



Prova 2016, questões

Programação Orientada a Objetos (Universidade de Aveiro)

UNIVERSIDADE DE AVEIRO
DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA
ATP2 de Programação Orientada a Objetos
29 de maio de 2017
Duração: 1h15

Nome _____ N.º 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. Se a classe ABC implementar a interface XYZ, então podemos usar um objeto do tipo ABC em situações que se espera um objeto do tipo XYZ.
2. Não é permitido que uma classe implemente múltiplas interfaces.
3. As classes internas podem ser classificadas como: estáticas, de instância, locais e anónimas.
4. Os enumerados são classes que não podem ter dados e operações associadas.
5. Uma classe parametrizada (i.e. `Stack<T>`) pode ser instanciada com um tipo primitivo (i.e. `Stack<int>`).
6. Podemos usar tipos genéricos em classes, interfaces e métodos.
7. É possível criar um array do tipo genérico T, como por exemplo `T[] array = new T[10];`
8. Coleções (Java Collections Framework) são um conjunto de classes, interfaces e algoritmos que representam e manipulam várias estruturas de dados.
9. As listas (`List`) permitem guardar elementos repetidos enquanto que os conjuntos (`Set`) não.
10. A classe `java.io.File` permite consultar informação e realizar operações sobre ficheiros e pastas.
11. A geração e tratamento de exceções permite controlar situações imprevistas durante o correr do programa.
12. Uma asserção é usada para controlar o fluxo de execução de um programa.

II. [3] Reescreva a classe seguinte utilizando genéricos, de forma a permitir criar pares de qualquer combinação de tipos, por exemplo, `new Pair<String, String>` ou `new Pair<Integer, Pessoa>`.

```
public class Pair {
    public Object first;
    public Object second;
    public Pair(Object x, Object y) {
        this.first = x;
        this.second = y;
    }
    public Object getFirst() { return first; }
    public Object getSecond() { return second; }
    public void setFirst(Object x) { this.first = x }
    public void setSecond(Object y) { this.second = y; }
}
```

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

- a) Os valores de temperatura (reais) capturados por uma estação meteorológica ao longo do tempo.
- b) Os números das camisolas dos jogadores de futebol da Seleção que participaram num determinado jogo.
- c) A quantidade de vezes que cada palavra ocorre num documento.
- d) O número de docentes por escola e por concelho de Portugal. Esta estrutura deverá permitir, por exemplo, listar todas as escolas de um concelho, ou o número de docentes por cada escola de cada concelho.

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 {
    GREEN, BLUE, RED
}

public class Resultado1 {
    public static void main(String[] a)
    {
        AlgoMais a = new Coisa(10);
        System.out.println(a);
        System.out.println(
            a.maisUmaCoisa());

        Coisa b = new Coisa(20);
        b.f();
        Coisa.Aquilo c =
            b.new Aquilo(Cores.BLUE);

        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(50);
        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("LOL");
    }

    _____ x = new _____ ();
    int p = 0;

    while(scf. _____ ()) {

        String s = scf. _____ ();

        int n = scf. _____ ();

        if(x. _____ (s)) {
            System.out.println("Cool!");
            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”:

```
aaa 1
bbb 2
bbb 2
ccc 3
ddd 4
ddd 4
aaa 4
```

Resultado da execução do programa:

```
LOL
Cool!
Cool!
Cool!
s=4
{aaa=5, ccc=1, bbb=3, ddd=5}
```