UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

ATP2 de Programação Orientada a Objetos

29 de maio de 2017

Duração: 1h15

Nome_____ Nº mec._____

I. [6] Relativamente às perguntas 1 a 12, assinale na tabela seguinte com um X na coluna "V" 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. Uma classe parametrizada (i.e. `Stack<T>`) pode ser instanciada com um tipo primitivo (i.e. `Stack<int>`).
2. Em Java é permitido que uma classe implemente múltiplas interfaces.
3. Os enumerados em Java são classes que podem ter dados e operações associadas.
4. Só podemos usar tipos genéricos em classes, nunca em interfaces ou métodos.
5. Não é possível criar um array do tipo genérico `T`, como por exemplo `T[] array = new T[10];`
6. Coleções em Java (Java Collections Framework) são um conjunto de classes, interfaces e algoritmos que representam e manipulam várias estruturas de dados.
7. As listas (`List`) não permitem guardar elementos repetidos enquanto que os conjuntos (`Set`) sim.
8. A classe `java.io.File` permite ler ou escrever dados em ficheiros.
9. As classes internas podem ser classificadas como: estáticas, de instância, locais e anónimas.
10. A geração e tratamento de exceções permite controlar situações imprevistas durante a execução do programa.
11. Uma asserção em Java deve ser usada para testar condições que nunca devem falhar num programa correto.
12. Se a classe `ABC` implementar a interface `XYZ`, não podemos usar um objeto do tipo `ABC` em situações em que se espera um objeto do tipo `XYZ`.

II. [3] Reescreva a classe seguinte utilizando genéricos, de forma a permitir criar pontos de qualquer tipo. Por exemplo, `new Point<String>` ou `new Point<Integer>`.

```
public class Point {
      public double x;
      public double y;
      public Point(double x, double y) {
            this.x = x;
            this.y = y;
      }
      public double getFirst()  { return x;  }
      public double getSecond() { return y; }
      public void setFirst(double x)  { this.x = x  }
      public void setSecond(double y) { this.y = y; }
      public String toString() { return "x = " + x + "y = " + y};
}
```

III. [3] Com base nas classes/interfaces de Java Collections defina objetos adequados para representar:

a) Os números das camisolas dos jogadores de futebol de um clube que participaram num determinado jogo.

a) O número de presenças de alunos de uma turma, ao longo das várias aulas.

c) A quantidade de faltas dadas por cada aluno (nome) de uma disciplina.

d) O número de docentes por departamento (dep) e por universidade de Portugal. Esta estrutura deverá permitir, por exemplo, listar todos os dep. de uma universidade, ou o número de docentes por dep. de cada universidade.

IV. [4] Considere o programa seguinte 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 enum Cores {
ORANGE, GREEN, WHITE
}

public class Resultado1 {
  public static void main(String[] a)
{
      AlgoMais a = new Coisa(5);
      System.out.println(a);
      System.out.println(
         a.maisUmaCoisa());

      Coisa b = new Coisa(10);
      b.f();
      Coisa.Aquilo c =
        b.new Aquilo(Cores.GREEN);

      System.out.println(c);
  }
}

interface AlgoMais{
  public String maisUmaCoisa();
}

class Coisa implements AlgoMais{
  private int x;
  public Coisa(int x){
      this.x = x;
  }


  public String maisUmaCoisa(){
      return "texto";
  }
```

```
  public String toString() {
      return "x=" + x;
  }

  public class Aquilo{
      private Cores z;
      Aquilo(Cores z){
          this.z = z;
      }

      public String toString() {
          String x = "";
          Cores a[] = Cores.values();
          for(Cores i : a)
             x += i + ",";
          return "z=" + z + "," + x;
      }
  }

  public void f(){
      class Outra{
          private int k;
          public Outra(int k){
             this.k = k;
          }

          public String toString() {
             return "Outra k=" + k;
          }

      }

      Outra o = new Outra(25);
      System.out.println(o);
  }
}// fim da class Coisa
```

Resultado da execução do programa:

_____

_____

_____

_____

V. [4] Considere o programa seguinte e o ficheiro "`f.txt`". Tenha em atenção o que foi impresso depois da execução do programa e inclua o código necessário nos espaços em branco para que seja possível obter o resultado impresso ao executar o programa.

```
public static void main(String[] args) {
          File f = new File("f.txt");
          Scanner scf = null;

          ___________________ {
             scf = new Scanner(f);

          } ____________________ (FileNotFoundException e) {
             System.out.println("Pois!"); System.exit(0);

          } ___________________ {
             System.out.println("Ola!");
          }

          ___________________________________ x = new ____________________();
          int p = 0;

          while(scf._____________()){

             String s = scf._____________();

             int n = scf._____________________();

             if(x._____________________(s)){
                   System.out.println("Nice!");
                   p = x._____________________(s);
                   p += n;
             }
             else{
                   p = 1;
              }

             x._____________________(s, p);
          }

          System.out.println("s=" + x._____________()());

          System.out.println(x);
          scf.close();
      }
```

Conteúdo do ficheiro "`f.txt`":

```
ola 1
aveiro 2
poo 2
ola 3
poo 4
aveiro 4
poo 4
```

Resultado da execução do programa:

```
Ola!
Nice!
Nice!
Nice!
Nice!
s=3
{poo=9, aveiro=5, ola=4}
```