Linguagem de Programação Orientada a Objetos I

Polimorfismo e Construtores

Prof. Tales Bitelo Viegas
https://fb.com/ProfessorTalesViegas

Polimorfismo

- Definição de Polimorfismo
  - O polimorfismo indica a capacidade de abstrair várias implementações diferentes em uma única interface
- Como são definidos:
  - Diferentes assinaturas: mesmo nome mas com parâmetros diferentes (em número e tipo)
- Polimorfismo estático:
  - Também conhecido como sobrecarga de métodos (overloading)
  - A decisão é feita pelo compilador baseado na assinatura dos métodos

Polimorfismo

- Exemplo: métodos polimórficos

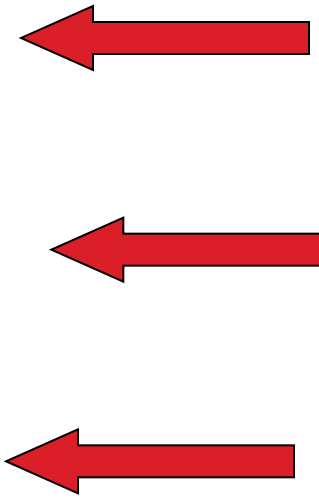
```
public void setVar(String n){
    nome = n;
}

public void setVar(String n, String em){
    nome = n;
    eMail = em;
}
```

Métodos Polimórficos

Polimorfismo

- Exemplo: métodos polimórficos

```
void alteraCampo(int id){
    idade = id;
}
void alteraCampo(String n){
    nome = n;
}
void alteraCampo(char s){
    sexo = s;
}
```



Métodos Polimórficos

Construtores

- Comandos a serem executados uma única vez, no momento da criação do objeto (new)
- Nome do construtor= nome da classe
- Construtores não possuem tipo de retorno (nem mesmo *void*)
- Quando nenhum construtor é definido, Java define um construtor padrão (que simplesmente instancia o objeto)

Construtores

- Construtor geralmente fornece valores iniciais (inicialização) para o(s) campo(s)
- Pode existir mais de um construtor para cada classe (diferentes assinaturas)
  - Padrão: construtor não parametrizado
  - Demais construtores: diferentes parâmetros

Construtores

```
public class Aluno{
   private String nome;
   private int cgu;
   private String dataNascimento;

   // Construtor
   public Aluno(){
        nome = "";
        cgu = 0;
        dataNascimento = "";
   }
   // demais metodos
}
```

Método Construtor sem parâmetros

Construtores

```
public class Aluno{
   private String nome;
   private int cgu;
   private String dataNascimento;

   // Construtor
   public Aluno(String n, int c, String d){
        nome = n;
        cgu = c;
        dataNascimento = d;
   }
   // demais metodos
}
```

Método Construtor com parâmetros

Construtores

Utilizando um Construtor

```
public class Main{

   public static void main(String[] args) {
        int idadeAluno;
        Aluno aluno1=new Aluno("Tales", 123, "06/09");
        idadeAluno = aluno1.calculaIdade();
        System.out.println("Idade do aluno:"+ idadeAluno);
   }
}
```

Construtores

- Se um construtor com parâmetros for criado, deverá ser criado explicitamente um construtor padrão. Caso contrário, devemos sempre instanciar os objetos com os parâmetros:

```
public class Aluno{
    private String nome;
    private int cgu;
    private String dataNascimento;

    // Construtor padrao
    public Aluno(){
    }
    // Construtor
    public Aluno(String n){
          nome = n;
          cgu = 0;
          dataNascimento = "";
    }
    // demais metodos
}
```

Palavra Reservada *static*

- Atributos Estáticos
  - Atributos declarados como *static* são chamados atributos de classe
  - O atributo estático será o mesmo para todas as instâncias (independente do número de instâncias da classe)
  - Exemplos:
    - Se uma instância alterar valor todas as outras instâncias irão detectar esta mudança

Palavra Reservada *static*

```
public class Pessoa {
    private String nome;
    private int idade;
    private static int nroPessoas;
    Pessoa(String nome, int idade) {
          this.nome = nome;
          this.idade = idade;
          this.nroPessoas ++;
    }
    public int getNroInstancias(){
           return nroPessoas;
    }
}
```

Palavra Reservada *static*

- Métodos/atributos definidos como estáticos podem ser utilizados sem que seja necessário criar uma instância da classe à qual pertencem

- Exemplos :
  - System.out: out é um atributo estático da classe System que mapeia a saída padrão (stdout)
  - public static void main (String[] args): é possível executar o método main, sem que seja necessário criar uma instância da classe Principal

Colocando em prática

- Exercícios
  - Ver no Moodle.