Nona Lista de Exercícios
Herança e Especificadores de Acesso

Norton Trevisan Roman

1 de junho de 2011

1. Considere as seguintes classes:

```
class Ponto {
    double x,y;

    public void limpa() {
        this.x = 0;
        this.y = 0;
    }
}

class Pixel extends Ponto {
    Color cor;
}
```

   Qual a diferença entre elas, em termos do que elas fazem?

2. Considere as seguintes classes:

```
public class A {
    int x;

    public A(int x) { this.x = x;}
}

public class B extends A {
    float y;

    public B(float y) { this.y = y;}
}
```

   Elas irão compilar? Por que? O que você pode fazer para consertar isso?

3. Dadas seguintes classes:

```
package p1;
public class Pessoa {
    public static int contador = 0;
```

```
    public Pessoa(){
        contador++;
    }

    protected void comer(){
        System.out.println("Pessoa  comendo.");
    }

    void caminhar(){
        System.out.println("Pessoa  caminhando.");
    }

    private void correr(){
        System.out.println("Pessoa correndo.");
    }
}

package p1;
public class Aluno extends Pessoa {
    public void estudar(){
        System.out.println("Aluno estudando.");
    }

    protected void comer(){
        System.out.println("Aluno comendo.");
    }

    private void correr(){
        System.out.println("Aluno correndo.");
    }
}
```

(a) Assinale com um X quais linhas do programa abaixo não funcionarão:

```
   package p1;
   public class ExecutarPessoas {
       public static void main(String[] args) {
           Pessoa p1;
           Pessoa p2;
           Aluno a1;
           Aluno a2;
[ ]        p1 = new Pessoa();
[ ]        p2 = new Aluno();
[ ]        a1 = new Aluno();
[ ]        System.out.println("Contador A1: " + a1.contador);
[ ]        System.out.println("Contador Pessoa: " + Pessoa.contador);
[ ]        a2 = new Pessoa();
[ ]        p2.estudar();
[ ]        p2.comer();
[ ]        p2.caminhar();
[ ]        p2.correr();
```

```
        }
    }
```

(b) Após excluir as linhas que não funcionam, escreva abaixo o que será impresso como resultado da execução desse programa?

4. Em uma farmácia de manipulação, um remédio é composto por várias substâncias. O preço de um remédio é calculado pela soma ponderada dos preços das substâncias que o compõem. Substâncias nobres têm o peso 2 e simples têm o peso 1. Por exemplo: o remédio "Veneno de rato" é composto por 3 quantidades de álcool (simples, com preço = 1) e 5 quantidades de potássio (nobre, com preço = 3). O preço total da unidade deste remédio é 3*1*1 + 5*3*2 = 33. Supondo que já foi implementada a classe Substancia em Java (dada abaixo):

```
class Substancia {
    String _nome;
    int _tipo; // 1=simples    2=nobre
    double _preco;

    void defineNome(String pnome) { nome = pnome; }
    String obtemNome () { return nome; }
    void defineTipo(char ptipo) { tipo = ptipo; }
    int obtemTipo () { return tipo; }
    void definePreco(int ppreco) { preco = ppreco; }
    double obtemPreco () { return preco; }
}
```

(a) Altere a classe Substancia para que os atributos não tenham acesso direto, mas garanta acesso público a eles por meio dos métodos de acesso (getters e setters).

(b) Acrescente um atributo na classe Substancia a fim de armazenar quantos objetos foram criados nesta classe.

(c) Acrescente um atributo na classe Substancia que informe a cotação da moeda corrente. Este atributo não poderá mais ser alterado após sua criação (especificador *final*).

(d) Acrescente um método na classe Substancia para mostrar todos os atributos, incluindo os atributo criados nos itens anteriores.

(e) Implemente uma classe Remedio com métodos para:

- definir e obter o nome do remédio;
- adicionar substâncias que fazem parte de sua composição e calcular o preço final do remédio. Deve ser armazenado somente o preço final do remédio. Não é necessário armazenar as substâncias em vetores. Você usará apenas um atributo para o armazenamento. Deve ser considerado o valor da cotação estabelecido na classe Substancia.
- imprimir o preço final do remédio, no seguinte formato: "Preço de uma unidade do remédio NOME_REMEDIO = PRECO". Por exemplo, se o valor da cotação for igual a 1, a execução do exemplo anterior imprimirá: "Preço de uma unidade do remédio Veneno de rato = 33.0".

5. Considere a seguinte classe:

```
package especificadores1;
public class Alpha {
    public static void alphaPublic(){
        System.out.println("Alpha public!");
    }
    static void alphaNada(){
        System.out.println("Alpha nada!");
    }
    protected static void alphaProtected(){
        System.out.println("Alpha protected!");
    }
    private static void alphaPrivate(){
        System.out.println("Alpha private!");
    }
    public static void main(String[]args) {
        alphaPublic();
        alphaProtected();
        alphaNada();
        alphaPrivate();
    }
}
```

(a) O que será impresso na tela?

(b) A seguinte subclasse irá rodar? Por que?

```
package especificadores1;
public class AlphaSub1 extends Alpha{
    public static void main(String[]args) {
        Alpha.alphaPublic();
        Alpha.alphaProtected();
        Alpha.alphaNada();
        Alpha.alphaPrivate();
    }
}
```

(c) E essa, vai rodar? Por que?

```
package especificadores1;
public class Beta {
    public static void main(String[]args) {
        Alpha.alphaPublic();
        Alpha.alphaProtected();
        Alpha.alphaNada();
        Alpha.alphaPrivate();
    }
}
```

(d) E essa, vai rodar? Por que?

```
package especificadores2;
import especificadores1.Alpha;
public class AlphaSub2 extends especificadores1.Alpha{
    public static void main(String[]args) {
```

```
        Alpha.alphaPublic();
        Alpha.alphaProtected();
        Alpha.alphaNada();
        Alpha.alphaPrivate();
    }
}
```

(e) E essa, vai rodar? Por que?

```
package especificadores2;
import especificadores1.Alpha;
public class Gamma{
    public static void main(String[]args) {
        Alpha.alphaPublic();
        Alpha.alphaProtected();
        Alpha.alphaNada();
        Alpha.alphaPrivate();
    }
}
```

6. Como fazer para criar uma classe na qual só seja possível construir um único objeto. Este tipo de objeto é chamado de Singleton. Dica: Limitar o acesso ao construtor.