Especificação do Trabalho Prático

O trabalho prático da disciplina consiste em desenvolver o mesmo sistema computacional para solução do problema descrito abaixo nas duas linguagens de programação apresentadas durante o curso: Java e C++.

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

1. Descrição do problema

Com a vastidão de formas de entretenimento que temos hoje, é complicado manter em mente o que já se viu e o que ainda se quer ver, bem como lembrar quem está com sua cópia do livro "O Senhor dos Anéis", seu Blu-ray do último filme do Tarantino ou seu DVD da última temporada de "Game of Thrones".

Propomos um sistema capaz de inventariar as "mídias" de entretenimento – considerando livros, filmes ou séries de TV[1] – com as quais o usuário tem algum tipo de relação, dentre as quais: (a) possui uma cópia; (b) já "consumiu" a mídia (ou seja, leu o livro ou assistiu o filme / a temporada da série), independente de possuir ou não uma cópia; (c) deseja consumir a mídia (também independente de ter ou não uma cópia).

Além da relação do usuário com a mídia e o nome da mesma, deseja-se registrar as fichas técnicas de cada uma, a saber:

- Para livros: nomes dos autores, número de páginas e gênero;
- Para filmes: diretor (considere que há somente um), atores principais (cujo registro é opcional), gênero e duração;
- Para temporadas de séries: atores principais (também opcional), gênero, duração, número da temporada e nome da série.

O gênero é a categoria usada pelo mercado para classificação de filmes, séries, livros e outras mídias, por exemplo: ação, comédia, ficção científica, etc. Quando mais de um gênero é aplicável (ex.: ação e comédia), o sistema se limitará a registrar apenas um (considerado o principal).

Ao marcar uma mídia como emprestada, o sistema deve permitir registrar a data de empréstimo, uma data prevista para devolução e para quem a mídia foi emprestada. O sistema, então, ajudaria a controlar não só o que está emprestado, mas, dentre estes, o que já deveria ter sido devolvido e a quem devemos cobrar a devolução.

Por fim, como algumas mídias são desejadas e ainda não se possui uma cópia, deseja-se registrar o preço das mesmas.

Com tais informações, espera-se que o sistema seja capaz de gerar para o usuário alguns relatórios úteis como, por exemplo:

- Listagem das mídias emprestadas, indicando com quem está, se já deveriam ter sido devolvidas e, neste caso, quantos dias de atraso acumulam;

[1] Para séries de TV, considera-se uma temporada inteira da série, pois geralmente as mídias (DVDs, Blu-Rays) são vendidas desta maneira. O sistema não deve registrar se o usuário assistiu um ou outro episódio de uma série, mas sim temporadas inteiras.

- Listagem das mídias por autor/diretor/ator: para cada nome (destas pessoas), indicar as mídias relacionadas a elas (ex.: filmes/séries em que Selton Mello participa como ator ou diretor);
- *Wishlist*, ou seja, lista de mídias que se deseja consumir, mas que ainda não se possui, e seus respectivos preços, dando ideias de possíveis presentes que o usuário deseja ganhar no futuro;
- Estatísticas gerais:
  - Estimativas de tempo[2]: horas gastas com mídias já consumidas, horas necessárias para consumir os demais itens;
  - Total de mídias por tipo e por gênero;
  - Quantidade de temporadas de cada série que se possui e que já se consumiu.

As próximas seções do documento detalham os dados de entrada e relatórios de saída esperados.

2. Formatos de entrada e saída

Os cadastros dos dados do inventário 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.: 3,9), os dados serão exportados utilizando ponto-e-vírgula como separadores (ex.: O Senhor dos Anéis: o Retorno do Rei;L;32,54 – representando um livro da série "O Senhor dos Anéis" que custa R$ 32,54).

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 quatro os arquivos de entrada de dados:

- Cadastro de gêneros;

[2] Apenas para mídias que possuam indicação de duração, como filmes e séries.

- Cadastro de pessoas;
- Inventário de mídias;
- Controle de empréstimos.

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:

Cadastro de gêneros

```
<sigla>;<nome>
```

Ambos podem ser lidos como texto (string).

Cadastro de pessoas

```
<código>;<nome>
```

O código é numérico (inteiro), nome pode ser lido como texto.

