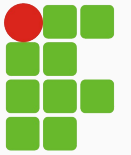
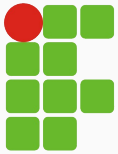


Programação orientada a objetos

Maximilian Jaderson de Melo

Aula 10



Introdução a Orientação a objetos

Conteúdo de hoje

- Polimorfismo.
  - Sobrescrita.
  - Paramétrico.

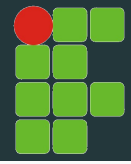
Sobrescrita

- Quando um método definido na superclasse deve ser executado de maneira diferente em cada subclasse.

Sobrescrita

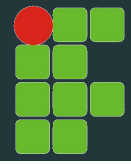
- Classe Animal apresenta métodos para:
  - Comer.
  - Dormir.
  - Fazer ruído.
  - Mover.

Sobrescrita



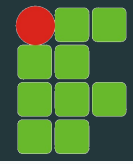
- Se o método é implementado na super classe, todas as subclasses compartilharão o mesmo comportamento.
- O som que todo animal emite é o mesmo?
  - Cães latem.
  - Gatos miam.
  - Galinhas cacarejam.
- A implementação deveria então sair da superclasse?

Sobrescrita em Java



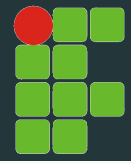
```java
//classe Animal
public class Animal{
   public void fazerRuido(){
        System.out.println("ruido animal");
   }
}
```

Sobrescrita em Java



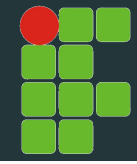
```java
//classe Cachorro
public class Cachorro extends Animal{
   public void fazerRuido(){
        System.out.println("wooof");
   }
}
```

Sobrescrita em Java



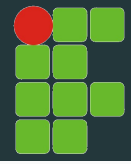
```java
//classe Gato
public class Gato extends Animal{
   public void fazerRuido(){
        System.out.println("meow");
   }
}
```

Sobrescrita em Java



```java
//classe Main
public class Main{
  public static void main(String[] args){
      Animal a = new Animal();
      Cachorro c = new Cachorro();
      Gato g = new Gato();
      a.fazerRuido();//ruido animal
      c.fazerRuido();//wooof
      g.fazerRuido();//meow
  }
}
```

Sobrescrita em Java

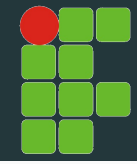


- Perceba que a classe Animal também apresenta o método fazerRuido.
  - Dessa forma, é possível que um objeto animal possa executar o método.
  - Todos os animal fazem ruídos, mas a ideia fazer ruído não faz sentido se não estiver associada a um animal concreto.
  - Pensando bem, a classe animal também não é concreta.
- A classe Animal não deveria ser instanciável.

Sobrescrita em Java

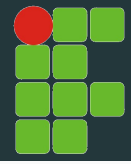
```java
//classe Animal
public abstract class Animal{
  public abstract void fazerRuido();
}
```

Sobrescrita em Java



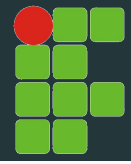
```java
//classe Main
public class Main{
  public static void main(String[] args){
      Cachorro c = new Cachorro();
      Gato g = new Gato();

      c.fazerRuido();//wooof
      g.fazerRuido();//meow
  }
}
```

Sobrescrita em Java



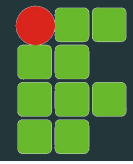
- Quando a classe é abstrata, não é instanciável.
- Quando o método é abstrato na superclasse, não é adicionada a implementação.
  - Também obriga que todas as subclasses implementem sua versão do método.
- No exemplo do slide anterior, todas as subclasses de Animal são obrigadas a implementar o método fazerRuido.

Instancias polimórficas



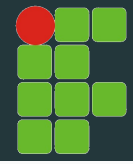
- Se gato e cachorro são subclasses de uma mesma superclasse, a declaração do objeto pode ser feita a partir do tipo da superclasse.
  - A instância segue sendo pela subclasse.

