Classes e Objetos Parte 1

Sumário

- . Classes
- . Objetos
- . Construtores
- . Variáveis de Instância
- . Métodos de Instância
- . Sobrecarga de Métodos



O que são classes?

Uma classe é uma estrutura que define o comportamento e o estado de um objeto. Ela serve como um modelo (ou planta) para criar objetos.

Uma classe pode conter campos (variáveis) para representar o estado do objeto, e métodos (funções) para definir os comportamentos.



O que são classes?

```
public class Time {
 private String nome;
 private int gols;
 private List<Titulo> titulos;
 public String getNome() {
    this.nome = nome;
 public void adicionaTitulo(Titulo titulo) {
    this.titulos.add(titulo);
```



O que são objetos?

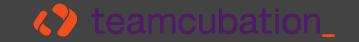
Um objeto é uma instância de uma classe. Podemos dizer que um objeto representa uma entidade do mundo real e possui características associadas a ele.

Os objetos podem ser criados a partir do operador new e podem ser atribuídos a uma variável para ser manipulada.



O que são objetos?

```
Time selecaoBrasileira = new Time();
time.adicionaTitulo(new Titulo("Copa America"));
```



Um construtor é um método especial que é chamado automaticamente quando um objeto é criado.

Geralmente é utilizado para inicializar o estado do objeto.

Em Java o construtor tem o mesmo nome da classe, não tem um tipo de retorno explícito, e quando não há um construtor escrito pelo programador, a aplicação assume que a classe possui um construtor default.



Classe

```
public class Time {
   // Construtor default
   public Time() {
   }
}
```

```
public class Time {
}
```

Instanciação

```
Time time = new Time();
```

Equivalentes



Classe

```
. .
public class Time {
  private String nome;
  public Time(String nome) {
    this.nome = nome;
```

Instanciação

```
Time time = new Time("Sao Paulo")
```

Erro

```
Time time = new Time(); // ERRO!!
```



Classe

```
public class Time {
  private String nome;
  public Time() {
    this.nome = "Sem nome";
  public Time(String nome) {
    this.nome = nome;
```

Instanciação

```
Time mandante = new Time("Sao Paulo");
Time visitante = new Time();
```



Quase tudo o que se pode fazer dentro de um método é possível fazer dentro de um construtor. Exemplos de coisas que podem ser feitas em um construtor:

- instanciar variáveis;
- criar e usar objetos de outras classes;
- chamar outros métodos da mesma classe;
- chamar outros construtores da mesma classe (possui regras especiais).



Variáveis de instância são aquelas declaradas dentro de uma classe, mas fora dos métodos. Elas representam o estado do objeto. Podemos chamá-las de campos ou atributos.

Ao criar um objeto, cada instância deste objeto terá sua própria cópia dessas variáveis.



As variáveis de instância (ou campos do objeto) podem ser acessadas da seguinte forma:

- de fora do objeto, através da variável na qual o objeto está representado, seguida do operador ponto "." + o nome do campo. Obs.: a variável precisa ter visibilidade pública, ou ter visibilidade default e estar sendo chamada dentro do mesmo pacote (assunto para próximas aulas).
- de dentro do próprio objeto, chamando diretamente o nome do campo, ou através da referência this + operador ponto + o nome do campo, em casos especiais.

```
public class Time {
 private String nome;
 public String getNome() {
   return nome;
 public void setNome(String nome) {
   this.nome = nome;
```

```
Time time = new Time();
time.setNome("Sao Paulo");

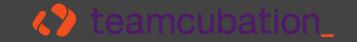
String nomeDoTime = time.getNome();
// Sao Paulo
```



```
public class Time {
  public String nome;
}
```

```
Time time = new Time();
time.nome = "Sao Paulo";

String nomeDoTime = time.nome;
// Sao Paulo
```



Métodos de instância são métodos declarados em uma classe, e que podem ser chamados através das instâncias de objetos dessa classe.

Eles operam nos dados da instância do objeto e podem acessar e modificar as variáveis de instância.



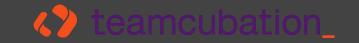
Os métodos de instância podem ser chamados da seguinte forma:

- de fora do objeto, através da variável na qual o objeto está representado, seguida do operador ponto "." + o nome do método + possíveis parâmetros. Obs.: o método precisa ter visibilidade pública, ou ter visibilidade default e estar sendo chamado dentro do mesmo pacote (assunto para próximas aulas).
- de dentro do próprio objeto, com ou sem a referência this, seguida do operador ponto + o nome do método + possíveis parâmetros.



```
public class Time {
 private String nome;
 public String getNome() {
    return this.nome;
  public void setNome(String nome) {
   this.nome = nome;
```

```
Time a = new Time();
Time b = new Time();
a.setNome("Sao Paulo");
b.setNome("Palmeiras");
System.out.println(a.getNome()); // Sao Paulo
System.out.println(b.getNome()); // Palmeiras
```



```
public class Time {
  public String nome;
  public int gols;
  public Time(String nome, int gols) {
    this.nome = nome;
    this.gols = gols;
  public String getNome() {
    return nome;
  public int getGols() {
    return gols;
  public String getInformacoes() {
    return this.getNome() + ":" + getGols();
```



Sobrecarga de Métodos

Em um mesmo objeto podemos ter mais de um método (ou construtor) com o mesmo nome, desde que os parâmetros sejam diferentes em:

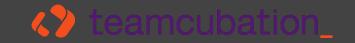
- quantidade de parâmetros; ou,
- tipo de variáveis; ou,
- ordem dos parâmetros.



Sobrecarga de Métodos

```
public class Time {
  private String nome;
  public Time() {
    this.nome = "Sem nome";
  public Time(String nome) {
    this.nome = nome;
```

```
public class Time {
  private String nome;
  private int gols;
  public void setDados(String nome) {
   this.nome = nome;
   gols = 0;
  public void setDados(String nome, int gols) {
   this.nome = nome;
   this.gols = gols;
```



Sobrecarga de Métodos

Vantagens:

- flexibilidade: mesmo que eu tenha tipos diferentes consigo utilizar os comportamentos daquela classe;
- facilidade de uso: não precisar saber vários nomes diferentes para tipos de parâmetros diferentes;
- reutilização de código: em algumas situações podemos até reutilizar o mesmo algoritmo para diferentes tipos de parâmetros.



Perguntas?