Inventário de mídias

```
<código>;<nome>;<tipo>;<diretor>;<a(u)tores>;<tamanho>;<gênero>;
<série>;<temporada>;<possui?>;<consumiu?>;<deseja?>;<preço>
```

Cada campo é preenchido como se segue:

- **Código**: numérico (inteiro);
- **Nome**: texto;
- **Tipo**: um caractere – 'L' para livros, 'F' para filmes e 'S' para temporadas de séries;
- **Diretor**: código da pessoa (vide cadastro de pessoas) que dirigiu o filme. No caso de livros e séries, ficará vazio;
- **A(u)tores**: códigos das pessoas (vide cadastro de pessoas) que são atores principais no filme ou na série, ou são autores do livro. Os códigos são separados por vírgula. Podem estar em branco no caso de atores (mas autores são obrigatórios);
- **Tamanho**: numérico (inteiro), indicando número de páginas para livros e duração em minutos para filmes e séries;
- **Gênero**: sigla (vide cadastro de gêneros) referente ao gênero principal da mídia;
- **Série**: nome da série. No caso de livros e filmes, ficará vazio;
- **Temporada**: número da temporada. No caso de livros e filmes, ficará vazio;
- **Possui?**: o caractere 'x' caso o usuário possua a mídia, vazio caso contrário;
- **Consumiu?**: o caractere 'x' caso o usuário já tenha consumido a mídia, vazio caso contrário;
- **Deseja?**: o caractere 'x' caso o usuário deseje consumir a mídia, vazio caso contrário;
- **Preço:** numérico (real).

Controle de empréstimos

```
<mídia>;<tomador>;<empréstimo>;<devolução>
```

Mídia é numérico (inteiro) e se refere ao código da mídia (vide inventário de mídias). Tomador pode ser lido como texto e se refere a quem tomou a mídia emprestada. Empréstimo e devolução são datas, no formato dd/mm/yyyy.

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
Estatísticas (não-CSV)	`1-estatisticas.txt`
Mídias por pessoa	`2-porpessoa.csv`
Empréstimos	`3-emprestimos.csv`
Wishlist	`4-wishlist.csv`

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

Estatísticas (não-CSV)

```
Horas consumidas: <HC> minutos
Horas a consumir: <HA> minutos

Mídias por gênero:
    <G>: <QG>

Temporadas por série:
    <S>: <QSC> assistidas, <QSA> a assistir
```

Este relatório é o único que não possui o formato CSV, mas texto simples. Os códigos acima devem ser substituídos pelos seguintes dados:

- <HC>: soma das durações das mídias já consumidas (exceto livros);
- <HA>: soma das durações das mídias que se deseja consumir (exceto livros);
- <G>: nome do gênero;
- <QG>: quantidade de mídias deste gênero no inventário;
    - Deve haver uma linha "<G>: <QG>" para cada gênero presente no cadastro de gêneros;
    - As linhas devem estar ordenadas primeiro pela quantidade (decrescente), em seguida pelo nome do gênero (alfabeticamente, crescente);
- <S>: nome da série;
- <QSC>: quantidade de temporadas desta série que já foram consumidas;
- <QSA>: quantidade de temporadas desta série que se deseja consumir;

- Deve haver uma linha "<S>: <QSC> assistidas, <QSA> a assistir" para cada série presente no inventário de mídias;
- Devem estar ordenadas pelo nome da série (alfabeticamente, crescente).

Mídias por pessoa

```
<nome da pessoa>;<nomes das mídias>
```

Este relatório deve ser ordenado por nome da pessoa, em ordem crescente. Caso alguma pessoa tenha participação (como ator, diretor ou autor) em mais de uma mídia, a coluna de nomes das mídias deve separar os nomes com vírgula e espaço (", "), apresentando-os também em ordem alfabética. No relatório devem constar apenas pessoas que estejam associadas a ao menos uma mídia.

Empréstimos

```
<data do empréstimo>;<tomador>;<está atrasado?>;<quantidade de
dias em atraso>
```

Este relatório deve ser ordenado por data do empréstimo, em ordem decrescente. Em caso de empate, deve-se ordenar pelo nome do tomador em ordem crescente. A coluna "está atrasado?" deve contar a palavra "Sim" ou a palavra "Não".

De modo a permitir uma correção automática dos trabalhos (vide seção 4), a coluna "quantidade de dias em atraso" deve considerar a data de 06/11/2015 como data de hoje (ao invés da data atual). Desta maneira, espera-se que todos os trabalhos produzam o mesmo relatório, independente de quando forem executados. Quando o empréstimo não estiver atrasado, na coluna deve constar o valor 0 (zero).

Wishlist

```
<tipo>;<nome da mídia>;<gênero>;<preço>
```

Este relatório deve ser ordenado primeiro pelo tipo da mídia (crescente), seguido pelo preço (decrescente) e, em caso de empate, pelo nome da mídia (crescente). Tipo e gênero devem ser escritos por extenso (e não seus códigos/siglas). O preço deve ser formatado com duas casas decimais e prefixado com "R$ ".

