Semana 8

Polimorfismo

Profª Marina de Lara

Classe Animal

Nós já aprendemos que quando temos um conjunto de objetos semelhantes, podemos representá-los implementando uma classe pai e então criamos classes filhas com as particularidades de cada objeto filho

Polimorfismo

do grego poli = muitas, morfos = formas

“qualidade ou condição do que está sujeito à mudar de forma ou do que se apresenta sob diversas formas”

Onde se espera uma instância de certa classe pode aparecer uma instância de qualquer subclasse daquela classe.

Ou seja, quando usamos o tipo do pai podemos criar um objeto do tipo de qualquer filho

Não é polimorfismo

Mesmo que as classes **Cao**, **Peixe** e **Mamifero** sejam **subclasses** da classe pai **Animal**, quando usamos a referência (tipo) da própria classe para instanciá-la (criar o objeto) não estamos utilizando o polimorfismo.

```
public class Main {
  public static void main(String[] args) {
    Animal animal = new Animal();
    Cao cao = new Cao();
    Peixe peixe = new Peixe();
    Mamifero mamifero = new Mamifero();
  }
}
```

Polimorfismo: Animal e Mamifero

Podemos considerar **polimorfismo** quando utilizamos uma referência (tipo) da **classe pai** para instanciar (criar) um objeto do **tipo filho** (subclasses).

```
public class Main {
  public static void main(String[] args) {
    Animal animal = new Vertebrado();
    Animal animal1 = new SangueFrio();
    Animal animal2 = new Mamifero();
    Animal animal3 = new Peixe();
    Animal animal4 = new Tartaruga();
    Animal animal5 = new Gato();
    …

    Mamifero mamifero = new Cao();
    Mamifero mamifero1 = new Gato();
    Mamifero mamifero2 = new Cavalo();
  }
}
```

Atribuição polimórfica

Podemos realizar uma **atribuição polimórfica** quando **atribuímos** um **novo** objeto de uma **subclasse** à uma **referência polimórfica** (tipo do pai). Ou seja, consideramos a **referência de Mamifero** como **polimórfica** pois a mesma está sendo **atribuída** à **novos** objetos do tipo de suas **subclasses** (classe filha).

```
public class Main {
  public static void main(String[] args) {
    Mamifero mamifero;

    mamifero = new Cao();
    mamifero = new Gato();
    mamifero = new Cavalo();
  }
}
```

Atribuição polimórfica

Podemos realizar uma **atribuição polimórfica** quando **atribuímos** um **objeto existente** de uma **subclasse** à uma **referência polimórfica** (tipo do pai).

```
public class Main {
  public static void main(String[] args) {
    Mamifero mamifero;

    Cao c = new Cao();
    mamifero = c;

    Gato g = new Gato();
    mamifero = g;

    Cavalo c = new Cavalo();
    mamifero = c;
  }
}
```

Atribuição polimórfica

Podemos realizar uma **atribuição polimórfica** quando **vinculamos** um objeto a um **parâmetro** de um **método** que é uma **referência polimórfica** (tipo do pai).

```
public class Zelador {
  public void alimentar(Mamifero m) { ... }
}
```

```
public class Main {
  public static void main(String[] args) {
    Zelador zelador = new Zelador();
    Mamifero mamifero;
    mamifero = new Cao();
    zelador.alimentar(mamifero);
    mamifero = new Gato();
    zelador.alimentar(mamifero);
  }
}
```

Atribuição polimórfica

Podemos realizar uma **atribuição polimórfica** quando **vinculamos** um objeto ao **retorno** de um **método** que é uma **referência polimórfica** (tipo do pai).

```
public class Fazendeiro {
  public Mamifero comprar(int k) {
    Mamifero m;
    switch(k):
    case 1: m = new Cao(); break;
    case 2: m = new Gato(); break;
    default: m = new Cavalo(); break;
    return m;
  }
}
```

```
Fazendeiro f = new Fazendeiro();
Mamifero mamifero = f.comprar(2);
Mamifero mamifero1 = f.comprar(3);
```

Compatibilidade entre Classes

```
Animal a;
Mamifero m;
Cao c;

a = m;
m = c;
a = c;
```

O conjunto dos animais inclui conjunto dos mamíferos, que inclui o conjunto dos cães.

Mamífero é compatível com **Animal**
Cao é compatível com **Mamífero**
Cao é compatível com **Animal**

```
Mamifero m;
Cao c = m;
```

Mamífero não é compatível com **Cao**

```
Mamifero m;
Cao c = (Cao) m;
```

Operação de *type cast* (conversão de tipo)

Verificação da Classe de um

referência **instanceof** *classe*

Podemos verificar se uma **classe** é do **tipo da referência**. O comando retorna verdadeiro quando o **objeto referenciado** é uma **instância da classe**.

```
public class Veterinario {
  public void examinar(Mamifero m) {
    if (m instanceof Cao) {
      Cao x = (Cao)m; x.latir();
    }
    else if (m instanceof Gato) {
      Gato y = (Gato)m; y.miar();
    }
    else if (m instanceof Cavalo) {
      Cavalo z = (Cavalo)m; z.relinchar();
    }
  }
}
```

Chamada de método polimórfica

***sobrescrita* (override)**

Podemos realizar uma **chamada de método polimórfica** quando o método é chamado a partir de uma **referência polimórfica** e quando o método é **sobrescrito** nas classe filha (subclasse).

```
public class Mamifero extends SangueQuente{
  public void soar() {
    System.out.println("Som de mamífero");
  }
}
```

```
public class Cao extends Mamifero {
  public void soar() {
    System.out.println("auau");
  }
}
```

```
public class Gato extends Mamifero {
  public void soar() {
    System.out.println("miau");
  }
}
```

```
public class Cavalo extends Mamifero {
  public void soar() {
    System.out.println("iiirrrrí");
  }
}
```

Chamada de método polimórfica

```
public class Veterinario {
  public void examinar(Mamifero m) {
    m.soar();
  }
}
```

Qual implementação do método *soar()* será chamada nesse caso?

Chamada de método polimórfica

Depende de qual classe está vinculada ao objeto m (Mamifero)

```
Veterinario v = new Veterinario();
Cao c = new Cao();
v.examinar(c);
Gato g = new Gato();
v.examinar(g);
```

Coleção de Objetos Polimórfica

Uma **coleção de objetos** é considerada **polimórfica** quando é composta por **referências polimórficas**, ou seja, o **tipo definido** para os objetos da lista é uma **referência** do **objeto pai**.

```
ArrayList<Mamifero> rebanho = new ArrayList<Mamifero>();

rebanho.add(new Cao());
rebanho.add(new Gato());
rebanho.add(new Cavalo());

for (Mamifero m: rebanho) {
  m.soar();
}
```