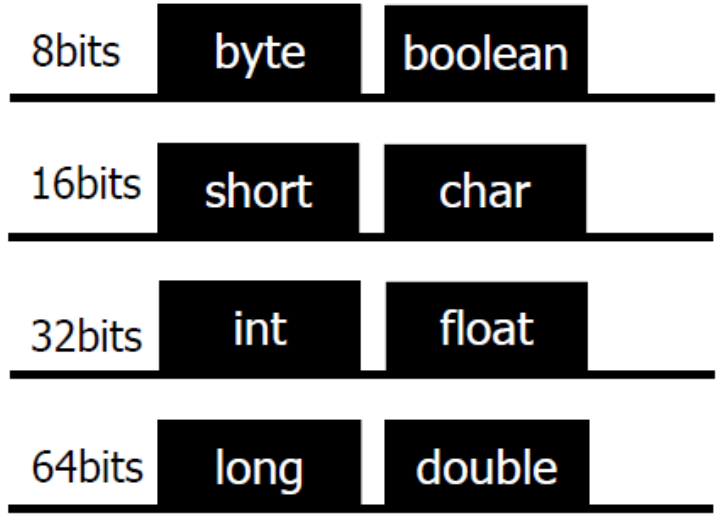
Tipos default (o compilador conhece):

- 2 -> tipo **int**
- 2.0 -> tipo **double**
- 2L ou l -> tipo **long**
- 2.0F ou f -> tipo **float**

Valores que cabem nas variáveis

- Tipos menores podem ser atribuídos diretamente a tipos maiores (coerção implícita).
- No entanto, tipos maiores, não cabem totalmente em tipos menores. Assim, onde vai acontecer erro?

```
1. long n = 2;          // (ok) long <- int
2. double x = 2.0F;     // (ok) double <- float
3. int i = 2L;          // (erro) int <- long
4. float p = 2.0;       // (erro) float <- double
```



Coerção Implícita (promoção)

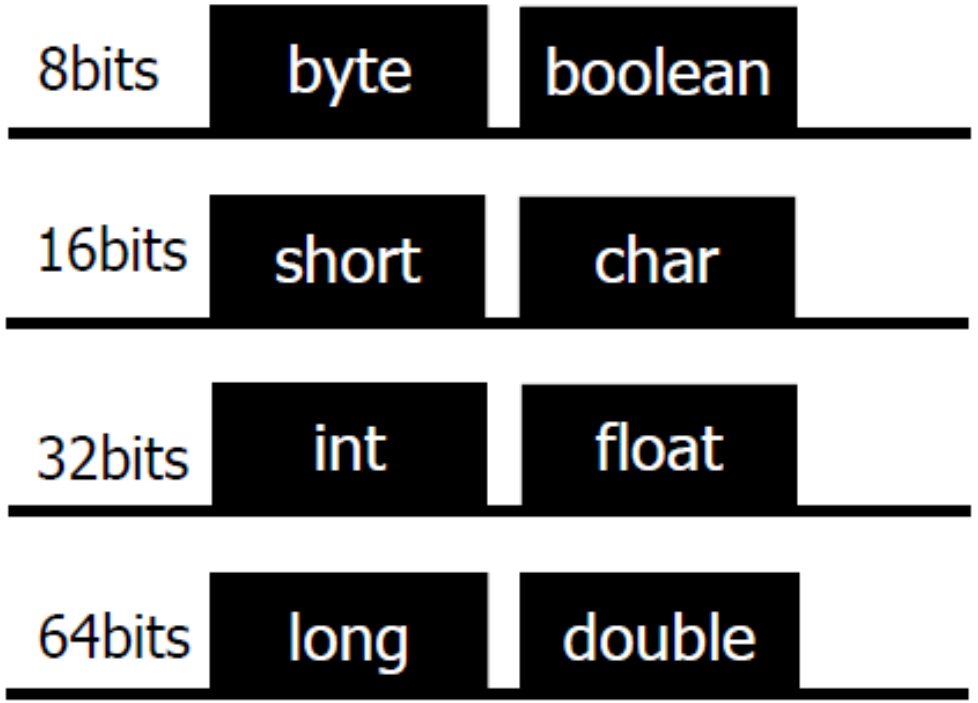
Os tipos menores são convertidos para os tipos maiores, automaticamente.

