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 atributos de visibilidade (`private`, `protected`, `public`, `package`) permitem implementar o encapsulamento das classes.
2. Um método abstrato tem de ser reescrito (`overriding`) numa subclasse, mesmo que esta subclasse seja declarada como `abstract`.
3. Os métodos definidos como `final` não podem ser reescritos em classes derivadas.
4. O compilador cria automaticamente um construtor por omissão, sem argumentos, caso não seja definido qualquer construtor na classe.
5. Os objetos da classe `String` são imutáveis, ou seja, não podem ser alterados.
6. As variáveis estáticas, ou variáveis de classe, são comuns a todos os objetos dessa classe.
7. A menor quantidade representável com uma variável do tipo `byte` é -128.
8. É possível invocar um método estático sem que existam objetos dessa classe.
9. Podemos reduzir a visibilidade de métodos herdados numa classe derivada.
10. A expressão `c1 == c2` verifica se os objetos referenciados por `c1` e por `c2` são iguais do ponto de vista dos dados de cada objeto.
11. Considere que a classe `Pepino` é derivada de `Vegetal`. Uma referência do tipo `Pepino` pode apontar para um objeto do tipo `Vegetal`.
12. Vários objetos de uma classe em Java têm o mesmo comportamento mas podem guardar dados diferentes.

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.

```
public class XPTO {
  public static void main(String[] args) {
      String x = "1-de-janeiro-2017";
      int n = 0, p;
      for(p = 0 ; p < x.length() ; p++){
          if(Character.isDigit(x.charAt(p))){
             n++;
          }
      }
      System.out.println("n=" + n);
      System.out.println("p=" + p);
      String y[] = x.split("-");
      System.out.println(y.length);
      for(String s : y){
          System.out.print(s);
      }
      System.out.println();
  }
}
```

Resultado da execução do programa:

_____

_____

_____

_____

```
class MyClass{
  private int x;
  private static int y = 100;
  public final static int K = 10;
  public MyClass(int x){
     this.x = x;
     System.out.println("Novo=" + y);
     y = y + 10;
  }
  public boolean equals(Object o){
     MyClass tmp = (MyClass)o;
     if(this.x != tmp.x){
         return false;
     }
     return true;
  }
  public String toString(){
     return "x = " + this.x;
  }
}
```

```
public class XPTO {
   public static void main(String[] a){
      MyClass obj1 = new MyClass(5);
      MyClass obj2 = new MyClass(10);
      if(obj1.equals(obj2)){
             System.out.println(obj1);
      }
      else{
             System.out.println(obj2);
      }
      System.out.println(MyClass.K);
     }
}
```

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 int s;
    public OneClass(int s){
        this.s = s;
    }
    public ____________________ getS(){
    // colocar código aqui




    }
}

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




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




    }
}

public class ATP3 {
    public static void main(String[] args) {
        OneClass obj1 = new OneClass(5);
        OtherClass obj2 = new OtherClass(10, 20);
        OneClass obj3 = new OtherClass(100, 50);
        System.out.printf("info=%d\n", obj1.getS());
        System.out.printf("%s\n", obj2.getInfo());
        System.out.printf("info=%d\n",obj3.getS());
    }
}
```

Resultado da execução do programa:

```
info=5
AllInfo:1020
info=100
```

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

a. **Arvore**, que pode ser Pinheiro, Eucalipto, todos caracterizados por um nome (String), e uma data de plantação (Data).
b. **ArvoreDeFruto**, que pode ser Macieira, Oliveira ou Pereira tem nome (String), data de plantação (Data) e cor do fruto (String).
c. **Latada**, são alguns tipos de árvores que podem crescer em latada contra uma parede ou sobre arames estendidos (e.g., `void setSuporte(String s)`). Considere como tipo Latada as entidades Macieira e Pereira.
d. **Quinta**, contém um conjunto dos elementos anteriores, sejam Arvore ou ArvoreDeFruto e um nome (String).
e. **QuintaComLatada**, uma quinta constituída exclusivamente por árvores em latada.

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.