UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

ATP1 de Programação Orientada a Objetos

27 de março de 2017

Duração: 1h15

Nome_____ Nº mec._____

I. [6] Relativamente às perguntas 1 a 12, assinale na tabela seguinte um X na coluna "V" para as declarações que estão corretas e na "F" para as que estão incorretas. Note que estas questões têm por base a linguagem Java. Cada uma destas perguntas vale 0.5 valores e cada resposta errada desconta 0.25 valores. Questões não respondidas valem 0.

	V	F
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

1. Os métodos definidos como `final` podem ser reescritos em classes derivadas.
2. Todos os métodos abstratos têm de ser reescritos (`overriding`) numa subclasse se quisermos instanciar objetos da subclasse.
3. Os atributos de visibilidade (`private, protected, public, package`) permitem implementar o encapsulamento das classes.
4. O compilador cria automaticamente um construtor por omissão, sem argumentos, mesmo sendo definido outro construtor na classe.
5. A expressão `c1 != c2` verifica se os objetos referenciados por `c1` e por `c2` são diferentes do ponto de vista dos dados de cada objeto.
6. A maior quantidade representável com uma variável do tipo `byte` é 256.
7. Não é possível invocar um método estático sem que existam objetos dessa classe.
8. Podemos reduzir a visibilidade de métodos herdados numa classe derivada.
9. Os objetos da classe `String` podem ser alterados.
10. Considere que a classe `Pepino` é derivada de `Vegetal`. Uma referência do tipo `Vegetal` pode apontar para um objeto do tipo `Pepino`.
11. Vários objetos de uma classe em Java têm o mesmo comportamento mas podem guardar dados diferentes.
12. As variáveis estáticas, ou variáveis de classe, são comuns a todos os objetos dessa classe.

II. [8] Considere os programas seguintes e indique o que é impresso no terminal (use as linhas vazias que se seguem aos programas). Não existem espaços no conteúdo a ser impresso.

```
class MyClass{
   public final static String S = "PI";
   private double x;
   private static int y = 100;
   public MyClass(double x){
      System.out.println("Obj=" + y);
      this.x = x;
      y = y + 5;
   }
   public boolean equals(Object o){
      MyClass tmp = (MyClass)o;
      if(Math.round(this.x) !=
         Math.round(tmp.x)){
              return false;
      }
      return true;
   }
   public String toString(){
      return "x = " + this.x;
   }
}
```

```
public class XPTO {
    public static void main(String[] a){
       System.out.println(MyClass.S);
       MyClass obj1 = new MyClass(5.5);
       MyClass obj2 = new MyClass(5.2);
       if(obj1.equals(obj2))
              System.out.println(obj1);
       else
              System.out.println(obj2);
    }
}
```

Resultado da execução do programa:

_____

_____

_____

_____

```
public class XPTO {
   public static void main(String[] args) {
       String x = "dia,1,mes,12,ano,2016";
       int n = 0, p;
       String y[] = x.split(",");
       System.out.println(y.length);
       for(p = 0 ; p < x.length() ; p++){
           if(Character.isLetter(x.charAt(p))){
              n++;
           }
       }
       System.out.println("p=" + p);
       System.out.println("n=" + n);
       for(String s : y){
           System.out.print(s);
       }
       System.out.println();
   }
}
```

Resultado da execução do programa:

_____

_____

_____

_____

III. [3] Considere as classes declaradas abaixo e o programa de teste. Tenha em atenção o que foi impresso depois da execução do programa e inclua o código necessário nos espaços em branco e no corpo dos métodos vazios para que seja possível a sua execução. Não precisa de acrescentar mais métodos nem atributos.

```
class OneClass{
    private String s;
    public OneClass(String s){
        this.s = s;
    }
    public _____________________ getS(){
    // colocar código aqui




    }
}

class OtherClass _____________________________________{
    private int x;
    public OtherClass(String s, int x){
   // colocar código aqui




    }
    public _____________________ getInfo(){
    // colocar código aqui




    }
}

public class ATP3 {
    public static void main(String[] args) {
        OneClass obj1 = new OneClass("POO");
        OtherClass obj2 = new OtherClass("JAVA", 10);
        OneClass obj3 = new OtherClass("CLASSES", 20);
        System.out.println(obj1.getS());
        System.out.println(obj2.getInfo());
        System.out.println(obj3.getS());
    }
}
```

Resultado da execução do programa:

```
text=POO
text=JAVA / number=10
text=CLASSES
```

III. [3] Considere as seguintes entidades e caraterísticas:

a. **Vegetal**, que pode ser Couve, Alface, Tomate, todos caracterizados por um nome (String), e uma data de sementeira (Data).
b. **Leguminosa**, que pode ser Feijão ou Ervilha, tem nome (String), pode ser colhido seco (Boolean) e tem data de sementeira (Data).
c. **Suspenso**, são alguns tipos de vegetais e leguminosas que precisam de suporte para subir (e.g., `void setSuporte(String s)`). Considere como tipo Suspenso as entidades Tomate e Feijão.
d. **Horta**, contém um conjunto dos elementos anteriores, sejam vegetais ou leguminosas e um nome (String).
e. **HortaComSuportes**, uma horta constituída exclusivamente por produtos suspensos.

Construa um diagrama com todas as entidades e associações, usando elementos gráficos como os apresentados ao lado. Nota: Não é necessário incluir atributos nem métodos nem usar todos os elementos apresentados.