Exemplos de promoção:

int char

```
long n = 1 + 'A';          //1+65 = 66L
double d = n + 2.0;        //68.0
```

long double




char msg[]={78, 111, 119, 32, 72, 105, 114, 105, 110, 103, 0};
EA
CANADA
EACanada.com

Coerção Explícita (casting)

- Conversão de tipos maiores para tipos menores pode acarretar perda de informação (é de sua responsabilidade gerenciar essa perda de dados!).
- Usa-se o operador de **coerção de tipo.**
- Sintaxe: **(casting) expressão**

Exemplos:

1. int i = (int) 2L; //2 int <- long não perde
2. int i = (int) 2.75F; //2 int <- float perde 0.75
3. int i = (int) 2.75; //2 int <- double perde 0.75
4. float x = (float) 6.5; //6.5f float <- double não perde

Exercício

As conversões abaixo acarretarão em perda de informação?

Sim ou não?

1. int j = (int)(long) 1.5;
2. double d = (int) 1.5;
3. double d2 = (float) 1.5;

Exercício

As conversões abaixo acarretarão em perda de informação?

Sim ou não?

1. int j = (int)(long) 1.5; **SIM**
2. double d = (int) 1.5; **SIM**
3. double d2 = (float) 1.5; **NÃO**

Problema entre *float* e *double*

Utilizando o Eclipse, alguns alunos tentaram criar uma variável do tipo **float** mas não conseguiram inserir um valor.

- Por exemplo: float f = 32.5; (tentem executar essa instrução).
- **Java considera todo número em ponto flutuante como sendo do tipo double**.
- Então temos duas formas de resolver esse problema:

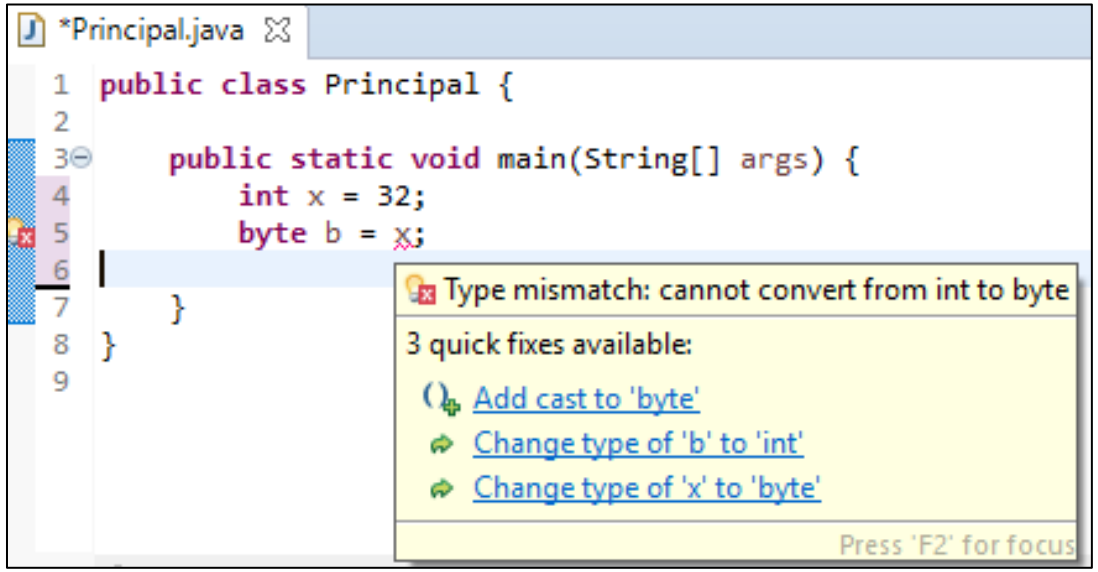
1. Colocar um "f" (ou F) ao final do número, **float f = 32.5f;**
2. Alterar o tipo de float para double, portanto ficaria:

double f = 32.5;

