Aula 5

Programação Orientada a Objetos

Prof. Leonardo Gomes

1

Conversa Inicial

2

- Polimorfismo
- Polimorfismo na linguagem Java
- Classe abstrata
- Interface
- Enum

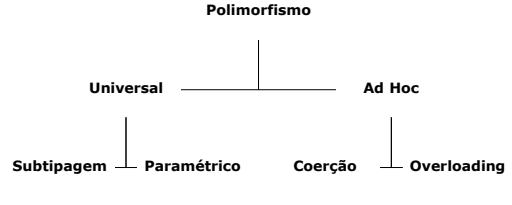
3

Polimorfismo

4

- Grego
  - *Polýs* = muitas
  - *Morphé* = formas
- Tipo mais conhecido de polimorfismo
  - Capacidade de uma mesma referência de superclasse assumir métodos e atributos de diferentes subclasses

5



6

Subtipagem

- Subclasses reimplementam métodos com outro comportamento
- Subclasses herdam o tipo da superclasse também

7

Paramétrico

- O tipo de dado é descrito de forma genérica
- Permite que um mesmo código seja aplicado em diferentes tipos
- Template (C++)
- Generics (Java e C#)

8

Coerção

- Aplicação de conversão de tipos
- Ocorre de forma implícita ou explícita
- Facilita a leitura de códigos, mas pode ser fonte de erros de difícil detecção

9

Overloading

- Funções com mesmo nome
- Parâmetros diferentes
- Comportamentos diferentes

10

Polimorfismo na linguagem Java

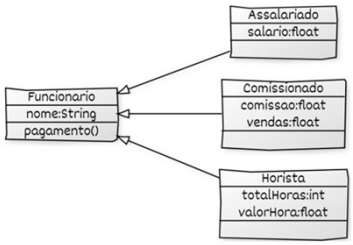
11

Referência

```
Aluno mario = new Aluno();
Aluno luigi = mario;
Aluno yoshi = mario;

mario.nome = "super mario";
System.out.println(mario.nome); //imprimirá  "Super Mario"
System.out.println(luigi.nome);//imprimirá  "Super Mario"
System.out.println(yoshi.nome);//imprimirá  "Super Mario"
luigi.nome = "Super luigi";
System.out.println(mario.nome); //imprimirá  "Super Luigi"
System.out.println(luigi.nome); //imprimirá  "Super Luigi"
System.out.println(yoshi.nome); //imprimirá  "Super Luigi"
```

12

Exemplo funcionário



13

Exemplo funcionário

```
Funcionario f;

f = new Horista("Mario",100,40.5f);
System.out.println("Horista: " +f.pagamento());
f = new Comissionado("Luigi",50000,0.05f);
System.out.println("Comissionado: " +f.pagamento());
f = new Assalariado("Yoshi",3500);
System.out.println("Assalariado: " +f.pagamento());
```

14

Classe abstrata

15

Classe abstrata

- **Classe que não desejamos instanciar**
- **Permite métodos abstratos que não desejamos implementar**
- **Útil para utilização como superclasse**

16

Exemplo classe forma

```
public abstract class FormaGeometrica {
   public double area;
   public abstract void calculaArea();
}

class Quadrado extends FormaGeometrica{
   public void calculaArea(){
         Scanner teclado = new Scanner(System.in);
         System.out.println("Digite a medida do lado");
         double lado = teclado.nextDouble();
         area = lado*lado;
   }
}
```

17

Exemplo classe forma

```
public static void main(String[] args) {
        FormaGeometrica forma;

        //Forma é um círculo
        forma = new Circulo();
        forma.calculaArea();
        System.out.println(forma.area);

        //Forma agora é um quadrado
        forma = new Quadrado();
        forma.calculaArea();
        System.out.println(forma.area);
    }
```

18

Interface

19

- Conceito semelhante ao de classe abstrata
- No Java não é possível uma mesma classe ter múltiplas superclasses
- Interface é o meio de ligação entre dois sistemas
- Conjunto de assinaturas

20

Exemplo de interface

- HDMI
  - Uma interface
  - Conecta dispositivos eletrônicos e telas
  - Funciona com diferentes modelos

21

Interface animal

```
interface Animal {
  public void emitirSom(); //método sem corpo
  public void dormir(); // método sem corpo
}
class Gato implements Animal {
  public void emitirSom() {
    System.out.println("O gato fala: miau miau");
  }
  public void dormir() {
    System.out.println("Zzz");
  }
}

class Principal {
  public static void main(String[] args) {
    Gato tom = new Gato();
    tom.emitirSom();
    tom.dormir();
    Animal obj = tom;
    obj.emitirSom();
    obj.dormir();
  }
}
```

22

Diferenças

Propriedade	Interface	Classe abstrata
Herança	Classes podem implementar diversas interfaces	Classes só podem herdar uma única superclasse
Métodos	Interface só possui a assinatura dos métodos	Uma classe abstrata pode implementar códigos dentro de seus métodos, que serão ou não sobrescritos
Atributos	Só pode possuir atributos estáticos	Pode ter tantos atributos estáticos quanto não-estáticos.
Adaptação	É fácil adaptar uma classe existente para implementar uma interface, basta implementar os métodos conforme a assinatura	Adaptar uma classe existente para herdar uma classe abstrata pode ser trabalhoso por ser necessário modificar a hierarquia já existente de heranças
Quando usar?	Classes que compartilham mesmo comportamento, assinatura	Classes que compartilham mesmos atributos e precisam ter seu estado avaliado de forma compatível
Modificações adicionais	Ao adicionar um novo método em uma interface, todas as classes devem trazer suas implementações	Ao adicionar um novo método, é possível trazer uma implementação padrão, que servirá para todas as classes filhas

23

Enum

24

- **Classe especial**
- **Simples**
- **Capaz de representar constantes enumeradas**
- **Conceito existente em diversas linguagens**

25

Exemplo estação

```
enum Estacao {
  VERAO,
  OUTONO,
  INVERNO,
 PRIMAVERA
}
```

26

Exemplo estação

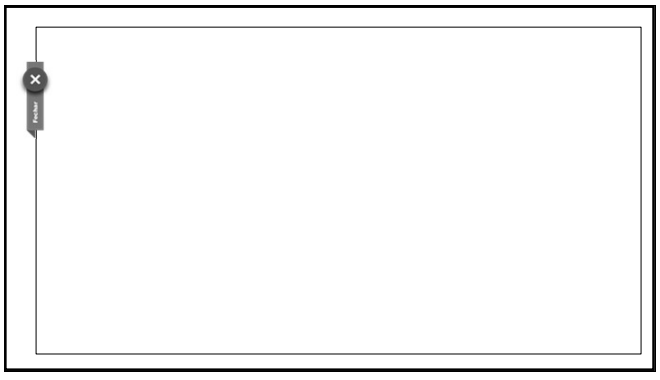
```
public class Principal{
  public static void main(String[] args) {
    Estacao estacaoRoupa = Estacao.INVERNO;

    switch(estacaoRoupa) {
      case VERAO:
        System.out.println("Arrase na praia");
        break;
      case OUTONO:
         System.out.println("Passe o outono com elegância");
        break;
      case INVERNO:
        System.out.println("Se agasalhe bem e com estilo");
        break;
      case PRIMAVERA:
        System.out.println("Se vista bem na estação das
flores");
        break;
    }
  }
}
```

27

Exemplo estação

```
for (Estacao est : Estacao.values()) {
  System.out.println(est);
}
```

28



29