Introdução à Ciência da Computação - MAC 110/115 - EP3

Campeonato Computacional de Futebol [1]

1 Introdução

O Campeonato Computacional de futebol é disputado por 10 clubes (indexados de 0 a 9). São eles respectivamente: Real Matrix, InSUNos, Winchester United, JAVAlis, KNUThil, ESCorpiões, SemCTRL, ENTERrados, GOLgle, GOL++.

A competição é disputada em um turno de pontos corridos, ou seja, todos jogam contra todos (sem repetição de jogos) e o campeão, obviamente, é o primeiro colocado. A classificação dos times é definida pelos seguintes critérios:

- **V**: vitórias;
- **E**: empates;
- **D**: derrotas;
- **GM**: gols marcados;
- **GS**: gols sofridos;
- **S**: saldo de gols ($GM - GS$);
- **PG**: pontos ganhos ($3V + E$);

Ao final do campeonato, deseja-se saber a classificação geral e algumas estatísticas do campeonato a serem explicitadas na próxima seção.

2 Implementação

A implementação consiste de uma classe *Campeonato* em um arquivo $Campeonato.java$. Você tem liberdade para criar variáveis e métodos, porém alguns métodos são requisitos do EP e têm que estar presentes. A formatação da impressão de resultados também fica por sua conta. Para explicar o trabalho, utilizamos notação matemática no texto que se segue. No código fonte, por outro lado, você deve utilizar variáveis com nomes mais significativos do que R, E, etc.

2.1 Gerando os resultados das partidas

Por motivos de simplicidade, a implementação não será dividida em rodadas. Isso significa que não importa a ordem em que serão gerados os resultados. No primeiro EP, você viu como gerar números aleatórios utilizando a classe **Random**. Os resultados das partidas serão gerados aleatoriamente e armazenados em uma matriz R de dimensões 10×10, onde R_{ij} significa que o time indexado por i fez R_{ij} gols no time indexado por j. Mais concretamente,

[1] baseado no ex. 13 de http://www.ime.usp.br/~macmulti/exercicios/matrizes/index.html

o placar do jogo InSUNOS e JAVAlis, por exemplo, é de $R_{13} \times R_{31}$. R será preenchida de acordo com a regra:

$$R_{ij} = \begin{cases} a_{ij}, & i \neq j \\ -1, & c.c., \end{cases}$$

onde a_{ij} é um número inteiro aleatório tal que $0 \leq a_{ij} \leq 9$.

Requisito 1: Crie um método `int[][] geraResultados()` para criação da matriz de resultados.

2.2 Estatísticas

Na Seção 1 apresentamos os critérios definidos para classificação dos times. Uma matriz de estatísticas 10×7 pode ser derivada da matriz de resultados R de forma simples. No exemplo a seguir, a linha 0 da matriz de Estatísticas corresponde ao time Real Matrix, a linha 1 ao time InSUNos e assim por diante, seguindo a ordem da seção 1. Observe que a impressão das estatísticas com os nomes de times (tabela 2) está em ordem alfabética do nome do time (e não na mesma ordem das linhas da matriz).

10	0	0	45	0	45	30
9	1	0	37	9	28	28
6	3	1	25	15	10	21
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabela 1: Exemplo de matriz de estatísticas

Time	V	E	D	GM	GS	S	PG
InSUNos	9	1	0	37	9	28	28
Real Matrix	10	0	0	45	0	45	30
Winchester United	6	3	1	25	15	10	21
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabela 2: Imprimindo a matriz de estatísticas por ordem de nomes

Requisito 2: Crie um método `int[][] calculaEstatisticas(int[][] R)` para calcular a matriz de estatísticas.

Requisito 3: Crie um método `void imprimeEstatisticas(int[][] E)` para mostrar na tela as estatísticas em ordem alfabética dos times. Veja abaixo como usar métodos auxiliares para implementar este método e o método do **Requisito 4**.

2.3 Classificação

Dada a matriz de estatísticas, pode-se derivar uma classificação dos times. Uma classificação realista dos times envolveria ordenar os times sucessivamente em relação a diversos critérios dentre aqueles definidos na Seção 1, sendo que a segunda ordenação serve para desempatar

os times empatados de acordo com o primeiro critério, a terceira ordenação desempata os times empatados em relação aos dois primeiros critérios e assim por diante. Para simplificar, utilizaremos apenas os PONTOS GANHOS como base para ordenação.

Requisito 4: Crie um método `void imprimeClassificação(int[][] E)` para imprimir os NOMES dos times de acordo com a classificação por PONTOS GANHOS (em ordem decrescente).

Requisito 5: Para tratar dos dois requisitos anteriores, crie uma **interface** `ComparaTimes`, que contém um único método `boolean éMenor(int i, int j)`, onde `i` e `j` são índices de times, de acordo com a ordem definida na seção 1. Esta interface deve ser implementada por 2 classes: `ComparaTimesPorNome` e `ComparaTimesPorPGDecrescente`. Para que os métodos destas classes tenham acesso à informação de que eles precisam (vetor de nomes no primeiro caso e vetor de PONTOS GANHOS no segundo caso), utilize construtores apropriados para enviar estas informações aos atributos das classes.

Já foi aprendido em sala de aula como ordenar um vetor. As ordenações dos times por nome ou em ordem decrescente de PONTOS GANHOS podem ser feitas indiretamente manipulando-se apenas um conjunto de índices que representam os times, como segue. Seja T um vetor de 10 posições representando os times, inicializado com valores crescentes de 0 a 9. A ordenação do vetor T usando como critério o método `éMenor` produzirá um vetor com os índices de 0 a 9 permutados, correspondendo ou à ordem dos nomes dos times ou à ordem dos valores de PG, dependendo da implementação da interface `ComparaTimes` utilizada. A adaptação dos algoritmos de ordenação vistos em aula é trivial, substituindo-se as comparações da forma $T_i < T_j$ por chamadas da forma `éMenor(T[i],T[j])`.

Requisito 6: Crie um método `int[] ordenaTimes(ComparaTimes comparador)`, que produz um vetor de índices que corresponda à ordenação dos times de acordo com o critério `comparador`. Este método deve ser utilizado pelos métodos dos **Requisitos 3 e 4**.

Requisito 7: Para demonstrar o funcionamento do seu programa, crie um método `main` para a classe *Campeonato*.

3 Recomendações

Todas as recomendações dadas nos últimos EPs continuam válidas. Em particular:

- coloque um cabeçalho informativo no início de seu EP.
- preste muita atenção à documentação do seu código!
- os EPs podem ser feitos em dupla, mas todos precisam submeter seus EPs no PACA.
- entregue apenas o arquivo .java

Bom Trabalho!