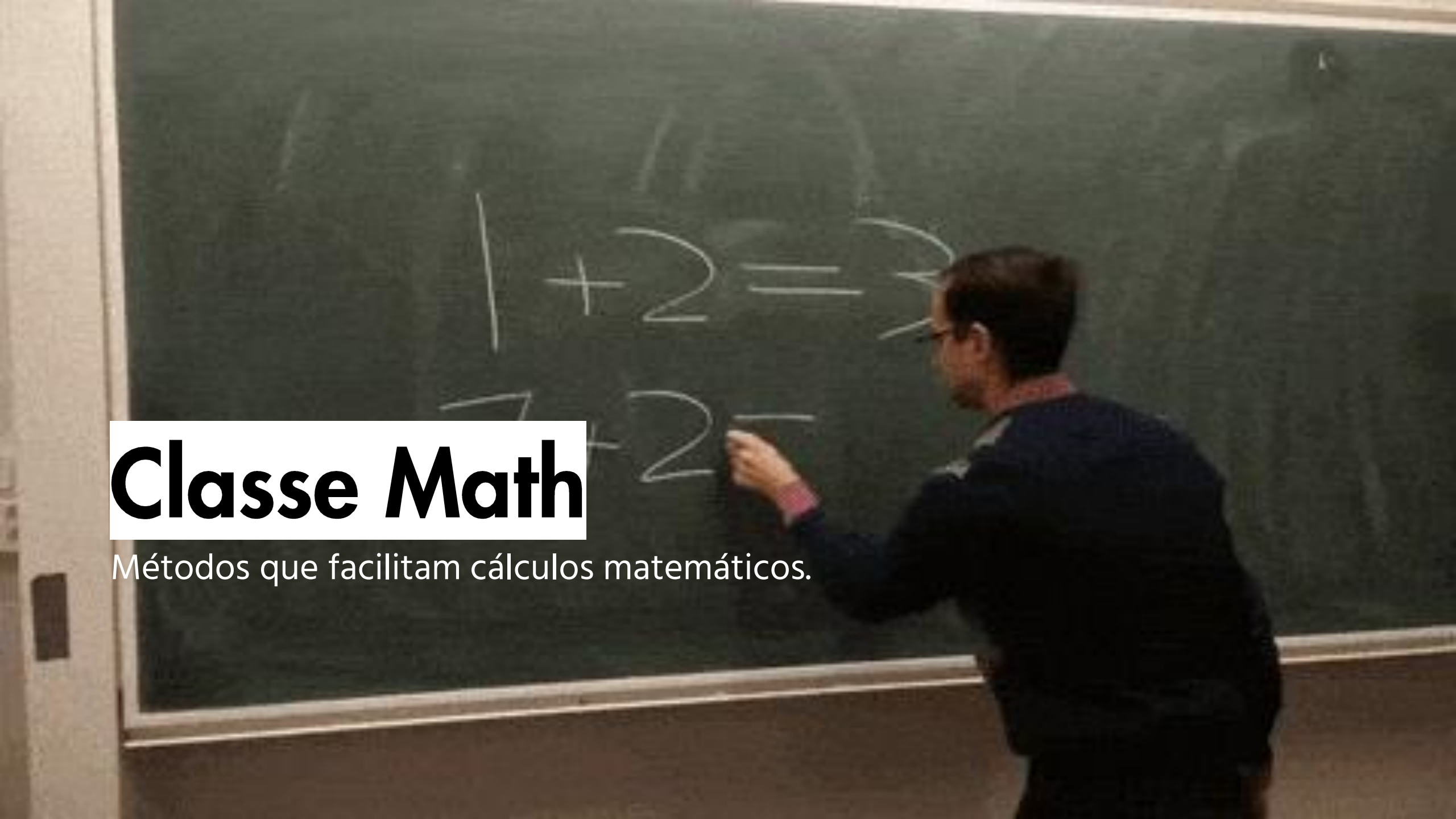
Matutando o problema

Por que o trecho abaixo está errado?

```
int x = 32;
byte b = x;
```



O tamanho de uma variável do tipo byte é de 8 bits. Portanto consegue armazenar valores de -128 a 127. Qualquer valor fora desse escopo, será **"adaptado"** para um valor dentro desse escopo. Uma solução é realizar o casting de int para byte. No caso acima, é possível realizar o casting para byte sem perder informação. No entanto, dependendo do valor da variável x, pode ocorrer perda de informação.



