

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

Programação Modular
2º Semestre de 2016
Prof. Douglas G. Macharet

Alunos: Caíque Bruno Fortunato
Pâmela Carvalho da Silva

Matrícula: 2013062731
2013073474

Trabalho Prático 1 - Olimpíadas Modulares

Documentação

1. Introdução

A Olimpíada é o período em que ocorrerem os Jogos Olímpicos, compondo um evento desportivo que ocorre a cada quatro anos, reunindo atletas de quase todos os países do mundo para competirem em várias modalidades. Aos três primeiros classificados de cada prova, são atribuídas medalhas de ouro (primeiro classificado), prata (segundo classificado) e bronze (terceiro classificado).

O primeiro trabalho prático de Programação Modular consiste na construção de um sistema que simula a exibição dos resultados de uma olimpíada que contém cinco modalidades diferentes, sendo elas: Corrida, Natação, Levantamento de Peso, Salto em altura e Ginástica Artística.

Para que o programa funcione corretamente são fornecidos como entrada os nomes e identificadores dos países participantes da olimpíadas assim como os atletas com suas respectivas notas nas modalidades que disputam.

Após a inserção dos dados (a maneira correta da inserção será especificada mais adiante) o sistema retorna dois tipos de estatísticas, de acordo com a escolha do usuário: um quadro de medalhas contendo a classificação geral dos países em todos os esportes e a classificação final para um esporte específico.

2. Implementação

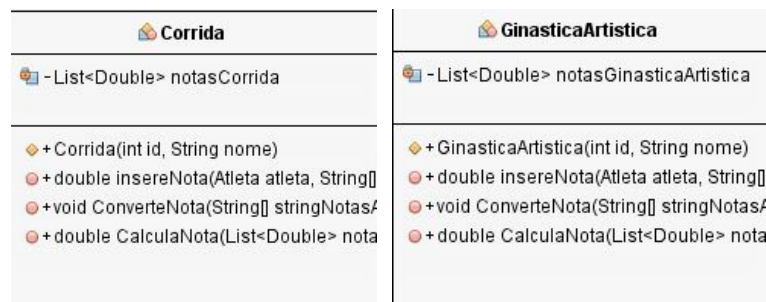
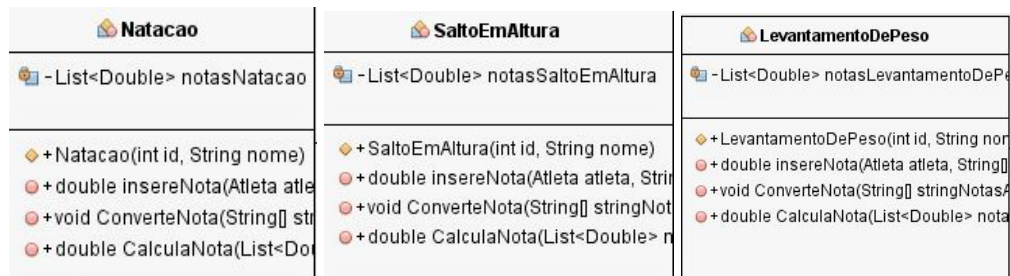
Para o desenvolvimento do sistema foram utilizados conceitos de Programação orientada a objetos através da linguagem JAVA SDK 8 construído na NetBeans IDE 8.1.

Abaixo, segue o detalhamento das estruturas de dados.

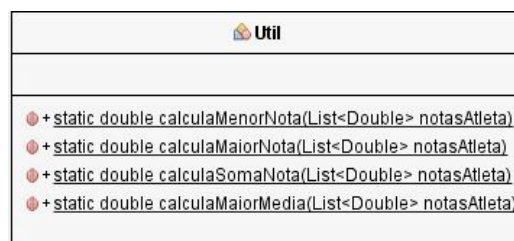
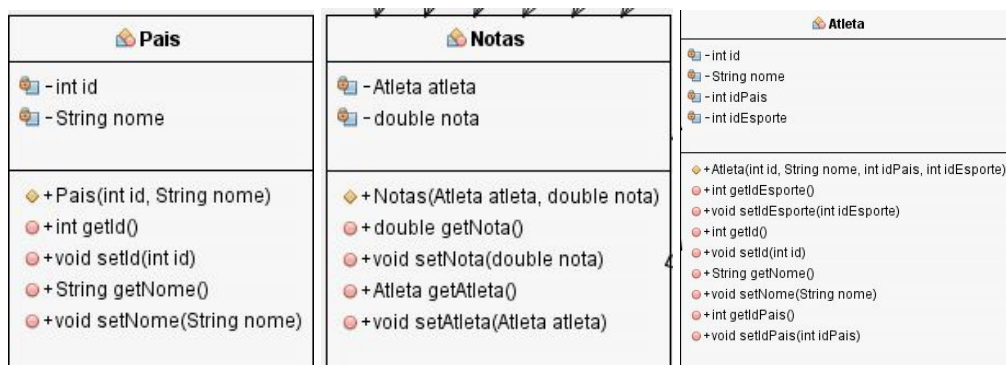
Superclasse Esporte:
























