

## Aula 2

### Programação Orientada a Objetos

Prof. Leonardo Gomes

1

### Conversa Inicial

2

- ▀ Classes e atributos
- ▀ Métodos
- ▀ Padrões e modificador *static*
- ▀ Interação entre objetos
- ▀ Construtores

3

### Classes e atributos

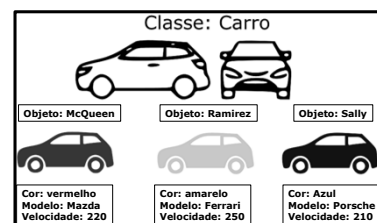
4

### Classe

- ▀ Classificação de objetos
- ▀ Abstração para modelar o mundo
- ▀ Aproximação da lógica da programação com a da linguagem falada

5

### Classe carro



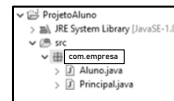
6

### Classe em Java

```
package com.empresa;  
  
public class PrimeiraClasse {  
  
    public static void main(String[] args) {  
        System.out.println("Alo Mamae");  
    }  
}
```

7

### Classe aluno



```
package com.empresa;  
  
public class Aluno {  
    String nome;  
    int matricula;  
    String cpf;  
}
```

8

### Classe aluno

```
public static void main(String[] args) {  
    Aluno mario = new Aluno();  
    mario.cpf="111.111.111-1";  
    mario.nome="Super Mario";  
    mario.matricula=1001;  
}
```

9

### Métodos

10

- **Atributos**
  - Do que o objeto é composto
- **Métodos**
  - O que o objeto faz
- **Estado**
  - Em que situação o objeto está

11

### Classe aluno

```
public class Aluno {  
    String nome;  
    int matricula;  
    String cpf;  
    public void info(){  
        System.out.println("nome: " + nome);  
        System.out.println("matricula: " + matricula);  
        System.out.println("cpf: " + cpf);  
    }  
}
```

12

### Classe aluno

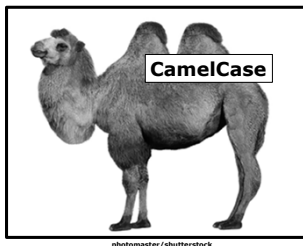
```
Aluno mario = new Aluno();  
mario.cpf="111.111.111-1";  
mario.nome="Super Mario";  
mario.matricula=1001;  
  
Aluno luigi = new Aluno();  
luigi.cpf="222.222.222-2";  
luigi.nome="Super Luigi";  
luigi.matricula=1002;  
  
mario.info();  
luigi.info();
```

13

### Padrões e modificador *static*

14

### Nomenclatura



15

### Nomenclatura

- Pacotes: são descritos inteiramente em letras minúsculas
- Classes: inicia com letra maiúscula e segue o *camel case*
- Métodos, atributos e variáveis: inicia com letra minúscula e segue o *camel case*
- Constantes: inteiramente com letras maiúsculas separadas por *underline*

16

### *Static*

- Método *static*
  - Método é executado sem estar associado a uma instância
- Atributo *static*
  - O atributo se torna global. É acessada a mesma variável, mesma posição de memória, por todas as instâncias

17

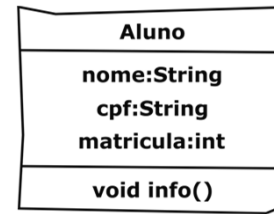
### Interação entre objetos

18

## UML

- **Unified Modeling Language**
- Define padrões para diversas etapas do desenvolvimento de *softwares*
- Muito conhecido pelo diagrama de classes

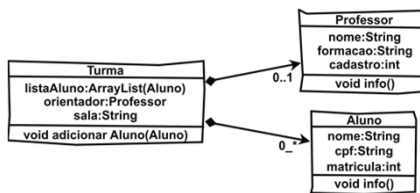
## Diagrama de classe



19

20

## Relação de classes



21

## Professor

```
public class Professor {
    String nome;
    String formacao;
    int cadastro;

    void info();
}
```

22

## Turma

```
package com.empresa;
import java.util.ArrayList;

public class Turma {
    ArrayList<Aluno> listaAlunos = new ArrayList();
    Professor orientador;
    String sala;

    void adicionarAluno(Aluno aluno) {
        listaAlunos.add(aluno);
    };
}
```

## Construtores

23

24

### Construtores

- Código que executa no momento da instanciação
- Semelhante a um método
- Não tem valor de retorno
- Não pode ser invocado sem ser durante a instanciação

25

### Classe aluno

```
public class Aluno {  
    String nome;  
    int matricula;  
    String cpf;  
  
    Aluno(String nome, int matricula, String cpf){  
        this.nome = nome;  
        this.matricula = matricula;  
        this.cpf = cpf;  
    }  
    Aluno(String nome){  
        this.nome = nome;  
    }  
    Aluno(){  
        System.out.println("Construtor sem parâmetro");  
    }  
}
```

26

### Construtor

- `Aluno mario = new Aluno("Super Mario", 1001, "987.654.321-00" );`
- `Aluno mario = new Aluno("Super Mario");`
- `Aluno mario = new Aluno();`

27