Classe Math

Métodos que facilitam cálculos matemáticos.

Métodos da classe Math

A classe **Math** disponibiliza diversos métodos para uma gama de operações.

Métodos da Classe Math

Considere: double **x**; int **i**; long **lo**;

- x = Math.abs(-3); // 3 (módulo)
- x = Math.pow(2,3); // 8 (potência)
- x = Math.sqrt(16); // 4 (raiz)
- lo = Math.round(2.75); // 3 (arredonda)
- **x = Math.random();** // sorteio aleatório (0 a 0.9999)
- x = Math.sin(3.1415); // 0
- x = Math.cos(0.0); // 1
- x = Math.toDegrees(3.1415); // 180
- x = Math.toRadians(180); // 3.1415...
- x = Math.PI; // valor de pi

Exercício 1

Crie um programa para sortear 3 números (entre 0 e 99).

Em seguida mostre os números sorteados para o usuário.

Início da Solução

Para sortear um número entre 0 e 100, deve-se multiplicar o resultado do Math.random() por 100, em seguida fazer o casting para um número inteiro.

...

```
int x = ( int ) ( Math.random() * 100 );
System.out.println(x);
```

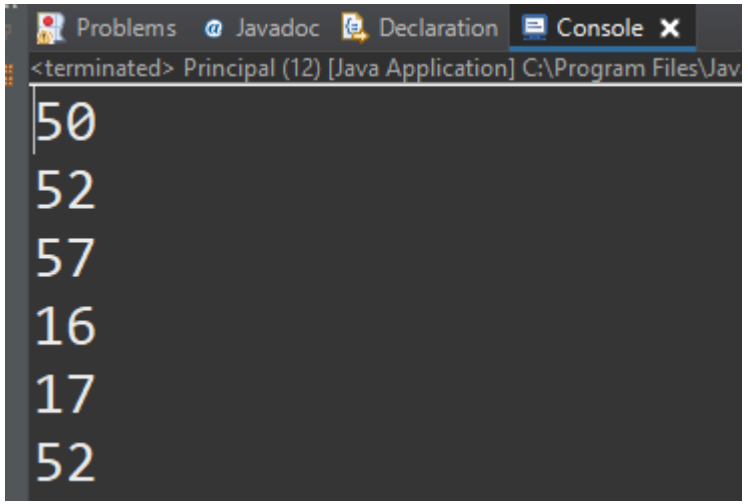
Exercício 1 - Resposta

Exercício 2

- Crie um programa que sorteia os 6 números da mega sena.
- Desafio: Os números não podem ser repetidos.

Exercício 2 - Resposta

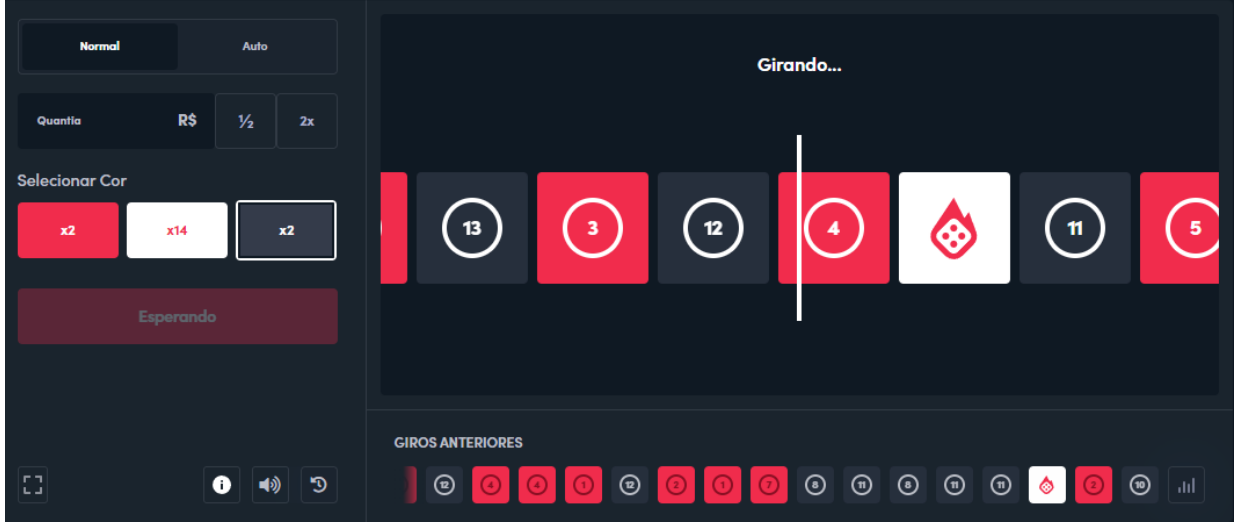
```java
public static void exercício3() {
    int n1 = (int) (Math.random() * 60);
    int n2 = (int) (Math.random() * 60);
    int n3 = (int) (Math.random() * 60);
    int n4 = (int) (Math.random() * 60);
    int n5 = (int) (Math.random() * 60);
    int n6 = (int) (Math.random() * 60);
    System.out.println(n1);
    System.out.println(n2);
    System.out.println(n3);
    System.out.println(n4);
    System.out.println(n5);
    System.out.println(n6);
}
```