2.4. Tratamento de exceções

Leitura de dados de arquivos, formatação, etc. são fontes comuns de erros e exceções. 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);
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);
3. Inconsistência nos dados, ou seja, não conformidade com a especificação dos arquivos de entrada da seção 2.1. Exemplos: usar um código de pessoa no arquivo de mídias (diretor, ator ou autor) e não existir uma pessoa com esse código no arquivo de pessoas; colocar algo diferente de "x" na coluna "Possui?" do cadastro

de mídias; etc. Nestes casos, o programa deve exibir a mensagem "Dados inconsistentes (<coluna>: <valor>)" (sem as aspas).

a. <coluna> deve trazer o nome da coluna, enquanto <valor> deve trazer o valor inválido que foi colocado na mesma. Por exemplo, se foi o usado o código 123 na coluna Diretor mas esta pessoa não existe, o programa deve imprimir "Dados inconsistentes (Diretor: 123)" (sem as aspas).

As mensagens acima devem ser impressas na tela e o programa deve ser terminado em seguida. No entanto, 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 como opções de linha de comando, especificadas a seguir:

- -g <arquivo>: cadastro de gêneros;
- -p <arquivo>: cadastro de pessoas;
- -m <arquivo>: inventário de mídias;
- -e <arquivo>: controle de empréstimos.

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 generos.csv, pessoas.csv, midias.csv e emprestimos.csv como arquivos de entrada, o comando seria:

```
java trabalho.Main -g generos.csv -p pessoas.csv -m midias.csv
-e emprestimos.csv
```

Recuperação de pontos[3]: além dos parâmetros acima, a versão Java do seu programa pode, se quiser recuperar pontos perdidos (vide Seção 5), 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 -g generos.csv -p pessoas.csv -m midias.csv -e emprestimos.csv`: 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 -g generos.csv -p pessoas.csv -m midias.csv -e emprestimos.csv`: quando especificada a opção `--read-only`, o programa deve ler os arquivos de entrada, montar as estruturas de

[3] Diferentemente dos pontos extras (vide Seção 6), a recuperação de pontos apenas recupera pontos perdidos, o que significa que a nota do trabalho não poderá ultrapassar o valor máximo de 10 pontos.

objetos em memória e serializar esta estrutura em um arquivo chamado `inventario.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 `inventario.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 -g generos.csv -p pessoas.csv -
m midias.csv -e emprestimos.csv
```

É equivalente a:

```
java trabalho.Main -e emprestimos.csv -p pessoas.csv -m
midias.csv --read-only -g generos.csv
```

Por fim, a versão C++ do programa não deve implementar as opções `--read-only` e `--write-only`, ou seja, a recuperação de pontos não é oferecida para o trabalho 2.

4. Condições de entrega

O trabalho deve ser feito obrigatoriamente em dupla[4] e em duas versões: uma utilizando a linguagem Java, outra utilizando a linguagem C++. O primeiro deve ser entregue até o dia **06/11/2015** e o segundo até o dia **04/12/2015**, impreterivelmente. As duplas para os trabalhos Java e C++ não precisam, necessariamente, ser as mesmas. No entanto, é preciso avisar com antecedência sobre eventuais trocas.

Dado que existem várias versões dos compiladores Java e C++, fica determinado o uso das versões instaladas nas máquinas do LCEE (Laboratório de Computadores da Engenharia Elétrica) como versões de referência para o trabalho prático. Seu trabalho deve compilar e executar corretamente nas máquinas deste laboratório. 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 o monitor da disciplina (edox86@gmail.com) e com cópia para o professor (vitor.souza@ufes.br). Serão aceitos trabalhos entregues até as 23h59 da data limite[5]. O assunto do e-mail deverá ser o seguinte:

---

[4] Exceto quando permitido extraordinariamente pelo professor, acordado com antecedência. Caso contrário, o aluno/grupo sofrerá uma penalidade de 2 pontos.

[5] Além da entrega do trabalho, será feita uma entrevista com cada grupo, separadamente. Tais entrevistas devem acontecer até a data de entrega do trabalho em questão, em horário de atendimento do professor. Os alunos podem enviar o trabalho depois da entrevista (até as 23h59 da data limite), porém devem trazer uma cópia do trabalho para apresentação no momento da entrevista.

`PAC - Trab1 - Nomes dos alunos`

substituindo `Nomes dos alunos` pelos nomes dos alunos do grupo, separado por vírgula. Para a entrega do trabalho de C++, substitua `Trab1` por `Trab2`.

Assim que possível, o monitor responderá o e-mail com um *hash* MD5[6] do arquivo recebido. Para garantir que o arquivo foi recebido sem ser corrompido, gere o *hash* MD5 do arquivo que você enviou[7] e compare com o *hash* recebido na confirmação. Caso você não receba o e-mail de confirmação ou caso o valor do *hash* seja diferente, envie o trabalho novamente. Se o problema persistir, contate o professor o mais rápido possível.

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

Os trabalhos práticos da disciplina serão avaliados 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 ao professor.

O script de teste funciona somente em ambientes Linux/MacOS e foi testado no LCEE. Recomenda-se fortemente que os testes finais do seu trabalho sejam feitos no LCEE, 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 teaching-br-pac-20152-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 PAC 2015/2 - Trabalho 1
```

