2º Teste de "Introdução à Programação (A)" 2007-08
9 de Janeiro de 2008

Duração: 1:30H + 0:20 (tolerância)

Instruções importantes:

- Responda a todos os grupos em folhas separadas.
- Identifique todas as folhas com o seu número e nome.
- Antes de começar a resolver um exercício, leia o enunciado do princípio até ao fim.
- Pode usar caneta ou lápis.
- Não é permitido consultar elementos para além deste enunciado.
- Qualquer tentativa de fraude comprovada acarretará a reprovação na disciplina.
- Não é permitido sair da sala antes que o teste termine.

I.

Pretende-se programar, em Java, uma classe **`Player`** cujos objectos representam jogadores. Cada jogador tem um nome e uma pontuação. Os construtores da classe estão definidos da seguinte forma:

- **`public`** `Player(String name)`: Constrói um jogador com o nome dado e pontuação igual a zero;
- **`public`** `Player(String name, int points)`: Constrói um jogador com o nome dado e a pontuação dada.

A interface pública da classe **`Player`** está definida da seguinte forma:

- **`public`** `String getName()`: Devolve o nome do jogador;
- **`public int`** `getPoints()`: Devolve a pontuação do jogador;
- **`public void`** `setPoints(int points)`: Actualiza a pontuação do jogador com os pontos dados;
- **`public boolean`** `equals(Player c)`: Indica se dois jogadores são iguais. Dois jogadores são iguais se os seus nomes forem iguais.

Programe, em Java, a classe **`Player`**. Não se esqueça das variáveis de instância (memória ou estado) da classe.

II.

Para construir um jogo na Internet, para ser jogado entre equipas, é preciso definir uma classe **`PlayerTeam`** que representa uma equipa de jogadores. Cada jogador é representado por um objecto da classe **`Player`**.

Cada objecto da classe **`PlayerTeam`** representa uma equipa, e deve manter pontuações obtidas pelos seus jogadores durante um dado jogo. (**dica**: na classe **`PlayerTeam`** use um vector para guardar os vários jogadores).

O construtor da classe `PlayerTeam` é o seguinte:

- **`public`** `PlayerTeam(`**`int`** `n)`: Constrói um objecto da classe **`PlayerTeam`**. O parâmetro n indica o número máximo de jogadores que a equipa pode admitir.

A interface pública da classe **`PlayerTeam`** é a seguinte:

- **`public void`** `addPlayer(Player p)` : Adiciona um novo jogador à equipa, caso seja possível. Caso um jogador com o mesmo nome já exista ou a equipa não possa admitir mais jogadores, esta operação não faz nada;
- **`public int`** `countPlayersWithScore(int point)`: devolve o número de jogadores com pontuação igual à dada;
- **`public int`** `countPlayers()`: Devolve o número total de jogadores na equipa.

a) Programe, em Java, a classe **`PlayerTeam`**. . Não se esqueça das variáveis de instância (memória ou estado) da classe.

b) Escreva um programa principal em Java que leia, do ficheiro de texto "players.txt", as pontuações obtidas num dado jogo e escreva a seguinte informação na consola:

- Número de jogadores;
- Escreva, para cada pontuação entre 0 e 20, o número de jogadores com essa pontuação (histograma de pontuação).

O ficheiro de texto tem o seguinte formato: na primeira linha aparece um número inteiro não negativo que indica o total de jogadores e nas seguintes linhas aparecem as informações dos jogadores. Em cada linha de informação sobre um jogador aparece a pontuação seguida do nome do jogador, separados por um espaço.

III.

A abundância de um número inteiro positivo *n*, é dada pela fórmula $\alpha(n) = \sigma(n) - 2n$ em que $\sigma(n)$ é a soma de todos os divisores do número *n* (recorde que todos os números têm como divisores eles próprios e 1).

Um *número abundante* (ou *excessivo*) é um número cuja abundância é superior a zero.

Como exemplo considere o número 24. Os seus divisores são 1, 2, 3, 4, 6, 8, 12 e 24, cuja soma é 60. Como 60 é superior a 2×24, o número 24 é abundante. A sua abundância é $60 - 2 \times 24 = 12$.

Implemente a classe **`AbundantChecker`**. Para tal:

a) Indique as variáveis de instância (memória ou estado) da classe.

b) Programe o construtor da classe, cuja descrição é a seguinte:

- **`public`** `AbundantChecker(`**`int`** `number)` – Constrói um objecto da classe **`AbundantChecker`**, que recebe um número natural para analisar.

c) Programe o método:

- **`public int`** `sumOfDivisors()` – Devolve a soma dos divisores do número, incluindo o próprio.

d) Programe o método:

- **`public int`** `abundance()` – Devolve o valor da função de abundância para o número.

e) Programe o método:

- **`public boolean`** `isAbundant()` – Diz se o número em questão é ou não abundante.

IV.

Alguns números inteiros contêm sequências de dígitos repetidos, por exemplo, o número 225566650 contém 5 blocos de dígitos (1º bloco – 22, 2ºbloco – 55, 3º bloco – 666, 4ºbloco – 5, 5º bloco – 0). O bloco mais longo é o terceiro, de tamanho 3.

Pretende-se programar em Java uma classe **`BlockedNumber`** cujos objectos representam números inteiros não negativos, mas vistos como sequências de blocos.

O construtor da classe recebe um número inteiro não negativo que representa o número a analisar. As operações da classe **`BlockedNumber`** são as seguintes:

- **`public int`** `blockSizes()` : devolve um número inteiro que representa os tamanhos dos vários blocos do número, separados por zero;
- **`public int`** `getNumBlocks()`: devolve o número de blocos contidos no número.
- **`public int`** `getMaxBlock()` : devolve o tamanho do maior bloco do número;

Comportamento esperado dos objectos:

```
> BlockedNumber it1 = new BlockedNumber (225566650);
> BlockedNumber it2 = new BlockedNumber (10);
>it1. getMaxBlock ()
3 (int)
>it1. getNumBlocks ()
5 (int)
>it1. blockSizes ()
202030101 (int)
>it2. getMaxBlock ()
1 (int)
>it2. blockSizes ()
101 (int)
>it2. getNumBlocks ()
2 (int)
```

Programe, em Java, a classe **`BlockedNumber,`** `para tal`:

a) Indique as variáveis de instância (memória ou estado).

b) Programe a operação `blockSizes()`

c) Programe a operação `getNumBlocks()`

d) Programe a operação `getMaxBlock()`