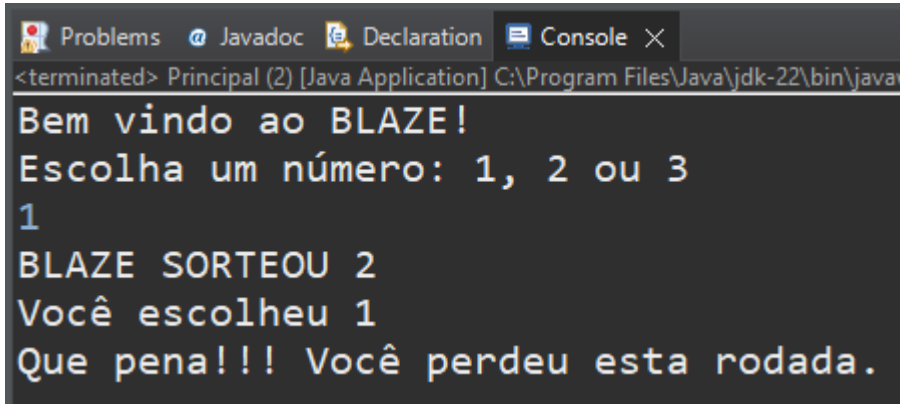
Exercício 2 – Resposta Desafio

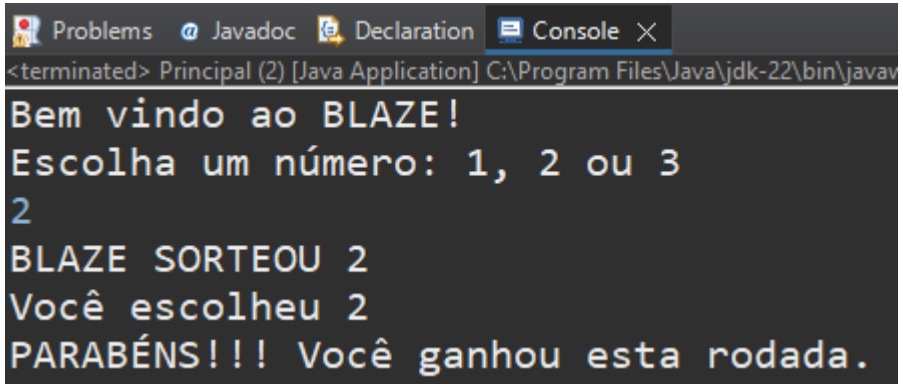
```
28      public static void exercício3() {
29          int total = 0;
30          ArrayList<Integer> numeros = new ArrayList<Integer>();
31          while (total<6) {
32              int n = (int) (Math.random() * 60);
33              if (numeros.contains(n) ) {
34                  continue;
35              } else {
36                  numeros.add(n);
37                  total++;
38              }
39          }
40
41          for (int n: numeros) {
42              System.out.print(n + " ");
43          }
44      }
```

Exercício 3



- Faça um programa para sortear números de 1 a 3. Simulando as cores do Blaze (vermelho, preto ou branco).
- Pergunte ao usuário qual a escolha dele (entre 1 e 3).
- Na sequência faça o sorteio e diga se o usuário ganhou ou perdeu.