Instancias polimórficas



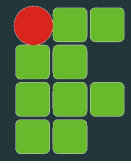
```
//classe Main
public class Main{
  public static void main(String[] args){
      Animal c = new Cachorro();
      Animal g = new Gato();

      c.fazerRuido();//wooof
      g.fazerRuido();//meow
  }
}
```

Polimorfismo paramétrico



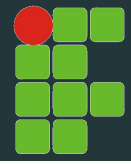
- Quando uma classe deve ser flexível a qualquer tipo de dados.
  - É mais útil em linguagens estaticamente tipadas (o tipo de uma variável deve ser definido antes da compilação).
- É definido pelo operador <>.
- Em comum é utilizado por classes utilitárias para listas (vetores).

Polimorfismo paramétrico



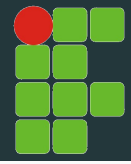
```
//classe Main
public class Main{
  public static void main(String[] args){
      List<Integer> lint = new
        ArrayList<Integer>();
      List<Float> lfloat = new
        ArrayList<Float>();
  }
}
```

Polimorfismo paramétrico



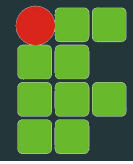
- List é uma classe para declarar listas.
- Dentro do <> é especificado qual tipo deve ser vinculado à classe em tempo de execução.
- Em seguida, é criada uma classe customizada para criar um utilitário para pontos 2D.
    - A classe pode ser aplicada para pontos de números inteiros ou com casa decimal.

Polimorfismo paramétrico



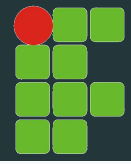
```
public class Ponto <E>{
  private E x;
  private E y;

  public Ponto(E x,E y) {
      this.x = x;
      this.y = y;
  }

  //continua
```

Polimorfismo paramétrico



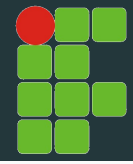
```
//continuação

    public static void main(String[] args) {
        Ponto<Integer> pi = new Ponto<>(2,2);
        Ponto<Double> pf=new Ponto<>(2.0,2.5);
    }
}
```

