Introdução à Linguagem Java

Lista 05

1 de abril de 2016

1. Verifique a que a construção automática de classes derivadas funciona, através do seguinte exemplo:

```
class Base {
    Base() {
        System.out.println("Constrói Base");
    }
}
public class Derivada extends Base {
    Derivada () {
        System.out.println("Constrói Derivada");
    }
    public static void main(String []argc){
       Derivada obj = new Derivada();
    }
}
```

   Tente prever o que acontece se adicionamos um parâmetro inteiro ao construtor da classe Base ? Veja se a mensagem de erro corresponde ao que você pensou. Corrija este erro sem alterar a classe Base.

2. Crie uma classe com um membro static final e um membro final e demonstre com código a diferença entre os dois.

3. Crie uma classe final e tente criar uma classe derivada. Qual é a mensagem de erro ?

4. Mostre que quando um objeto final é recebido como argumento não se pode alterar o seu ponteiro. O que acontece se este objeto é alterado para null ?

5. Altere o seguinte programa para verificar as várias possibilidades de polimorfismo. O que acontece se um novo método `sangueQuente()` é adicionado a Mamífero, pode se chamá-lo com um objeto da classe Animal?

```
class Animal {
        void nasce() {
                System.out.println("Nasceu um animal");
        }
        void cresce() {
                System.out.println("Cresceu um animal");
        }
}
class Mamifero extends Animal {
        void nasce(int i)  {
                System.out.println("Nasceu um mamifero: "+i);
        }
        void cresce() {
                System.out.println("Cresceu um mamifero");
        }
        public static void main(String []argc) {
                Animal x = new Animal();
                x.nasce();
                Mamifero m = new Mamifero();
                m.nasce(1);
```

```
                x = m; // OK, pois Mamifero deriva de Animal
                x.nasce();
                // x.nasce(1); e esta linha ?
        }
}
```

6. Verifique a diferença entre Overriding e Overloading através do seguinte exemplo:

```
class Animal {
        void nasce() {
                System.out.println("Nasceu um animal");
        }
        void cresce() {
                System.out.println("Cresceu um animal");
        }
}
class Mamifero extends Animal {
        void nasce()  { // coloque um parametro aqui por exemplo...
                System.out.println("Nasceu um mamifero");
        }
        void cresce() {
                System.out.println("Cresceu um mamifero");
        }
}
public class Homem extends Mamifero  {
        void nasce()  {
                System.out.println("Nasceu um homem");
        }
        void cresce()  {
                System.out.println("Cresceu um homem");
        }
        public static void main(String []argc) {
                Animal x = new Animal();
                x.nasce();
                Homem h = new Homem();
                h.nasce();
                x = h; // OK, pois Homem deriva de Animal
                x.nasce();
        }
}
```

7. No programa abaixo, verifique que os métodos não abstratos das classes abstratas estão disponíveis nas classes derivadas. Tente criar objetos da classe Veiculos, qual a mensagem de erro ?

```
abstract class Veiculo  {
        private double preco;
        public void set(double valor) {
                preco = valor;
        }
        public double get() {
                return preco;
        }
        abstract public void move();
        abstract public void tamanho();
}

class Carro extends Veiculo {
        public void move() {
                System.out.println("Por vias pavimentadas");
        }
        public void tamanho() {
                System.out.println("Entre 2 e 4 metros");
        }
}

class Aviao extends Veiculo  {
        public void move() {
                System.out.println("Pelo ar");
        }
```

```
        public void tamanho() {
                System.out.println("Entre 4 e 200 metros");
        }
}
class TecoTeco extends Aviao  {
        public void move() {
                System.out.println("Pelo ar, mas baixo");
        }
        public void tamanho() {
                System.out.println("Entre 4 e 6 metros");
        }
}
public class TestaVeiculos {
        public static void main(String [] argc) {
                Veiculo []v = new Veiculo[5]; //vetor com 5 veiculos
                v[0]=new Carro();
                v[1]=new Aviao();
                v[2]=new TecoTeco();
                v[3]=new Aviao();
                v[4]=new TecoTeco();
                for(int i=0;i<5;i++) {
                        System.out.print("O veiculo "+i+" se move: ");
                        v[i].move();
                }
        }
}
```

8. Crie uma classe base com dois métodos [1]. No primeiro método chame o segundo. Crie uma classe derivada que altera o segundo método. Crie um objeto da classe derivada, transforme-o no seu tipo base, chame o primeiro método e verifique o que acontece. Explique.

9. Crie uma classe abstrata sem métodos. Derive uma classe e adicione um método. Crie um método estático que tem como argumento uma referência a classe base, efetua o downcast a classe derivada, e chama o método. No método `main()`, mostre que isto funciona. Agora, coloque uma definição abstrata do método na classe base. O downcast continua sendo necessário?

10. Em várias linguagens orientadas a objeto é possível fazer heranças múltiplas, isto é, fazer com que uma classe herde características de duas ou mais classes base. Verifique se isto é possível em Java.

11. Crie um programa com pelo menos três classes onde cada uma delas contém o método `public void main(String [] args)`. Você consegue compilar o programa ? Explique. Existe algum interesse em ter um método main associado a cada classe ?

[1] Do ponto de vista didático, este métodos devem apenas imprimir uma mensagem, de forma a sinalizar a sua chamada. Note que raramente imprime-se mensagens em métodos !!