Programação Orientada a Objetos

Polimorfismo

Cristiano Lehrer, M.Sc.

ybadoo

Introdução (1/3)

- **Polimorfismo** significa ter muitas formas.
  - Em termos de programação, muitas formas significa que um único nome pode representar um código diferente, selecionado por algum mecanismo automático.
  - Assim, o polimorfismo permite que um único nome expresse muitos comportamentos diferentes.
  - Uma linguagem que suporta polimorfismo é uma linguagem polimórfica.
  - Em contraste, uma linguagem monomórfica não suporta polimorfismo e, em vez disso, restringe tudo a um e apenas um comportamento estático, pois cada nome é estaticamente vinculado ao seu código.

Introdução (2/3)

```
public class PersonalityObject {
  public String speak() {
    return "I am an object.";
  }
}

public class PessimisticObject extends PersonalityObject {
  public String speak() {
    return "The glass is half empty.";
  }
}

public class OptimisticObject extends PersonalityObject {
  public String speak() {
    return "The glass is half full.";
  }
}

public class IntrovertedObject extends PersonalityObject {
  public String speak() {
    return "Hi...";
  }
}
```


ybadoo

Introdução (3/3)

```
public static void main(String[] args) {
  PersonalityObject personalities = new PersonalityObject[4];
  personalities[0] = new PersonalityObject();
  personalities[1] = new PessimisticObject();
  personalities[2] = new OptimisticObject();
  personalities[3] = new IntrovertedObject();

  for(PersonalityObject personality: personalities) {
    System.out.println(personality.speak());
  }
}
```

```
I am an object.
The glass is half empty.
The glass is half full.
Hi...
```

Formas de Polimorfismo

- Infelizmente, ainda há pouco consenso na comunidade de Orientação a Objetos quando se trata de polimorfismo.
- Em vez de entrar na controvérsia, consideraremos quatro formas de polimorfismo:
  - Polimorfismo de inclusão
  - Polimorfismo paramétrico
  - Sobreposição
  - Sobrecarga

ybadoo

Polimorfismo de Inclusão (1/2)

- O polimorfismo de inclusão, às vezes chamado de polimorfismo puro, permite que se trate objetos relacionados genericamente.
  - O exemplo apresentado anteriormente é um polimorfismo de inclusão.
- O polimorfismo de inclusão é útil porque diminui a quantidade de código que precisa ser escrito.
  - Em vez de ter de escrever um método para cada tipo concreto de `PersonalityObject`, pode-se simplesmente escrever um método que manipule todos os tipos.

ybadoo

Polimorfismo de Inclusão (2/2)

- O polimorfismo é o motivo pelo qual não se deve associar automaticamente herança com reutilização de implementação.
  - Em vez disso, deve se usar herança principalmente para permitir um comportamento polimórfico através de relacionamentos com capacidade de substituição.
  - Se definir corretamente os relacionamentos com capacidade de substituição, a reutilização será automática.
  - O polimorfismo de inclusão permite que se reutilize a classe base, qualquer descendente, assim como os métodos que usam a classe base.

ybadoo

Polimorfismo Paramétrico (1/2)

- O polimorfismo paramétrico permite que se crie métodos e tipos genéricos.
  - Assim como o polimorfismo de inclusão, os métodos e tipos genéricos permitem que se codifique algo uma vez e faça isso trabalhar com muitos tipos diferentes de argumentos.

ybadoo

Polimorfismo Paramétrico (2/2)

```
public class Operation<T> {

  public T add(T a, T b) { return a + b; }

  public static void main(String[] args {
    Operation<Integer> op1 = new Operation<Integer>();
    System.out.println("Integer: " +
                       op1.add(new Integer(10), new Integer(20));

    Operation<Double> op2 = new Operation<Double>();
    System.out.println("Double: " +
                       op2.add(new Double(5.5), new Double(7.3));
  }
}
```

```
Integer: 30
Double: 12.8
```

Sobreposição

```
public abstract class MoodyObject {

  protected abstract String getMood();

  public void queryMood() {
    System.out.println("I feel " + getMood() + " today!");
  }
}

public class HappyObject extends MoodyObject {

  protected String getMood() {
    return "happy";
  }

  public void laugh() {
    System.out.println("hehehe... hahaha... HAHAHA!!!");
  }
}
```


ybadoo

Sobrecarga

- A sobrecarga, também conhecida como polimorfismo *ad-hoc*, permite que se use o mesmo nome de método para muitos métodos diferentes.
  - Cada método difere apenas no número e no tipo de seus parâmetros.

```
public static int max(int a, int b);

public static long max(long a, long b);

public static float max(float a, float b);

public static double max(double a, double b);
```


ybadoo