Subclasses:














Outras classes:



 DeterminaMedalhas
 + <u>static void SetaMedalhas(ArrayList<Notas> listaNotas, ArrayList<MedalhasPais> medalhasPais)</u>

 NotasEsporte
 #ArrayList<Notas> listaNotasCorrida  #ArrayList<Notas> listaNotasNatacao  #ArrayList<Notas> listaNotasLevantamentoDePeso  #ArrayList<Notas> listaSaltoEmAltura  #ArrayList<Notas> listaGinasticaArtistica  #ArrayList<Notas> listaMedalhasPais
 -void OrdenaNotas()  +void InsereNotaCorrida(Athleta atleta, Double menorNota)  +void ImprimeNotaCorrida(Corrida corrida)  +void InsereNotaNatacao(Athleta atleta, Double menorNota)  +void ImprimeNotaNatacao(Natacao natacao)  +void InsereNotaLevantamentoDePeso(Athleta atleta, Double maiorNota)  +void ImprimeNotaLevantamentoDePeso(LevantamentoDePeso levantamentoDePeso)  +void InsereNotaSaltoEmAltura(Athleta atleta, Double maiorNota)  +void ImprimeNotaSaltoEmAltura(SaltoEmAltura saltoEmAltura)  +void InsereNotaGinasticaArtistica(Athleta atleta, Double maiorNota)  +void ImprimeNotaGinasticaArtistica(GinasticaArtistica ginasticaArtistica)  +void ImprimeQuadroMedalhas(ArrayList<MedalhasPais> medalhasPais)

 LeDados
 - Corrida corrida  - Natacao natacao  - LevantamentoDePeso levantamentoDePeso  - SaltoEmAltura saltoEmAltura  - GinasticaArtistica ginasticaArtistica
 +void LeituraPais(ArrayList<Pais> listaPais)  +void LeituraAtleta(ArrayList<Atleta> listaAtleta, NotasEsporte notasAtleta)  +void LeituraSaida(NotasEsporte notasAtleta, ArrayList<Pais> listaPais)

 TPOlimpiadas
 + <u>static void main(String[] args)</u>



Estruturas de dados e funcionamento das principais funções

Neste tópico será apresentado o funcionamento das principais funções e procedimentos utilizados no desenvolvimento da aplicação.

Classes gerais:

Atleta	Classe que possui atributos dos dados do atleta (id, nome, idPais e idEsporte), possuindo apenas construtor e métodos get e set.
Pais	Classe que possui atributos dos dados do país (id, nome), possuindo apenas construtor e métodos get e set.
MedalhasPais	Classe que possui atributos dos dados das medalhas que cada país possui (prata, ouro e bronze), incluindo um tipo País. Tal classe possui construtor e métodos get para manipular seus atributos (atleta, ouro, prata e bronze). Além disso, a classe possui também métodos para incrementar o valor de medalhas.

Notas	Classe que possui a nota final de cada atleta, incluindo um tipo atleta.
Esporte (SuperClasse)	<p>Classe abstrata que possui o id e nome do esporte, contendo os seguintes métodos finais e abstratos:</p> <ol style="list-style-type: none"> 1. InsereNota - Função geral que calcula a nota final do atleta, para isso ele chama as seguintes funções abaixo: 2. ConverteNota - Converte o vetor de notas passado por parâmetro de string para o formato double. 3. CalculaNota - Com o array de notas em double, é calculado a nota final, conforme especificação de cada esporte
Subclasses de Esporte: Natação, Corrida, Ginástica Artística, Levantamento de Peso, Salto em Altura	<p>Subclasses que herdam da classe esporte, possuindo métodos polimórficos herdados da classe pai. Cada método calcula a nota de acordo com a especificação da modalidade.</p> <p>As subclasses possuem também um vetor temporário para armazenar as notas convertidas de string para double.</p>

Demais classes:

Util é uma classe que possui apenas métodos estáticos realizam procedimentos que irão ocorrer em diferentes classes. Por exemplo: a corrida e a natação calculam o menor tempo considerando as notas do atleta. De modo a evitar a repetição do código, tais classes utilizam métodos da Util.

