Introdução a Classes e Objetos

Professor: Henrique Delegrego

Introdução a Classes e Objetos

O que são classes

- Classes são estruturas ou moldes que definem a forma e comportamento de um objeto
- Um objeto é uma instância de uma classe com estado e comportamento específico

Introdução a Classes e Objetos

Exemplo de Classe e Objeto

Classe Veiculo

- Modelo
- Comprimento
- Passageiros
- Velocidade Máxima
- Cor

Objeto Civic

- Modelo = “Civic”
- Comprimento = 4.70
- Passageiros = 5
- Velocidade Máxima = 200
- Cor = “Preto”

(Atributos de classe)

Introdução a Classes e Objetos

Classe no Java

Classe Veiculo

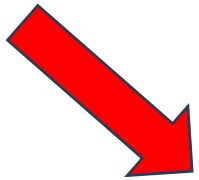
- Modelo
- Comprimento
- Passageiros
- Velocidade Máxima
- Cor

```
public class Veiculo {
    String modelo;
    double comprimento;
    int passageiros;
    int velMax;
    String cor;
}
```

Introdução a Classes e Objetos

Instanciação de um objeto

```
2  public class Principal {
3
4      public static void main(String[] args) {
5          Veiculo civic = new Veiculo();
6          Veiculo sandero = new Veiculo();
7          Veiculo kwid = new Veiculo();
8
9      }
10
11 }
```



Criação de um objeto

Introdução a Classes e Objetos

```
public class Principal {

    public static void main(String[] args) {
        Veiculo civic = new Veiculo();

        civic.modelo = "Civic";
        civic.comprimento = 4.7;
        civic.passageiros = 5;
        civic.velMax = 200;
        civic.cor = "Preto";
        System.out.println("Velocidade max civic: " + civic.velMax + " km/h");

    }

}
```

```
Velocidade max civic: 200 km/h
```

Agora: Façam o exercício 1 a 3

Introdução a Classes e Objetos

Métodos em classes

- Vão definir os comportamentos / funções de um objeto

```
3  public class Quadrilatero {
4
5      double largura;
6      double altura;
7
8      public double calcularArea() {
9          return largura * altura;
10     }
11
12 }
```

```
5      public static void main(String[] args) {
6
7          Quadrilatero q1 = new Quadrilatero();
8          q1.altura = 2;
9          q1.largura = 3;
10
11         System.out.println(q1.calcularArea());
12
13     }
```

Agora: Façam o exercício 4 a 6

Introdução a Classes e Objetos

Método construtor

- É usado para inicializar o objeto, passando os valores dos atributos
- Chamado automaticamente quando um objeto é criado
- Terá o mesmo nome da classe
- Toda classe tem um construtor padrão
- Podem ser criados vários construtores

Introdução a Classes e Objetos

Método construtor

```
public class Veiculo {
    // Atributos de classe
    String modelo;
    double comprimento;
    int passageiros;
    int velMax;
    String cor;

    // Construtor
    public Veiculo(String modelo, double comprimento, int passageiros, int velMax, String cor) {
        this.modelo = modelo;
        this.comprimento = comprimento;
        this.passageiros = passageiros;
        this.velMax = velMax;
        this.cor = cor;
    }

}
```

Introdução a Classes e Objetos

Método construtor

```
Veiculo civic = new Veiculo();
```

```
Veiculo civic = new Veiculo("Civic", 4.7, 5, 200, "Preto");
```

Agora: Vamos criar o nosso método construtor

Agora: Façam o exercício 7

Introdução a Classes e Objetos

Encapsulamento

- Usado para limitar o acesso a atributos e métodos
- Também permite esconder detalhes internos de implementação
- Pode ser por motivos de segurança ou por motivos de validação de valores

```
private String modelo;
private double comprimento;
private int passageiros;
private int velMax;
private String cor;
```

Introdução a Classes e Objetos

Métodos get e set

- Fornecem uma maneira controlada de acessar os atributos de uma classe
- Necessários após encapsular os atributos

Getter:

- Um getter é um método que “pega“ o valor de um atributo privado de uma classe
- Normalmente não recebe parâmetro

Setter:

- Um setter é um método que define o valor de um atributo privado da classe
- Tem como parâmetro o novo valor do atributo

Agora: Vamos fazer os métodos get e set

Introdução a Classes e Objetos

Métodos get e set

- Podem ser colocadas validações nos métodos **set**

```
public void setAltura(double altura) {
    if (altura <= 0) {
        System.out.println("Erro, altura inválida");
    } else {
        this.altura = altura;
    }
}
```

