Universidade Federal de Minas Gerais
Instituto de Ciências Exatas - Departamento de Ciência da Computação

Programação Modular
2º Semestre de 2014
Prof. Douglas G. Macharet

Trabalho Prático 1 – Campeonato PM

Jean-Luc Nacif Coelho - 2011049207
João Francisco Moreira Penna - 2011049231

1 - Introdução:

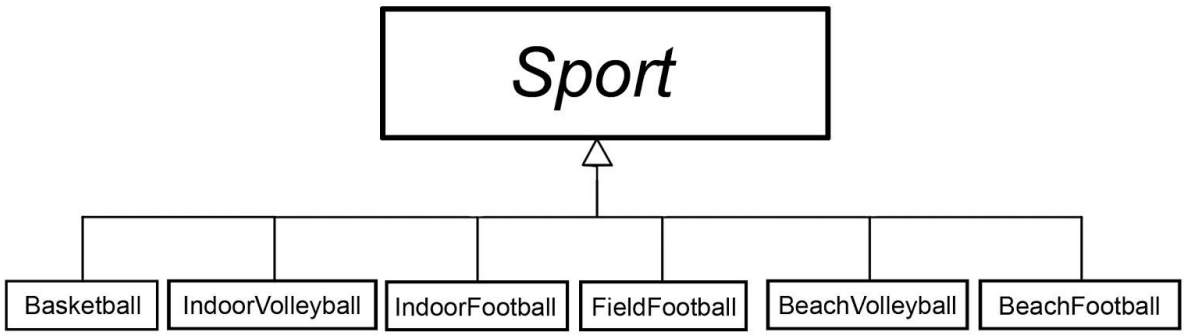
O objetivo deste trabalho prático é familiarizarmos com os principais conceitos de programação orientada a objetos e com a linguagem de programação Java. Para isso, construímos um programa capaz de acompanhar um campeonato de vários esportes e gerar um arquivo de saída com estatísticas do campeonato.

O funcionamento do programa é bem direto e intuitivo. Inicialmente é feito uma leitura de dados em arquivos de texto presentes no mesmo diretório do programa. Esses dados são cadastrados no sistema e com eles são construídos objetos que representam as equipes com suas características (esporte, país de origem, quais atletas). Nos arquivos de texto também estão presentes quais partidas são jogadas entre as equipes e seus respectivos placares. Com esses dados computados, geramos uma saída em formato de arquivo de texto exportando as estatísticas desejadas sobre o campeonato.

Para isso, implementamos algumas classes que vamos entrar em detalhes a seguir na documentação. Com poucas abstrações, já conseguimos simplificar o problema e fazer o programa acompanhar todo o campeonato.

2- Implementação:

A primeira classe criada foi a classe abstrata Sport. Ela tem características que são herdadas por seis esportes: Basquete, Vôlei, Futebol, Vôlei de praia, Futebol de areia e Futebol de salão. Ou seja, cada esporte tem sua própria subclasse, que levam o nome Basketball, IndoorVolleyball, IndoorFootball, FieldFootball, BeachVolleyball e BeachFootball. Optamos por nomear as classes e variáveis em inglês, por ser uma prática comum em vários ambientes de trabalho. Abaixo ilustramos um diagrama mostrando a relação de generalização das classes implementadas, ou seja, visualizando quais subclasses herdam da superclasse. Usamos a convenção visual baseada em UML, com retângulos representando as classes, o texto em itálico representando uma classe abstrata, e a seta indicando a relação de generalização.



Observamos que existem regras especiais para Vôlei de praia e Futebol de areia, nas quais o time vencedor é o time com o menor número de pontos. Isso poderia ser feito criando uma subclasse que partilharia essas características, mas por simplificação, apenas sobrescrevemos nas duas subclasses correspondentes alguns de seus métodos, mas sempre usando boas práticas de programação modular, como a notação @override nestes casos.

A classe Team é usada para representar as equipes. A decisão de quais atributos seriam necessários na implementação dessa classe veio diretamente de que dados eram passados nos arquivos de entrada. Assim, fazendo a leitura (que é detalhada no item 2.1 da documentação, logo a seguir), é possível compor os objetos que as representam corretamente. Como esse é um projeto em que a programação é orientada a objetos, essa classe possui alguns métodos importantes para imprimir as estatísticas e histórico dos jogos.

A classe Match é utilizada para as partidas em si. Ela que verifica quem foi o vencedor de uma partida a partir de uma instância de regras de esporte.

Ao longo do programa, como foi pedido, usamos classes que implementavam a interface Collection<E>. Algumas delas foram muito úteis, outras acabaram por ser substituídas ao longo do progresso com o código.. Informações sobre essa interface podem ser obtidas no link citado na bibliografia.

A classe PrintData é apenas uma classe de utilidade para definir se é para imprimir por seleção ou por esporte.

A classe Coreset é o conjunto principal do programa. Ela junta as informações e lança o main, inciando o InputProcessor - a leitura dos caracteres, e após o processamento dos dados, a impressão do arquivo de saída.