Nome: Util	Descrição dos métodos
calculaMenorNota	Calcula a menor nota que um determinado atleta recebeu em uma determinada modalidade.
calculaMaiorNota	Calcula a maior nota que um determinado atleta recebeu em uma determinada modalidade.
calculaSomaNota	Calcula a soma das notas que um determinado atleta recebeu em uma determinada modalidade.
calculaMaiorMedia	Calcula a média que um determinado atleta recebeu em uma determinada modalidade.

LeDados é uma classe que realiza as leituras dos dados existentes na especificação da aplicação, sendo eles: a leitura de pais.txt (contendo os dados do país), a leitura de atletas.txt (contendo os dados dos atletas: id, idPais, idEsporte, Notas) e a leitura de estatisticas.txt (contendo as saídas desejadas). Além de ler os dados, a LeDados também insere o conteúdo nas listas de objetos criadas.

Por fim, a LeDados cria os objetos para cada modalidade, como forma de inicializá-los.

Nome: LeDados	Descrição dos métodos
LeituraPais	<ol style="list-style-type: none"> 1. Lê o arquivo pais.txt 2. Recebe uma lista de objetos do tipo Pais como parâmetro e insere na mesma o id e nome do país existentes no arquivo de texto.
LeituraAtleta	<ol style="list-style-type: none"> 1. Lê o arquivo atletas.txt 2. Recebe uma lista de objetos do tipo Atleta como parâmetro e insere na mesma o id,

	<p>nome, idPaís, e idEsporte do atleta existentes no arquivo de texto.</p> <p>3. Lê as notas de cada atleta, salva em um array de notas enviando para um método da classe do esporte correspondente calcular a nota final do atleta.</p> <p>4. Passa como parâmetro para um método da classe NotasAtleta o atleta lido e sua nota final.</p>
LeituraSaida	<p>1. Lê o arquivo estatísticas.txt</p> <p>2. Identifica o tipo de estatística desejado: 1 ou 2</p> <p>3. Passa como parâmetro para os métodos necessários da classe NotasAtleta para que a mesma exiba os resultados.</p>

NotasEsporte é uma classe que possui métodos que imprimem os resultados desejados pelo usuário, sendo chamada na classe LeDados, possuindo também métodos que inserem as notas de determinado atleta. Tal classe funciona como um juiz geral: é acionada para determinar a nota de um atleta e imprimi-la, assim como o quadro de medalhas.

Além disso, a classe cria listas de objetos relacionadas às notas de cada modalidade (uma lista para cada modalidade) e também uma lista para as medalhas que cada país recebeu.

Nome: NotasEsporte	Descrição dos métodos
OdenaNotas	Ordena a lista de Notas (tipo Notas) contendo as notas finais de cada atleta que disputou os jogos de acordo com os critérios de cada modalidade.
InsererNota"X"	Inserer a nota final do atleta na lista de Notas

	da modalidade correspondente, contendo o tipo Atleta e a nota final.
ImprimeNota"x"	Imprime a lista de notas ordenadas de uma determinada modalidade.
ImprimeQuadroMedalhas	<ol style="list-style-type: none"> 1. Chama o método SetaMedalhas da classe DeterminaMedalhas para determinar quantas medalhas de ouro, prata e bronze cada país recebeu em cada modalidade. 2. Imprime o quadro de medalhas final.

DeterminaMedalhas determina a quantidade de medalhas de ouro, prata e bronze que cada país recebeu nas olímpiadas.

Nome: DeterminaMedalhas	Descrição dos métodos
SetaMedalhas	Incrementa o valor de 1 na quantidade de medalhas de ouro, prata e bronze correspondentes às 1°, 2° e 3° posições do vetor de notas já ordenado passado como parâmetro.

