Aula Prática 13

Objetivos

Resolução de problemas gerais de programação.

Problema 13.1

Considere as seguintes entidades:

- País: caracterizado por um nome (String), uma capital (Localidade) e por um conjunto pré-definido de regiões (Região).
- Região: caracterizada por um nome (String) e uma população (int)
- Estado: caracterizada por um nome (String), uma população (int) e uma capital (Localidade do tipo TipoLocalidade.CIDADE)
- Província: caracterizada por um nome (String), uma população (int) e um governador (String)
- Localidade: caracterizada por um nome (String), uma população (int) e um tipo (TipoLocalidade.CIDADE, TipoLocalidade.VILA, TipoLocalidade.ALDEIA)

a) Construa um programa que represente estas entidades. Crie construtores, os métodos *set/get/hasCode/equals* (...) que lhe pareçam adequados e outros que sejam fundamentais para uma boa modulação e para o bom funcionamento do programa. Inclua mecanismos de controlo de exceções em todos os pontos do programa em que possam surgir situações imprevistas.

b) Teste as classes desenvolvidas com a função *main* seguinte.

```
public static void main(String[] args) {

   Localidade cid1 = new Localidade("Szohod", 31212,
         TipoLocalidade.Cidade);
   Localidade cid2 = new Localidade("Wadesdah", 23423,
         TipoLocalidade.Cidade);
   Localidade cid3 = new Localidade("BedRock", 23423,
         TipoLocalidade.Vila);
   Estado est1 = new Estado("North Borduria", 223133, cid1);
   Estado est2 = new Estado("South Borduria", 84321, cid2);

   Pais p1 = new Pais("Borduria", est1.getCapital());
   Pais p2 = new Pais("Khemed", cid2);
   Pais p3 = new Pais("Aurelia");
   Pais p4 = new Pais("Atlantis");
   p1.addRegiao(est1);
   p1.addRegiao(est2);
   p2.addRegiao(new Provincia("Afrinia", 232475, "Aluko Pono"));
   p2.addRegiao(new Provincia("Eriador", 100000, "Dumpgase Liru"));
   p2.addRegiao(new Provincia("Laurania", 30000, "Mukabamba Dabba"));

   List<Pais> org = new ArrayList<Pais>();
   org.add(p1);
   org.add(p2);
   org.add(p3);
```

```
        org.add(p4);

        System.out.println("----Iterar sobre o conjunto");
        Iterator<Pais> itr = org.iterator();
        while (itr.hasNext())
           System.out.println(itr.next());

        System.out.println("-------Iterar sobre o conjunto - For each (java 8)");
        for (Pais pais: org)
           System.out.println(pais);
        // ToDo:
        // adicionar, remover, ordenar, garantir elementos únicos
    }
```

Resultado esperado:

```
----Iterar sobre o conjunto
Pais: Borduria, Populaçao: 307454 (Capital: Cidade Szohod, populaçao 31212)
Pais: Khemed, Populaçao: 362475 (Capital: Cidade Wadesdah, populaçao 23423)
Pais: Aurelia, Populaçao: 0 (Capital: *Indefinida*)
Pais: Atlantis, Populaçao: 0 (Capital: *Indefinida*)
-------Iterar sobre o conjunto - For each (java 8)
Pais: Borduria, Populaçao: 307454 (Capital: Cidade Szohod, populaçao 31212)
Pais: Khemed, Populaçao: 362475 (Capital: Cidade Wadesdah, populaçao 23423)
Pais: Aurelia, Populaçao: 0 (Capital: *Indefinida*)
Pais: Atlantis, Populaçao: 0 (Capital: *Indefinida*)
```

c) Altere a funções *main* para testar outros métodos de *ArrayList* e outras condições do programa. Por exemplo, se substituir a instrução *Pais p2 = new Pais("Khemed", cid2);* por *Pais p2 = new Pais("Khemed", cid3);* o programa deverá gerar a seguinte saída:

```
Exception in thread "main" java.lang.IllegalArgumentException: Capital Inválida
   ...
```

d) Construa uma nova solução (*main* e classes) na qual a instrução *Pais p2 = new Pais("Khemed", cid3);* passe a ser verificada em compilação (e a dar erro) em vez de gerar exceções em *runtime*.

Problema 13.2

Construa um programa que leia um ficheiro de texto e que conte todos os pares de palavras encontrados no ficheiro e o número de ocorrências de cada par. Despreze todas as palavras de tamanho inferior a 3 e considere como separadores os seguintes caracteres:

```
\t\n.,:'‘’;?!-*{}=+&/()[]”“\"\'
```

O resultado deverá ser armazenado por ordem alfabética num ficheiro “output.txt”. Se utilizar a primeira frase deste problema o resultado no ficheiro de saída deverá ser o seguinte:

```
construa={programa=1}
conte={todos=1}
ficheiro={texto=1}
leia={ficheiro=1}
palavras={encontrados=1}
pares={palavras=1}
programa={que=1}
que={conte=1, leia=1}
texto={que=1}
```

```
todos={pares=1}
```

Para cada palavra encontrada no ficheiro o programa deverá associar um conjunto de todas as palavras seguintes contando igualmente quantas vezes cada par ocorre.
Teste o programa com o ficheiro "Policarpo.txt". O resultado deverá ser semelhante a:

```
1864={tratava=1}
abacateiros={entoando=1, mangueiras=1, oitenta=1, suas=1, troncos=1}
abacates={ora=1, pouco=1}
abacaxis={coroados=1, que=1}
abafada={comprimida=1, tens=1}
…
voltava={aos=1, biblioteca=1, com=1, ficava=1, idéia=1, olhava=1, para=1, seu=1}
voltavam={mesmo=1, oficial=1, para=1, sorridentes=1}
voltou={acidente=1, agarrou=1, alegria=1, aos=1, bonde=1, general=1, instante=1,
lhe=1}
```

Problema 13.3

Para cada uma das tarefas seguintes, defina a(s) estrutura(s) de dados mais adequada(s), e teste-a usando 3 ou mais elementos, com as operações adicionar, listar e remover.

a) A empresa Brinca&Beira (BB) precisa de um registo com os nomes de todos os seus empregados.
b) Em cada mês é selecionado aleatoriamente um funcionário para receber um brinquedo grátis. Deve ser possível guardar todos os pares funcionário-brinquedo.
c) A empresa decidiu atribuir o primeiro nome de um empregado a cada produto. Prepare uma lista destes nomes sabendo que um nome só poderá ser usado uma vez.
d) A BB decide, entretanto, que só quer usar os nomes mais populares para os seus brinquedos. Precisamos de uma estrutura com o número de funcionários que têm cada primeiro nome.
e) A empresa adquire ingressos para a próxima temporada da equipa local de futebol, para serem distribuídos rotativamente pelos funcionários. Crie a estrutura mais adequada – pode usar uma ordem qualquer.