Exercício 3 - Resposta

```java
import java.util.Scanner;
public class Principal {
    public static void main(String[] args) {
        System.out.println("Bem vindo ao BLAZE!");
        System.out.println("Escolha um número: 1, 2 ou 3");
        Scanner teclado = new Scanner(System.in);
        int escolha = teclado.nextInt();
        int sorteio = (int)( Math.random() * 3) + 1;
        if (escolha==sorteio) {
            System.out.println("BLAZE SORTEOU "+sorteio);
            System.out.println("Você escolheu "+escolha);
            System.out.println("PARABÉNS!!! Você ganhou esta rodada." );
        } else {
            System.out.println("BLAZE SORTEOU "+sorteio);
            System.out.println("Você escolheu "+escolha);
            System.out.println("Que pena!!! Você perdeu esta rodada." );
        }
    }
}
```

Exercício 4

Implemente um programa em Java que leia as coordenadas de dois pontos e calcule a distância entre eles. Lembre-se que a distância entre dois pontos é dada pela seguinte equação:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Utilize as funções sqrt e pow da classe Math para calcular a raiz quadrada e elevar os números as suas devidas potencias.

Exercício 5

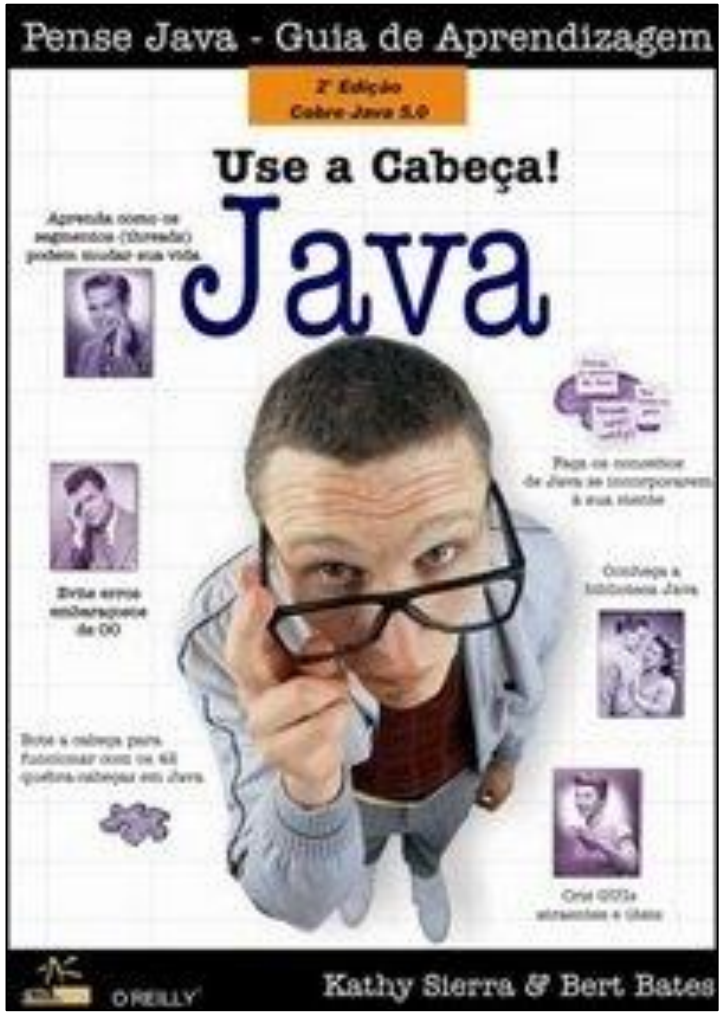
Crie um programa em Java que permita calcular o peso de uma pessoa em vários planetas. O programa deve perguntar o peso do usuário na terra e exibir na tela o peso do usuário em todos os planetas.

Gravidade relativa	Planeta
0,37	Mercúrio
0,88	Vênus
0,38	Marte
2,64	Júpiter
1,15	Saturno
1,17	Urano

A equação para calcular o peso em outro planeta é:

$$P_{planeta} = \frac{P_{Terra}}{10} * g_{planeta}$$

A partir de agora, já pode começar a ler o livro!