Decisões tomadas e demais detalhes

- Para representar as coleções de objetos existentes nas classes foram utilizados array de listas (ArrayList).
- Uma das peculiaridades é encontrada na ordenação. No quadro de medalhas, por exemplo, é necessário ordenar as medalhas de ouro e, caso aconteça empate é necessário ordenar as demais medalhas. Para isso, foi criada uma classe chamada: *OrdenaQuadroMedalhas* que implementa uma interface

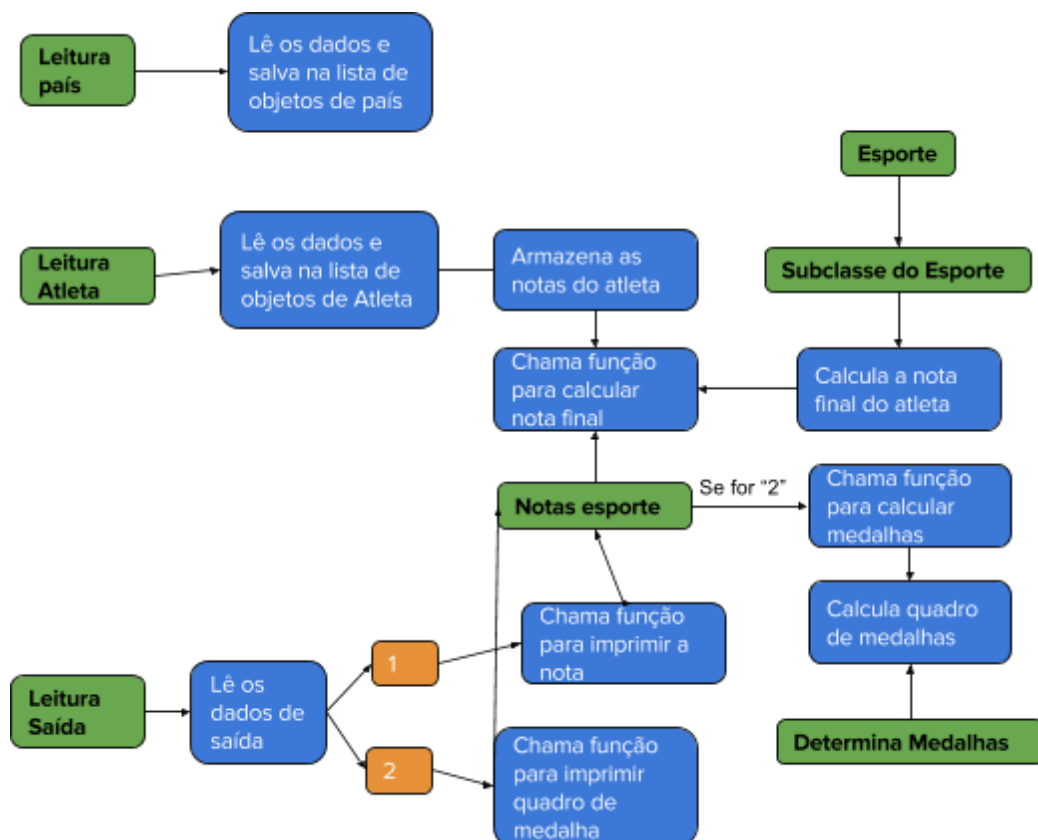
Comparator, para ser utilizada no método sort da Collections. Um dos métodos da classe é a compare, que compara dois objetos, se eles forem iguais é realizada a ordenação dos subatributos.

- A lista de objetos das notas do atleta foi ordenada de maneira semelhante com a ordenação do Quadro de Medalhas, conforme explicado acima, através da classe OrdenaNotasAtleta.

Exemplo de funcionamento

De forma a melhor representar o funcionamento do programa e o comportamento das classes foi construído um diagrama que exemplifica a ordem de execução da aplicação.

Como não foi lecionado ainda UML o diagrama realizado é bem simples, servindo apenas visualização para compreender melhor o que já foi apresentado neste tópico através das imagens e descrições.



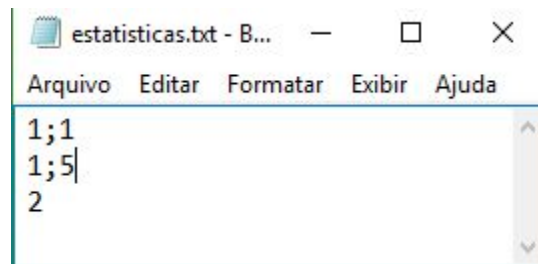
3. Testes

Para verificar o funcionamento do programa foram realizados diversos testes, incluindo o exemplo básico da documentação.