Polimorfismo paramétrico



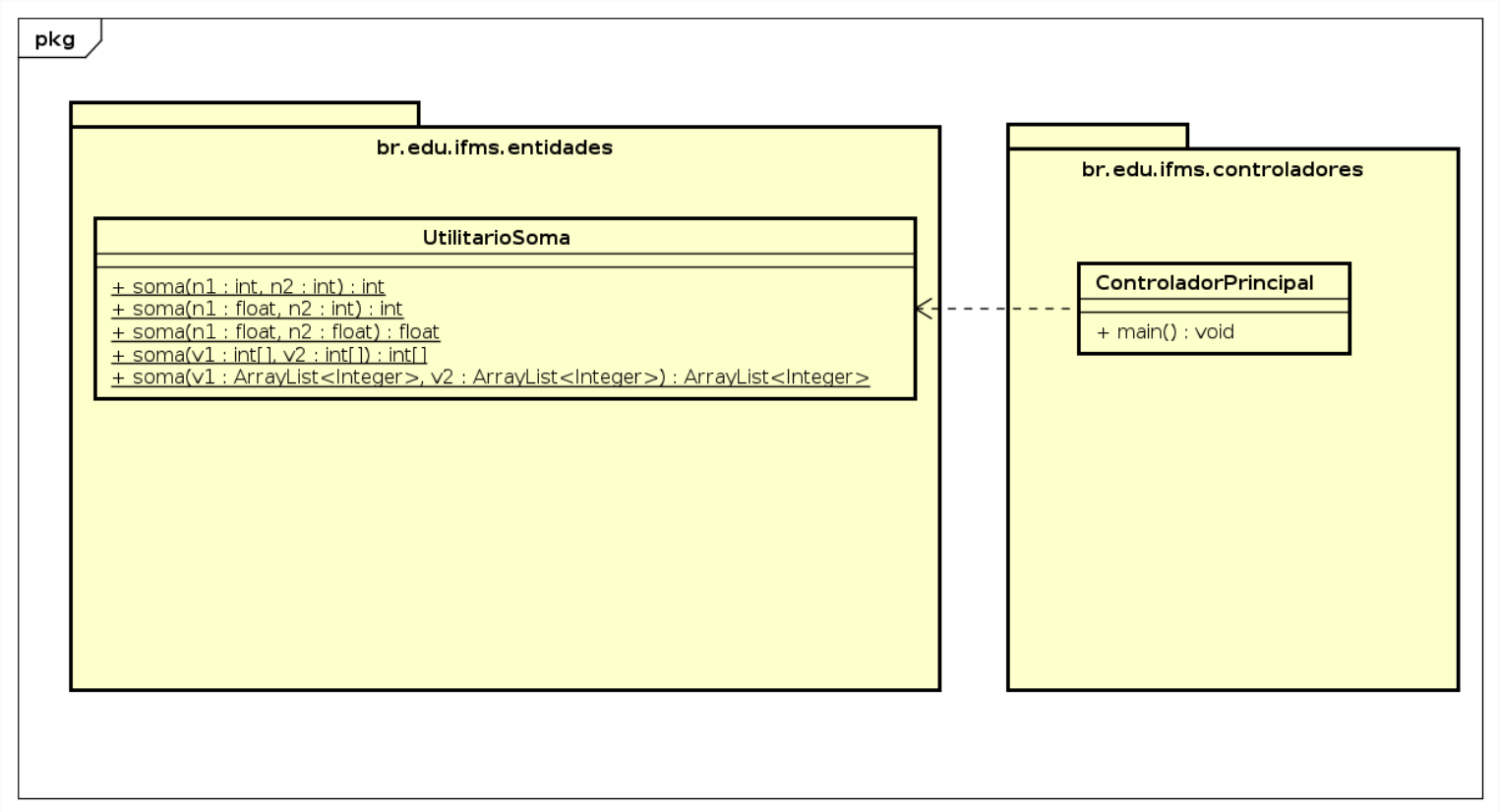
- No exemplo, E indica o recurso “Generics”, que em Java permite esse tipo de polimorfismo.
  - Ele permite vinculação tardia do tipo de dados associado à classe.
- Em tempo de execução, E é substituido pelo tipo informado dentro dos <>.
  - pi utiliza o tipo inteiro.
  - pf utiliza o tipo double.

Exercício



1. Considerando o exercício do diagrama de classes de soma (no próximo slide) refaça-o, mas para permitir a soma de diferentes tipos de dados por meio de polimorfismo paramétrico. Não é necessário suportar as versões com vetores.
2. Considerando o projeto do MMORPG, defina um método abstrato na superclasse para imprimir a descrição do personagem. Esse método mostra na tela o nome, do personagem, seu tipo, seus pontos de ataque, de defesa e de vida.

Exercício


pkg
br.edu.ifms.entidades
UtilitarioSoma
+ soma(n1 : int, n2 : int) : int
+ soma(n1 : float, n2 : int) : int
+ soma(n1 : float, n2 : float) : float
+ soma(v1 : int[ ], v2 : int[ ]) : int[ ]
+ soma(v1 : ArrayList<Integer>, v2 : ArrayList<Integer>) : ArrayList<Integer>
br.edu.ifms.controladores
ControladorPrincipal
+ main() : void

Próxima aula

- Polimorfismo em PHP.
- Definição do trabalho prático.

Referências

- [1] COELHO, ALEX. “JAVA-com orientação a objetos.” Editora Ciência Moderna (2012).

Dúvidas, críticas ou sugestões?

maximilian.melo@ifms.edu.br

max.mjm.melo@gmail.com