2.1 Leitura de arquivos de entrada:

Como mencionado anteriormente, as informações com as quais os objetos são construídos dentro do programa vem diretamente da leitura dos arquivos de extensão .txt que estão no mesmo diretório do programa. Esses cinco arquivos são descritos abaixo, primeiro o nome do arquivo de texto, seguido dos seus tipos de dados descritos, e por fim, o que esses dados representam.

esportes.txt : int ; string **->** idEsporte ; nome
selecoes.txt : int ; int ; string **->** idSelecao ; idEsporte ; nomePais
atletas.txt : int ; int ; string **->** id ; idSelecao ; nome
partidas.txt : int ; int ; int ; int ; int **->** id ; idSelecaoA ; idSelecaoB ; placarA ; placarB
estatisticas.txt : int ; int **->** idTipo ; (idEsporte ou idSelecao)

A leitura é feita através da classe InputProcessor, que usa o Scanner do Java para ler uma linha inteira, e o .split para dividir a string obtida em pedaços separados pelo caractere ponto-e-vírgula.

2.2 Escrita de arquivos de saída:

São exportados dois tipos de arquivos de saída, com estatísticas diferentes, detalhadas como o modelo a seguir:

estatistica-1-1.txt

```
Basquete
Times V     E     D
Japão 1     0     1
USA   0     1     1
Chile 1     1     0
```

estatistica-2-1.txt

```
Japão - Basquete
Atletas
Honda
Yamaha
Adversários      Placar
USA              50x37
Chile            19x31
```

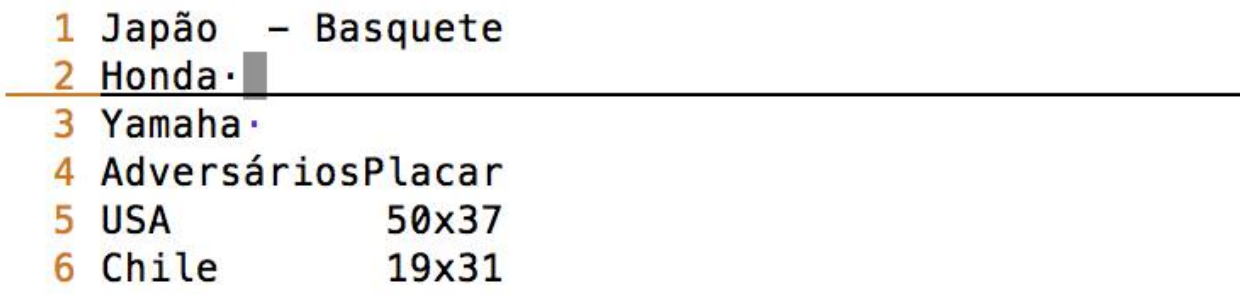
2.3 - Decisões de Implementação:

Algumas decisões foram tomadas tendo em vista a simplificação da resolução do problema. Por exemplo, ao representar os atletas, não percebemos como necessário a atribuição de um identificador único para cada um, e sim, que apenas a string com seu nome já seria o suficiente. Esse tipo de detalhe, ao nosso ver, não compromete o programa, sendo apenas uma decisão de implementação e modelagem. Da mesma maneira, os outros “id” estão sendo usados implicitamente. Isso não altera o resultado do programa, mas assume certas condições, por exemplo, que os ids sempre vem ordenados corretamente nos arquivos de entrada passados.

3- Testes

Para a entrada padrão, o programa atribuiu as estatísticas esperadas. Abaixo a saída para a entrada da especificação:

```
1 Basquete
2 Times        V          E          D
3 Japão        1          0          1
4 USA          0          1          1
5 Chile        1          1          0
~
~
```



Aqui cabe mencionar um defeito estético na saída que não foi tratado. Caracteres especiais como o ~ de Japão contam como dois caracteres mas ocupam o espaço de apenas um, dessa forma, o arquivo de saída “estatistica-1-1.txt” não fica completamente alinhado como deveria.

Também fizemos alguns outros testes com outras entradas, e novamente, o programa continuou com o comportamento esperado. Também previmos no código algumas possíveis exceções, como por exemplo, uma partida entre duas equipes iguais, ou com uma equipe que não existe.

4- Conclusão

De maneira geral, esse trabalho propiciou um maior contato com a linguagem de programação Java, com as técnicas de programação orientada a objetos e programação modular, e um amadurecimento do processo de trabalho.

As maiores dificuldades foram resolver detalhes de implementação por falta de familiaridade com a linguagem de programação Java. Como implementar a estrutura “collections” era um requisito, gastamos um tempo maior porque voltamos atrás em qual das classes que seria a melhor para ser empregada em nosso trabalho prático. Várias partes do código tiveram que ser refeitas e retrabalhadas visto a dificuldade de lidar com erros inesperados. Além disso, dificuldades no processamento de caracteres, coisas como “trailing spaces” atrasaram bastante o correto funcionamento do programa. Porém, isso não impediu o progresso no trabalho.

O objetivo do trabalho era fazer um programa que conseguisse acompanhar um campeonato e suas partidas e gerar um arquivo de saídas com as estatísticas. Com as classes implementadas e seguindo a especificação, conseguimos com sucesso obter resultados bem sucedidos como observado nos testes. Utilizamos várias das técnicas e práticas aprendidas em sala de aula, e observamos que o trabalho fica notavelmente mais legível e organizado. Por causa disso consideramos esse Trabalho Prático um grande sucesso.

5- Bibliografia

- MEYER, Bertrand. **Object-Oriented Software Construction.** Prentice Hall, 2000.
- WU, Thomas C. **An Introduction to Object-Oriented Programming with Java.** McGraw-Hill, 2009.

- SAVITCH, Walter. MOCK, Kenrick. **Absolute Java.** Addison-Wesley, 2012.
- DEITEL, Paul. DEITEL, Harvey. **Java: How to Program.** Prentice Hall, 2011.
- site: http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html.

6- Link para o Repositório no GitHub

- site: https://github.com/Temennigru/TPPM01