a. Teste 1

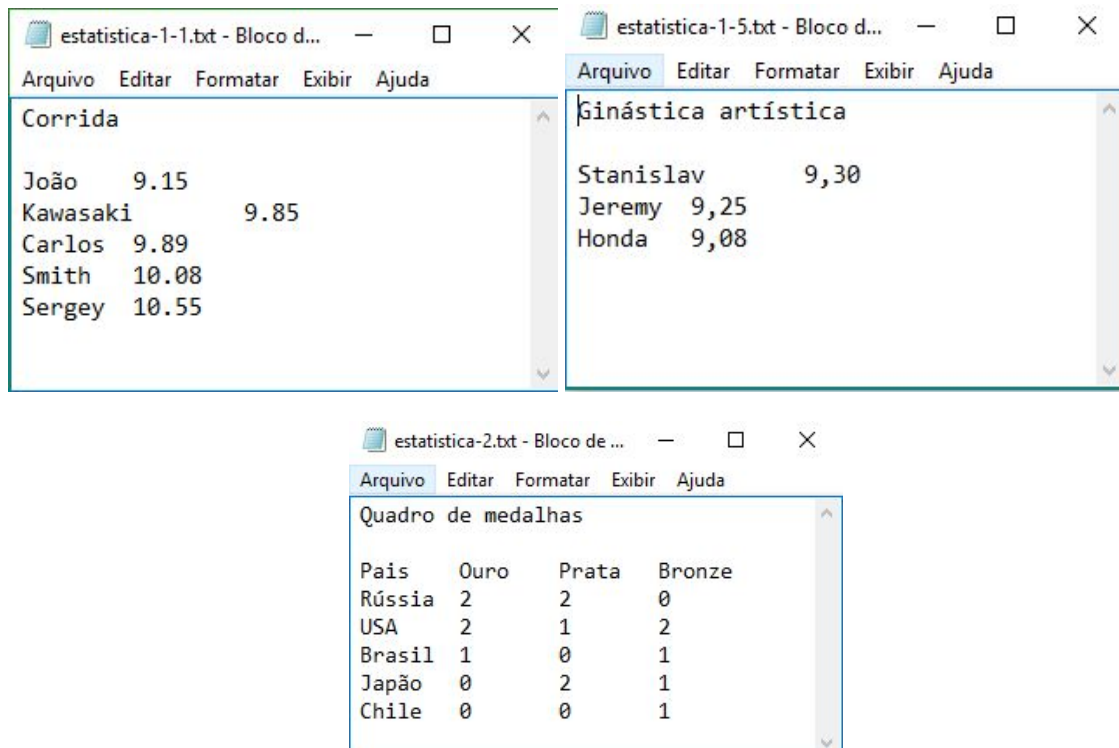
O primeiro teste consiste na execução do arquivo de teste, disponibilizado no Moodle.

i. Entrada:



```
estatisticas.txt - B...
Arquivo  Editar  Formatar  Exibir  Ajuda
1;1
1;5
2
```

ii. Saída:



```
estatistica-1-1.txt - Bloco d...
Arquivo  Editar  Formatar  Exibir  Ajuda
Corrida

João    9.15
Kawasaki      9.85
Carlos  9.89
Smith   10.08
Sergey  10.55

estatistica-1-5.txt - Bloco d...
Arquivo  Editar  Formatar  Exibir  Ajuda
Ginástica artística

Stanislav      9,30
Jeremy  9,25
Honda   9,08

estatistica-2.txt - Bloco de ...
Arquivo  Editar  Formatar  Exibir  Ajuda
Quadro de medalhas

Pais    Ouro    Prata    Bronze
Rússia  2        2        0
USA     2        1        2
Brasil  1        0        1
Japão   0        2        1
Chile   0        0        1
```

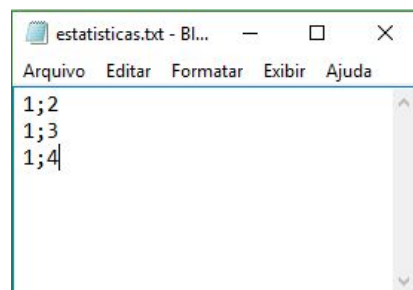
No exemplo acima, foi solicitado no arquivo de entrada a exibição da classificação geral da modalidade de identificador 1 e 5 (Corrida e Ginástica Artística), além do quadro de medalhas.

Vemos que a saída é exibida conforme o esperado.