Introdução a Classes e Objetos

Métodos get e set

- O construtor deve ser modificado para fazer atribuições pelos **sets**

```
public Quadrilatero(double altura, double largura) {
    setAltura(altura);
    setLargura(largura);
}
```

Introdução a Classes e Objetos

Método toString

- Quando você cria um objeto em Java e tenta exibi-lo diretamente, ele não gera uma saída legível por padrão
- Fornece uma representação em formato de texto de um objeto
- Criaremos um método na classe do objeto para imprimir os conteúdos dele
- Será necessário sobrescrever o método toString já existente por padrão

```
@Override
public String toString() {
    // Na classe veículo
    return "Informações do veículo:\nModelo: " + modelo + " (" + cor + ")\nComprimento: " + comprimento
            + " metros\nPassageiros: " + passageiros + "\nVel max: " + velMax + " km/h";
}
```

Introdução a Classes e Objetos

Método toString

```
public static void main(String[] args) {
    Veiculo civic = new Veiculo(4.7, 5, 200, "Preto", "Civic");
    System.out.println(civic.toString());
    // O toString é opcional
}
```

```
Informações do carro:
Modelo: Civic (Preto)
Comprimento: 4.7 metros
Passageiros: 5
Vel max: 200 km/h
```

Agora: Façam o exercício 8

Introdução a Classes e Objetos

Tratamento de Exceção

- Uma **exceção** é um evento inesperado que interrompe o fluxo normal do programa
- Pode ser causada por diversos fatores, como entrada inválida do usuário, arquivos não encontrados, divisão por zero ou problemas de conexão

```
int vetor[] = new int[5];

vetor[5] = 20;
```

Exceção != Erro

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
        at Exemplo1.main(Exemplo1.java:8)
```

Introdução a Classes e Objetos

Tratamento de Exceção

- Nomenclaturas do Java:
  - Uma exceção é lançada (**throw**)
  - Podemos pegar ela (**catch**) para que o programa não pare
- Para tratar exceções usamos:
  - Bloco de try catch
  - throw

Introdução a Classes e Objetos

Bloco try-catch

- O bloco **try-catch** permite que o programa lide com erros de forma controlada, sem falhar abruptamente
- O bloco **try** contém o código que pode gerar uma exceção, enquanto o **catch** captura e trata essa exceção caso ocorra
- Se uma exceção for lançada dentro do **try**, o fluxo do programa será transferido para o catch

```
6           Scanner input = new Scanner(System.in);
7
8           int vetor[] = new int[5];
9
10          try {
11              int i = input.nextInt();
12              vetor[5] = i;
13          } catch (Exception e) {
14              System.out.println(e);
15          }
16          System.out.println("O programa continuou");
17
18      }
```

Introdução a Classes e Objetos

Finally

- O bloco **finally** sempre será executado, independentemente de ocorrer uma exceção ou não
- Utilizado para liberar recursos, como arquivos e conexões com o banco de dados

```java
Scanner input = new Scanner(System.in);

int vetor[] = new int[5];

try {
    int i = input.nextInt();
    vetor[5] = i;
} catch (Exception e) {
    System.out.println("Não tem esse índice");
} finally {
    // Tudo que está aqui vai rodar
    // Independente se aconteceu algum exceção ou não
    // Usado para limpeza do código e legibilidade
    input.close(); // Por exemplo
}
```

Introdução a Classes e Objetos

Lançamento de Exceção

- O comando **throw** é usado para lançar exceções explicitamente
- Diferente do **try-catch**, que serve para tratar exceções, o **throw** permite que o próprio código gere uma exceção quando uma situação inesperada ocorre

```
11      public void setPreco(double preco) {
12          if (preco <= 0) {
13              throw new IllegalArgumentException("Preço inválido");
14          }
15          this.preco = preco;
16      }

21      public void setNome(String nome) {
22          if (nome == null || nome.isEmpty() || nome.isBlank()) {
23              throw new IllegalArgumentException("Nome invalido");
24          }
25          this.nome = nome;
26      }
```

Introdução a Classes e Objetos

Lançamento de Exceção

```
2  public class Principal {
3
4      public static void main(String[] args) {
5
6          Pessoa p1 = new Pessoa("    ");
7      }
8
9  }
```

Console

<terminated> Principal (5) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (6 Oct 2023, 00:44:30 – 00:44:30)

```
Exception in thread "main" java.lang.IllegalArgumentException: Nome invalido
        at Pessoa.setNome(Pessoa.java:29)
        at Pessoa.<init>(Pessoa.java:19)
        at Principal.main(Principal.java:6)
```

Agora: Terminem a lista de exercícios

entra21