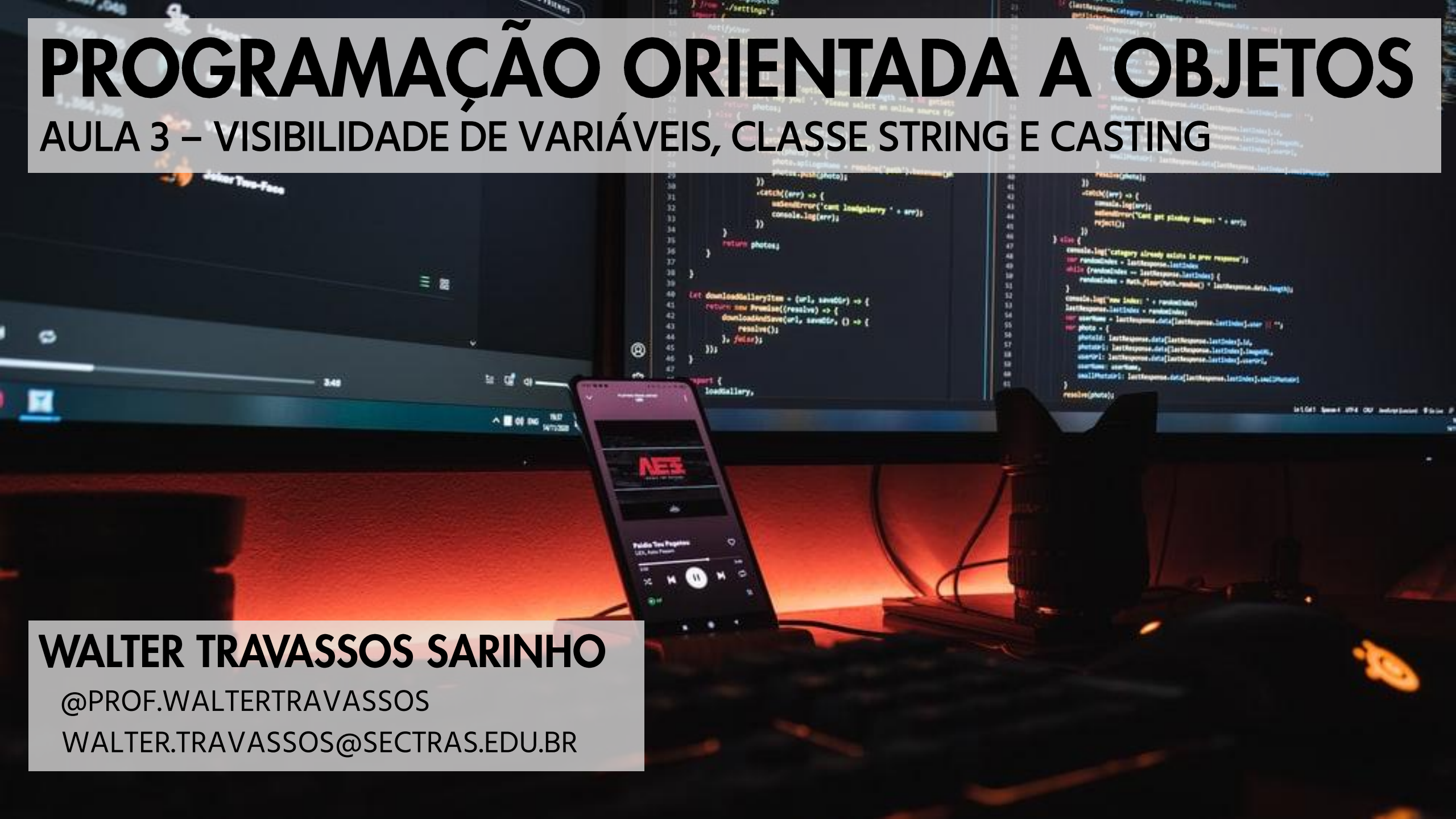
Capítulos 1 e 2.

SIERRA, K.; BATES, B. **Use a Cabeça! Java**. 2. ed. Rio de Janeiro: Alta Books, 2010.

Já está no ambiente virtual o pdf!

PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 3 – VISIBILIDADE DE VARIÁVEIS, CLASSE STRING E CASTING



WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR