Universidade Federal do Espírito Santo
Centro Tecnológico
Departamento de Informática

Programação III (INF 09331) – 2019/2
Prof. Jordana Salamon ft. Gabriel Nogueira (adaptado de Prof. Vítor E. Silva Souza)

Especificação do Trabalho Prático

O trabalho prático da disciplina consiste em desenvolver o sistema computacional para solução do problema descrito abaixo na linguagem de programação apresentada durante a segunda metade do curso: Java .

Novidades em relação a versões anteriores da especificação estão marcadas em amarelo.

1. Descrição do problema

O Programa de Pós-Graduação em Informática (PPGI) da UFES precisa de um sistema de facilite o processo de avaliação de docentes credenciados ao programa.[1] Tal cálculo é feito manualmente todo ano e a coordenação do programa gostaria de um software que automatizasse este cálculo.

Os docentes são avaliados anualmente pelas publicações que fizeram nos últimos anos. O PPGI possui regras de pontuação baseadas na qualificação dos veículos (conferências e periódicos onde os artigos são publicados), que é feita de acordo com seus fatores de impacto. Os veículos são divididos em categorias, chamadas Qualis: A1, A2, B1, B2, B3, B4, B5 e C (A1 tendo os maiores fatores de impacto e C os menores). O PPGI, então, associa determinada pontuação a cada nível, ex.: publicação B1 ou superior vale 10 pontos; B2 ou B3 vale 5 pontos; B4 ou B5 valem 1 ponto.

Publicações em periódicos valem mais, portanto o PPGI determina um fator multiplicador aplicado se a publicação é em periódico (ex.: 1,5 indica que valem 50% mais; 2,0 indica que valem o dobro, etc.). O sistema deve permitir o registro das regras de pontuação do PPGI, que indicariam ainda a vigência (datas de início e fim, pois as regras mudam ao longo do tempo), quantos anos devem ser considerados ao analisar as publicações de um docente (ex.: últimos 2 anos, últimos 3 anos) e qual a pontuação mínima que um docente deve alcançar para se manter credenciado ao programa (ex.: 20 pontos).

Para efetuar o cálculo da pontuação de cada docente, é preciso saber: (1) quais veículos existem; (2) qual a pontuação de cada veículo a cada ano; e (3) quais as publicações de cada docente. Para a parte (1), o sistema deve registrar a sigla, nome e fator de impacto (um número real) de cada veículo. Para periódicos, registra-se ainda código ISSN. Para a parte (2), o sistema deve permitir importar arquivos de avaliação referentes a um determinado ano, indicando qual é o Qualis de cada veículo. Para a parte (3), o sistema deve registrar, para cada publicação, ano, veículo, título, páginas inicial e final e quais docentes, dentre os cadastrados no sistema, são autores. Para publicações em periódico, registra-se também número e volume. No caso de conferências, registra-se também o número e o local da mesma. O cadastro de docente deve incluir nome, data de nascimento, data de ingresso no programa e se o docente é o coordenador do programa.

De posse de todas as informações, o sistema será usado para fins de recredenciamento de docentes: dado um ano (ex.: 2017), o sistema deverá calcular a pontuação de cada docente, verificando se o valor mínimo para recredenciamento foi atingido, considerando o sistema de pontuação vigente no dia 01/01 do ano especificado. Não entram nesse

[1] Inspirado em: http://www.informatica.ufes.br/pos-graduacao/PPGI/credenciamento-de-docentes

cálculo, no entanto, o coordenador do programa, membros que entraram a menos de 3 anos e membros que possuem acima de 60 anos de idade (são automaticamente recredenciados). Outros relatórios trazem dados de publicações (ano, veículo, Qualis e docentes) e estatísticas (número de artigos publicados por Qualis).

2. Formatos de entrada e saída

Os cadastros dos dados de docentes, veículos, publicações, etc. são feitos em planilhas eletrônicas. Para o processamento destes dados e geração dos relatórios desejados, as planilhas serão exportadas para um formato de texto simples com valores separados por vírgulas, conhecido como CSV (*Comma Separated Values*). No entanto, para evitar conflito com representação de valores decimais (ex.: 8,9), os dados serão exportados utilizando ponto-e-vírgula como separadores (ex.: SPE;Software: Practice and Experience;P;1097-024X;0,652 – representando que periódico Software: Practice and Experience tem fator de impacto 0,652).

Para facilitar a leitura dos relatórios produzidos pelo programa, será feita a importação dos dados dos relatórios do formato CSV para planilha eletrônica. Portanto, seu programa deve ser capaz, além de ler dados neste formato, também gerar os relatórios em CSV.

Esta seção descreve os dados que estarão presentes em cada um dos arquivos de entrada e os dados que devem estar presentes em cada um dos arquivos de saída (relatórios). Para saber como estes dados serão formatados, verifique os arquivos de exemplo disponibilizados juntamente com esta descrição.

É muito importante que o programa siga os padrões de formatação prescritos, pois do contrário pode apresentar erro na leitura ou diferenças nos relatórios durante a correção automatizada dos trabalhos (vide Seção 4). Note que tanto os arquivos de entrada quanto os de saída possuem linhas de título que devem ser levadas em consideração (ou seja, descartadas durante a leitura das planilhas de entrada e inseridas durante a escrita dos relatórios de saída).

2.1. Entrada de dados

São cinco os arquivos de entrada de dados:

- Planilha de docentes;
- Planilha de veículos;
- Planilha de publicações;
- Planilha de qualificações;
- Planilha de regras de pontuação.

Os nomes dos arquivos são especificados durante a execução do programa (vide Seção 3). Abaixo encontra-se especificada a ordem que os dados devem aparecer em cada um destes arquivos:

Planilha de docentes

<código>;<nome>;<data-nascimento>;<data-ingresso>;<coordenador?>

O código é numérico (inteiro, 16 dígitos); nome pode ser lido como texto; as datas são informadas no formato dd/mm/aaaa; e o coordenador é indicado com um X na respectiva coluna.

Planilha de veículos

<sigla>;<nome>;<tipo>;<fator de impacto>;<issn>

Sigla, nome e ISSN podem ser lidos como texto; tipo é um caractere, sendo 'C' para conferências e 'P' para periódicos; fator de impacto é numérico (real).

Planilha de publicações

<ano>;<sigla veículo>;<título>;<lista de autores>;<número>;<volume periódico>;<local conferência>;<página inicial>;<página final>

Ano, número, volume e páginas são numéricos (inteiros); sigla, título e local podem ser lidos como texto; a lista de autores é uma lista de códigos de docentes, separados por vírgula.

Planilha de qualificações

<ano>;<sigla-veículo>;<Qualis>

O ano é numérico (inteiro); a sigla do veículo pode ser lida como texto; o Qualis é um dos valores possível para qualificação do veículo: A1, A2, B1, B2, B3, B4, B5 ou C.

Planilha de regras de pontuação

<data início>;<data fim>;<lista de Qualis>;<lista de pontos>;<multiplicador periódicos>;<quantidade de anos a considerar>;<pontuação mínima para recredenciamento>

Datas são informadas no formato dd/mm/aaaa; o multiplicador é numérico (real); quantidade de anos e pontuação mínima são numéricos (inteiros). As listas de Qualis e de pontos possuem o mesmo tamanho, a primeira indicando os limites de cada categoria de pontuação e a segunda os respectivos pontos (números inteiros). Por exemplo, "A1,B2,B4,C;10,5,1,0" indica que de A1 a B1 vale 10 pontos, B2 a B3 vale 5 pontos, B4 a B5 vale 1 ponto e C vale 0 pontos.

2.2. Processamento

Lidos todos os dados, o programa deve criar objetos em memória representando as informações contidas nos arquivos de entrada. Tais objetos devem estar ligados adequadamente, conforme as associações entre as classes de objetos, observando os princípios da orientação a objetos.

Com os dados em memória (representados pelos objetos), o programa deve proceder para a escrita dos relatórios.

2.3. Escrita dos relatórios

Os relatórios gerados devem ser escritos em arquivos com os seguintes nomes:

Relatório	Nome do Arquivo
Relatório de recredenciamento	`1-recredenciamento.csv`

Relatório	Nome do Arquivo
Lista de publicações	`2-publicacoes.csv`
Estatísticas de publicações	`3-estatisticas.csv`

Abaixo encontra-se especificada a ordem que os dados devem aparecer em cada um destes arquivos:

Relatório de recredenciamento

```
<nome do docente>;<pontuação alcançada>;<recredenciado?>
```

Este relatório deve ser ordenado por nome do docente, em ordem crescente. Pontuação e recredenciamento devem ser calculados segundo a regra vigente no dia 01/01 do ano em questão. A pontuação deve ser indicada com 1 casa decimal. A última coluna indica os casos de credenciamento automático ou o credenciamento por pontos, conforme especificação a seguir:

- "Coordenador": docente é coordenador;
- "PPJ": docente entrou no programa há menos de 3 anos;
- "PPS": docente possui mais de 60 anos;
- "Sim": docente não se enquadra nos casos acima e alcançou a pontuação mínima;
- "Não": docente não se enquadra nos casos acima e não alcançou a pontuação mínima.

Lista de publicações

```
<ano>;<sigla veículo>;<nome veículo>;<Qualis>;<fator de
impacto>;<título da publicação>;<nomes dos docentes>
```

Este relatório deve ser ordenado por Qualis, em ordem decrescente (A1 primeiro), seguido por ano (decrescente) por sigla do veículo (crescente) e, por fim, pelo título da publicação (crescente). O fator de impacto deve ser formatado com 3 casas decimais e o nome dos docentes deve estar separado por vírgulas. Note que este relatório considera todas as publicações informadas e não apenas uma quantidade de anos, como no caso do recredenciamento.

Estatísticas de publicações

```
<Qualis>;<número de artigos>;<número de artigos por docente>
```

Este relatório deve ser ordenado por Qualis, em ordem decrescente (A1 primeiro). A segunda coluna indica o número total de artigos publicados em veículos com o respectivo Qualis (somatório em que cada artigo soma 1). Já a última coluna indica o número de artigos por docente em cada Qualis (mesmo somatório, só que cada artigo soma 1 dividido pelo número de docentes autores). A última coluna deve ser formatada com 2 casas decimais. Este relatório também considera todas as publicações.

2.4. Tratamento de exceções

Leitura de dados de arquivos, formatação, etc. são fontes comuns de erros e exceções. Além disso, é possível que os dados das planilhas não estejam consistentes. Seu programa deve tratar **apenas** os seguintes tipos de erro:

1. Erros de entrada e saída de dados como, por exemplo, o arquivo especificado não existir ou o programa não ter permissão para ler ou escrever em um arquivo. Nestes casos, o programa deve exibir a mensagem "Erro de I/O" (sem as aspas) e terminar sem produzir arquivos de saída;
2. Erro de formatação dos dados nos arquivos, ou seja, um valor formatado de forma incorreta nos arquivos de entrada (ex.: encontrado caractere onde esperava-se um número), causando erros de *parsing* dos dados. Nestes casos, o programa deve exibir a mensagem "Erro de formatação" (sem as aspas) e terminar sem produzir arquivos de saída;
3. Inconsistência nos dados, ou seja, não conformidade com a especificação dos arquivos de entrada da seção 2.1 ou com a descrição do problema na seção 1. Seu programa deve tratar **apenas** as inconsistências elencadas abaixo, exibindo a mensagem associada e terminar sem produzir arquivos de saída:

Inconsistência	**Mensagem**
O mesmo código foi usado para dois docentes ou veículos diferentes.	Código repetido para <objeto>: <código>. *(Substituir <objeto> por docente ou veículo; substituir <código> pelo código em questão. Aplicar este mesmo conceito nas mensagens abaixo).*
Sigla de veículo especificada para uma publicação não foi definida na planilha de veículos.	Sigla de veículo não definida usada na publicação "<título>": <sigla>.
Código de docente especificado para uma publicação não foi definido na planilha de docentes.	Código de docente não definido usado na publicação "<título>": <sigla>.
Tipo de um veículo não é nem 'C' nem 'P'.	Tipo de veículo desconhecido para veículo <sigla>: <tipo>.
Sigla de veículo especificada para uma qualificação não foi definida na planilha de veículos.	Sigla de veículo não definida usada na qualificação do ano "<ano>": <sigla>.
Qualis especificado para uma qualificação de veículo não é nenhuma das categorias existentes: A1, A2, B1, B2, B3, B4, B5 ou C.	Qualis desconhecido para qualificação do veículo <sigla> no ano <ano>: <qualis>.
Qualis especificado para uma regra de pontuação não é nenhuma das categorias existentes: A1, A2, B1, B2, B3, B4, B5 ou C.	Qualis desconhecido para regras de <data início vigência>: <qualis>

Apesar de terminar sem produzir arquivos de saída, seu programa não deve ser interrompido abruptamente com uma chamada a `System.exit()`, mas sim seguir até o final do método `main()` e terminar normalmente.

Quaisquer outras situações de erro devem ser ignoradas. Pode-se assumir que nos testes feitos durante a avaliação dos trabalhos outros tipos de erros diferentes dos listados acima nunca acontecerão.

3. Execução

Seu programa deve ser executado especificando os nomes dos arquivos de entrada e o ano a ser considerado no recredenciamento como opções de linha de comando, especificadas a seguir:

- -d <arquivo>: planilha de docentes;
- -v <arquivo>: planilha de veículos;
- -p <arquivo>: planilha de publicações;
- -q <arquivo>: planilha de qualificações;
- -r <arquivo>: planilha de regras de pontuação;
- -a <ano>: ano de recredenciamento.

Supondo que a classe do seu programa que possui o método **`main()`** chama-se **`Main`** e encontra-se no pacote `trabalho`, para executar seu programa lendo os arquivos docentes.csv, veiculos.csv, publicacoes.csv, qualis.csv e regras.csv como arquivos de entrada e considere o ano de 2017 para o recredenciamento, o comando seria:

```
java trabalho.Main -d docentes.csv -v veiculos.csv -p
publicacoes.csv -q qualis.csv -r regras.csv -a 2017
```

Além dos parâmetros acima, a versão Java do seu programa deve suportar dois parâmetros opcionais que estabelecem três modos de execução diferentes. Neste caso, o programa deve poder ser chamado das três formas, como a seguir:

- **`java trabalho.Main -d docentes.csv -v veiculos.csv -p publicacoes.csv -q qualis.csv -r regras.csv -a 2017`**: quando não forem especificadas opções de execução, o programa deve ler os arquivos de entrada, gerar os relatórios e escrevê-los nos arquivos de saída, como descrito anteriormente;
- **`java trabalho.Main --read-only -d docentes.csv -v veiculos.csv -p publicacoes.csv -q qualis.csv -r regras.csv -a 2017`**: quando especificada a opção **`--read-only`**, o programa deve ler os arquivos de entrada, montar as estruturas de objetos em memória e serializar esta estrutura em um arquivo chamado **`recredenciamento.dat`**. Os relatórios não devem ser gerados neste caso;
- **`java trabalho.Main --write-only`**: quando especificada esta opção, o programa deve carregar os objetos serializados no arquivo **`recredenciamento.dat`**, gerar os relatórios e escrevê-los nos arquivos de saída. Neste caso não há leitura de arquivo CSV.

Importante: as opções de execução podem ser passadas em qualquer ordem. Portanto, o comando

```
java trabalho.Main --read-only -d docentes.csv -v veiculos.csv
-p publicacoes.csv -q qualis.csv -r regras.csv -a 2017
```

É equivalente a:

```
java trabalho.Main -q qualis.csv -r regras.csv -p
publicacoes.csv --read-only -a 2017 -v veiculos.csv -d
docentes.csv
```

4. Condições de entrega

O trabalho deve ser feito em dupla e deve ser entregue até o dia **03/12/2019**, ***impreterivelmente***.

Dado que existem várias versões dos compiladores Java, fica determinado o uso das versões instaladas nas máquinas dos LabGrads (Laboratórios de Graduação do Departamento de Informática) como versões de referência para o trabalho prático. Seu trabalho deve compilar e executar corretamente nas máquinas destes laboratórios. Além disso, os arquivos de código-fonte devem estar codificados com Unicode (UTF-8) para evitar erros de compilação.

4.1. Entrega do trabalho

O código-fonte e o arquivo de *build* (vide seção 4.2) de sua solução deverão ser compactados e enviados por e-mail (anexo ao e-mail) para a professora (jssalamon@inf.ufes.br). Serão aceitos trabalhos entregues até as 23h59 da data limite[2]. O assunto do e-mail deverá ser o seguinte:

`Prog3 – Trab2 - Nomes dos alunos`

substituindo `Nomes dos alunos` pelos nomes dos alunos do grupo, separado por vírgula.

Dada a quantidade de trabalhos que devem ser avaliados, a correção dos trabalhos passará primeiro por um processo de testes automáticos e, em seguida, por uma avaliação subjetiva (mais detalhes na próxima seção). Para que os testes automáticos funcionem, o arquivo compactado enviado por e-mail deve estar no formato **`zip`** com o nome **`trabalho.zip`** e conter o arquivo de *build* (explicações a seguir) e o código-fonte. O arquivo enviado não deve conter nenhuma classe compilada. Os testes automáticos serão executados no diretório onde encontra-se o arquivo de *build*. O código-fonte pode ser organizado da forma que a dupla achar melhor, desde que o arquivo de *build* esteja adequado a esta estrutura.

4.2. Preparação e execução do script de testes

O trabalho prático da disciplina será avaliado em duas etapas (vide seção 5), sendo a primeira uma avaliação objetiva, com testes automáticos. Foi disponibilizado aos alunos um script para execução de alguns testes automáticos, sendo possível, portanto, garantir que o trabalho passa nesses testes antes de submetê-lo.

---

[2] Além da entrega do trabalho, será feita uma entrevista com cada grupo, separadamente.

O script de teste funciona somente em ambientes Linux/MacOS e foi testado no LabGrad. Recomenda-se fortemente que os testes finais do seu trabalho sejam feitos no LabGrad, pois as versões das ferramentas instaladas nas máquinas do laboratório serão consideradas como versões de referência para a correção do trabalho.

Para obter e executar o script, siga os passos abaixo:

1. Na página da disciplina, obtenha o arquivo **`script-java.zip`**;
2. Descompacte o arquivo em alguma pasta do seu sistema;
3. Abra um terminal, acesse esta pasta;
4. Execute o script: **`./test.sh`** e o resultado deve ser parecido com a saída abaixo:

```
$ ./test.sh
Script de teste Prog3 2019/2 - Trabalho 2

$
```

O script reconhece trabalhos se forem colocados em pastas no mesmo diretório em que se encontra o script **`test.sh`** e se os trabalhos seguirem as instruções contidas a seguir. Note que há já uma pasta **`testes`**, que contém os arquivos de teste executados pelo script. O script já é configurado a não considerar esta pasta como uma pasta de trabalho.

No exemplo abaixo, o comando **`ls`** mostra que há um diretório professor dentro do qual encontra-se a implementação do trabalho. Em seguida, mostra a execução do script de testes sem erro algum:

```
$ ls -Flpah
drwxr-xr-x@  6 jordana  staff   204B May 29 17:09 ./
drwxr-xr-x@ 14 jordana  staff   476B May 29 17:09 ../
drwxr-xr-x@  5 jordana  staff   170B May 29 17:26 professor/
-rwxr-xr-x@  1 jordana  staff   2.0K May 29 17:12 test.sh
drwxr-xr-x@  6 jordana  staff   204B May 29 17:11 testes/

$ ./test.sh
Script de teste Prog3 2019/2 - Trabalho 2

[I] Testando professor...
[I] Testando professor: teste 01
[I] Testando professor: teste 01, tudo OK em 1-
recredenciamento.csv
[I] Testando professor: teste 01, tudo OK em 2-publicacoes.csv
[I] Testando professor: teste 01, tudo OK em 3-estatisticas.csv
[I] Testando professor: teste 02
[I] Testando professor: teste 02, serialização OK!
[I] Testando professor: teste 03
[I] Testando professor: teste 03, serialização OK!
[I] Testando professor: teste 03, tudo OK em 1-
recredenciamento.csv
[I] Testando professor: teste 03, tudo OK em 2-publicacoes.csv
[I] Testando professor: teste 03, tudo OK em 3-estatisticas.csv
```

```
[I] Testando professor: pronto!

$
```

O script compara cada arquivo de saída gerado pelo trabalho do aluno com o arquivo de saída gerado pela implementação do professor. No exemplo acima, nenhum erro foi encontrado e tudo está OK. Quando diferenças são encontradas, as mesmas são mostradas na tela e implicam perda de pontos na correção automática. O aluno deve, portanto, tentar reduzir o número de diferenças ao máximo possível antes de entregar o trabalho.

Nota: alguns testes podem indicar tudo OK no arquivo output.txt, porém este arquivo não foi citado na especificação. Na verdade, este é um arquivo temporário criado pelo script para os casos em que há inconsistência no trabalho e o programa deve imprimir uma mensagem na tela. O script direciona as impressões de tela para este arquivo temporário e compara com a resposta oficial do professor.

Para testar o seu trabalho, crie uma pasta com um nome qualquer dentro do mesmo diretório em que se encontra o script **`test.sh`** e copie seu código-fonte para esta pasta. Além do código-fonte, crie um arquivo de *build* do Apache Ant (http://ant.apache.org) que indique como compilar e executar seu programa.

Os arquivos fonte podem estar organizados da forma que você achar melhor, desde que o Ant consiga compilá-los, executar as classes geradas e limpar o projeto. Para que isso seja feito de forma automatizada, o arquivo de *build* do Ant deve, obrigatoriamente, encontrar-se na raiz da pasta criada e chamar-se **`build.xml`**. Além disso, ele deve ser feito de forma a responder aos seguintes comandos:

Comando	Resultado esperado
`ant compile`	O código-fonte deve ser compilado, gerando os arquivos .class para todas as classes do trabalho.
`ant run`	O programa deve ser executado especificando as opções **`-d docentes.csv -v veiculos.csv -p publicacoes.csv -q qualis.csv -r regras.csv -a 2017`** como parâmetro.
`ant run-read-only`	O programa deve ser executado no modo **`--read-only`** (vide seção 3), especificando além disso as mesmas opções do comando **`ant run`** acima.
`ant run-write-only`	O programa deve ser executado no modo **`--write-only`** (vide seção 3).
`ant clean`	Todos os arquivos gerados (classes compiladas, relatórios de saída, arquivo de serialização) e eventuais arquivos de entrada de dados devem ser excluídos, sobrando somente o conteúdo original do arquivo compactado (ou seja, o código-fonte e o arquivo de *build*).

Caso você não implemente as opções *read-only* e *write-only*, faça com que os respectivos comandos funcionem da mesma forma que o comando *run*, ou seja, efetuem a execução normal.

Segue abaixo um exemplo de arquivo **build.xml** que atende às especificações acima. Em negrito encontram-se marcados os dados que devem ser adaptados dependendo do projeto:

- **src**: subpasta onde encontra-se todo o código-fonte;
- **bin**: subpasta onde serão colocadas as classes compiladas;
- **meupacote.MinhaClassePrincipal**: nome da classe principal do programa, ou seja, aquela que possui o método **main()**.

```
<project name="TrabalhoProg3_2019_2" default="compile" basedir=".">
    <description>Arquivo de build do trabalho de Prog3, 2019/2.</description>

    <!-- Propriedades do build. -->
    <property name="src" location="src" />
    <property name="bin" location="bin" />
    <property name="mainClass" value="meupacote.MinhaClassePrincipal" />

    <!-- Inicialização. -->
    <target name="init" description="Inicializa as estruturas necessárias.">
        <tstamp/>
        <mkdir dir="${bin}" />
    </target>

    <!-- Compilação. -->
    <target name="compile" depends="init" description="Compila o código-
fonte.">
        <javac includeantruntime="false" srcdir="${src}" destdir="${bin}" />
    </target>

    <!-- Execução normal. -->
    <target name="run" depends="compile" description="Executa o programa
principal, em modo normal.">
        <java classname="${mainClass}">
            <arg value="-d" />
            <arg value="docentes.csv" />
            <arg value="-v" />
            <arg value="veiculos.csv" />
            <arg value="-p" />
            <arg value="publicacoes.csv" />
            <arg value="-q" />
            <arg value="qualis.csv" />
            <arg value="-r" />
            <arg value="regras.csv" />
            <arg value="-a" />
            <arg value="2017" />
            <classpath>
                <pathelement path="${bin}" />
            </classpath>
        </java>
    </target>

    <!-- Execução somente leitura. -->
    <target name="run-read-only" depends="compile" description="Executa o
programa principal, em modo somente leitura.">
```

```
          <java classname="${mainClass}">
               <arg value="-d" />
               <arg value="docentes.csv" />
               <arg value="-v" />
               <arg value="veiculos.csv" />
               <arg value="-p" />
               <arg value="publicacoes.csv" />
               <arg value="-q" />
               <arg value="qualis.csv" />
               <arg value="-r" />
               <arg value="regras.csv" />
               <arg value="-a" />
               <arg value="2017" />
               <arg value="--read-only" />
               <classpath>
                    <pathelement path="${bin}" />
               </classpath>
          </java>
     </target>

     <!-- Execução somente escrita. -->
     <target name="run-write-only" depends="compile" description="Executa o
programa principal, em modo somente escrita.">
          <java classname="${mainClass}">
               <arg value="--write-only" />
               <classpath>
                    <pathelement path="${bin}" />
               </classpath>
          </java>
     </target>

     <!-- Limpeza. -->
     <target name="clean" description="Limpa o projeto, deixando apenas o
código-fonte." >
          <delete dir="${bin}"/>
          <delete><fileset dir="." includes="*.txt"/></delete>
          <delete><fileset dir="." includes="*.csv"/></delete>
          <delete><fileset dir="." includes="*.dat"/></delete>
     </target>
</project>
```

O grupo deve preparar e enviar à professora dois conjuntos de arquivos de entrada (docentes.csv, veiculos.csv, publicacoes.csv, qualis.csv e regras.csv) que atendam aos seguintes critérios:

- Não conter trechos iguais a outros arquivos de teste disponíveis no site (ou seja, não copiar);
- Conter o cadastro de pelo menos 5 docentes, 20 veículos, 30 publicações espalhadas pelos anos 2013 a 2016 (ao menos 10 delas devem ter mais de um autor), qualificação dos veículos em 3 anos diferentes e 2 regras de pontuação: uma para 2017 e outra para 2018;
- Assim como o código-fonte do trabalho, os arquivos devem estar em formato Unicode (UTF-8);

- Os dois conjuntos de arquivos devem ser quase iguais: um deles não deve ter inconsistência nenhuma enquanto o outro deve apresentar 1 inconsistência de dados (a escolha do grupo), como descrito na seção 0.

Os arquivos de teste enviados poderão, a critério da professora, ser disponibilizados aos demais alunos como parte do script de testes. Atualizações do script serão divulgadas em sala de aula.

5. Critérios de avaliação

Os trabalhos serão avaliados em duas etapas:

- Avaliação objetiva (com testes automáticos), valendo 10 pontos;
- Avaliação subjetiva (entrevistas), valendo 10 pontos.