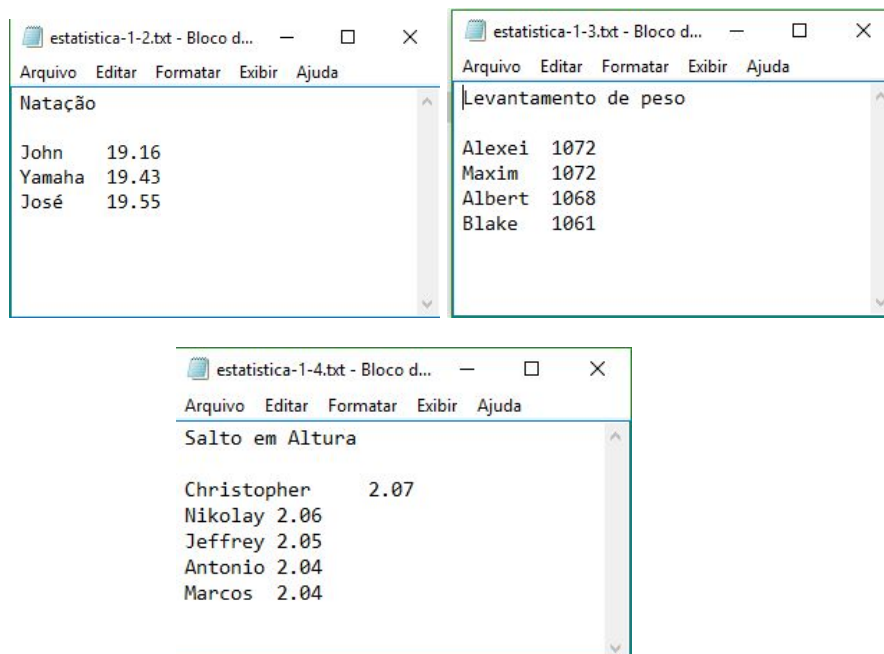
b. Teste 2 (Exibe demais estatísticas):

Como o teste anterior apresentou as estatísticas para o primeiro e segundo esportes, um novo caso de teste foi feito a fim de exibir o resultado dos esportes faltantes.

i. Entrada:



ii. Saída:



No exemplo acima, foi solicitado no arquivo de entrada a exibição da classificação geral da modalidade de identificador 2, 3 e 4 (Natação, Levantamento de Peso e Salto em Altura). Vemos que a saída é exibida conforme o esperado.

iii. Demais informações

Testamos ainda casos de exceção em que algum dos arquivos de entrada continham id de esporte não existente, por exemplo, e o problema foi tratado corretamente, ou seja, a execução foi interrompida e a mensagem de erro apresentada no console.

Os testes foram realizados em uma máquina com sistema operacional Microsoft Windows 10, Intel Core i5 5ª geração com 8GB de RAM e o tempo total médio para todos os casos apresentados acima foi de 0 segundos.

4. Conclusão

Com a finalização do trabalho foi possível alcançar dois principais objetivos pretendidos com o mesmo: a familiarização com o paradigma de Orientação a Objetos e a familiarização da linguagem Java, que é a forma de comunicação entre o que foi aprendido com a aplicação de OO.

No entanto, embora os resultados apresentados foram corretos é possível melhorar ainda mais a orientação a objetos do código, o que não foi feito devido ao tempo. Isso porque foi a primeira vez que a dupla manuseou a linguagem Java e utilizou o paradigma, sendo assim um bom tempo serviu para que fosse possível conhecer melhor os recursos e funcionamento do Java, assim como a modelagem do problema.

Como foi o primeiro contato com o Java, pode ser que alguns procedimentos que já estão implementados foram feitos “à mão”, no entanto, isso é algo que pretendemos melhorar nos próximos trabalhos, assim como a utilização da orientação a objetos.

A principal dificuldade do trabalho foi o uso da linguagem Java, principalmente na leitura de arquivos e no tratamento da ordenação, já que demoramos a entender e aplicar o método correto que permite tal operação.

Como o funcionamento da aplicação foi de 100% nos casos de testes, assumimos que o mesmo funciona, cumprindo o que foi solicitado.

No entanto, a principal contribuição do trabalho foi a modelagem e aplicação da orientação a objetos e reconhecermos que podemos melhorar nos outros trabalhos, preocupando cada vez mais com os princípios de qualidade do software.

5. Bibliografia

Definição de olimpíadas:

<https://mplitteraty.wordpress.com/arquivos/o-que-e-um-olimpiada/>

API: <http://docs.oracle.com/javase/8/docs/api/>