A nota final do trabalho é a média aritmética simples entre as notas acima. Para a avaliação objetiva, todo trabalho possui inicialmente nota 10 e sofre modificações nas situações descritas na tabela abaixo:

Situação	Modificação
Não observou as regras para envio do trabalho.	-1
Não compilou nos testes automáticos, mas foi possível compilar manualmente (ex.: arquivos não codificados em UTF-8).	-3
Não compilou nem manualmente.	-10
Não gerou saídas nos testes automáticos.	-5
Não gerou saídas nem manualmente.	-8
Pequenas diferenças das saídas em relação ao teste automático (ex.: formatação, arredondamentos, etc.).	-1
Grandes diferenças das saídas em relação ao teste automático (ex.: valores sensivelmente diferentes).	-2
Entrega fora do prazo.	-1 por dia

Para avaliação subjetiva, novamente os trabalhos começam com nota 10 e perdem pontos (que variam de acordo com a avaliação feita pelo professor) caso não estejam bem escritos ou organizados. Critérios utilizados na avaliação subjetiva incluem (mas não estão limitados a):

- Uso dos princípios básicos da orientação a objetos, como encapsulamento, abstração e modularização;
- Legibilidade (nomes de variáveis bem escolhidos, código bem formatado, uso de comentários quando necessário, etc.);
- Consistência (utilização de um mesmo padrão de código, sugere-se a convenção de código do Java: http://www.oracle.com/technetwork/java/codeconv-138413.html);
- Eficiência (sem exageros, tentar evitar grandes desperdícios de recursos);

- Uso eficaz da API Java (leitura com Scanner, API de coleções, etc.) e das funcionalidades das novas versões da plataforma (ex.: tipos genéricos, laço *for-each*, *try* com recursos fecháveis, etc.);
- Se o aluno sabe responder questões formuladas durante a entrevista sobre o código-fonte escrito pela dupla.

6. Pontos extra[3]

Para incentivar alunos que desejarem aprender conteúdos avançados da linguagem Java por conta própria,[4] são oferecidos pontos extra para os alunos que demonstrarem na entrevista que adicionaram uma ou mais das seguintes funcionalidades ao programa:

Funcionalidade opcional	Pontos extra
Implementação de uma interface Web que permita fazer o upload dos arquivos de entrada, gere os relatórios e os disponibilize para download. Nota: Applets não serão consideradas interfaces Web.	Até 3 pontos
Substituir a serialização descrita na seção 3 por armazenamento em banco de dados relacional. Podem ser utilizadas soluções de mapeamento objeto/relacional, se desejado.	Até 3 pontos

A coluna "Pontos extra" indica o máximo de pontos extra que podem ser obtidos pela implementação da funcionalidade extra correspondente. A pontuação exata será estabelecida após avaliada a qualidade do código, que deve ser explicado pelos alunos, e do resultado (ex.: quão bem feita é a interface gráfica/web?).

Note que o trabalho Java para correção automática deve ser enviado no prazo do trabalho. Posteriormente ele pode ser utilizado como base para criação do programa com interface gráfica/Web ou banco de dados para obtenção dos pontos extra, porém ao enviá-lo para a correção no prazo inicial ele deve responder aos scripts dos testes automáticos, do contrário sofrerá as penalidades descritas na especificação da correção objetiva.

7. Observações finais

Caso haja algum erro neste documento, serão publicadas novas versões e divulgadas erratas em sala de aula. É responsabilidade do aluno manter-se informado, frequentando as aulas ou acompanhando as novidades na página da disciplina na Internet.

---

[3] Ao contrário da recuperação de nota, os pontos extras permitem que a nota do trabalho Java ultrapasse o valor máximo de 10 pontos. No entanto, no cálculo da média parcial do aluno, a nota máxima continua sendo 10, não podendo ser ultrapassada.

[4] Os alunos devem buscar recursos próprios para aprender tais tecnologias, visto que não há tempo hábil durante o semestre para aulas destes assuntos.