[6] Para saber mais sobre *hash* MD5, visite sua página na Wikipedia: https://pt.wikipedia.org/wiki/MD5#Hashes_MD5

[7] Para gerar *hash* MD5 de arquivos no Linux, veja as instruções em http://roneymedice.com.br/2009/07/30/gerando-hash-md5-dos-arquivos-no-linux/. Já na página http://www.mundodoshackers.com.br/como-gerar-e-checar-hashs-md5 você encontra instruções também para Windows (porém os exemplos mostram geração de *hash* para strings, não para arquivos).

```
$
```

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 do professor. Em seguida, mostra a execução do script de testes sem erro algum:

```
$ ls -Flpah
drwxr-xr-x   6 vitor  staff   204B Oct 13 15:31 ./
drwxr-xr-x@ 16 vitor  staff   544B Oct 13 15:01 ../
drwxr-xr-x   4 vitor  staff   136B Oct 13 15:26 professor/
-rwxr-xr-x@  1 vitor  staff   2.0K Oct  9 08:21 test.sh
drwxr-xr-x   7 vitor  staff   238B Oct 13 15:31 testes/

$ ./test.sh
Script de teste PAC 2015/2 - Trabalho 1

[I] Testando professor...
[I] Testando professor: teste 01
[I] Testando professor: teste 01, tudo OK em 1-estatisticas.txt
[I] Testando professor: teste 01, tudo OK em 2-porpessoa.csv
[I] Testando professor: teste 01, tudo OK em 3-emprestimos.csv
[I] Testando professor: teste 01, tudo OK em 4-wishlist.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-estatisticas.txt
[I] Testando professor: teste 03, tudo OK em 2-porpessoa.csv
[I] Testando professor: teste 03, tudo OK em 3-emprestimos.csv
[I] Testando professor: teste 03, tudo OK em 4-wishlist.csv
[I] Testando professor: teste 04
[I] Testando professor: teste 04, tudo OK em output.txt
[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.

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 `-g generos.csv -p pessoas.csv -m midias.csv -e emprestimos.csv` 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="TrabalhoPAC_2015_2" default="compile" basedir=".">
    <description>Arquivo de build do trabalho de PAC, 2015/2.</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="${mainClass}">
            <arg value="-g" />
            <arg value="generos.csv" />
            <arg value="-p" />
            <arg value="pessoas.csv" />
            <arg value="-m" />
            <arg value="midias.csv" />
            <arg value="-e" />
            <arg value="emprestimos.csv" />
            <classpath>
                <pathelement path="${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="${mainClass}">
            <arg value="-g" />
            <arg value="generos.csv" />
            <arg value="-p" />
            <arg value="pessoas.csv" />
            <arg value="--read-only" />
            <arg value="-m" />
            <arg value="midias.csv" />
            <arg value="-e" />
            <arg value="emprestimos.csv" />
            <classpath>
                <pathelement path="${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>
```

Recuperação de pontos[8]: será dado 1 ponto extra ao grupo que preparar e enviar ao professor, até o prazo do trabalho 1, dois conjuntos de arquivos de entrada (generos.csv, pessoas.csv, midias.csv e emprestimos.csv) que atendam aos seguintes critérios:

- Não conter trechos iguais a outros arquivos de teste disponíveis no site;
- Conter o cadastro de pelo menos 10 gêneros, 30 pessoas, 20 mídias e 5 empréstimos;
- 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 2 ou 3 inconsistências de dados (a escolha do grupo), como descrito na seção 2.4.

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

4.3. Apresentação do trabalho em entrevista

Os trabalhos devem ser também apresentados ao professor por meio de entrevista. Para tal, os alunos devem acessar o sistema de marcação de horários YouCanBook.Me no seguinte endereço:

https://vitorsouza.youcanbook.me

Na tabela de horários que se apresenta, cada dupla deve agendar um horário, dentre os horários disponíveis, até as respecticas datas limite, com duração de 30 minutos e fornecendo os dados solicitados pelo formulário (nome e e-mail). Para o propósito da reunião, selecionar a opção "Aluno(a) de Programação Aplicada de Computadores".

Atenção aos seguintes detalhes sobre o agendamento:

- O sistema só permite agendamentos com antecedência mínima de 8 horas (e máxima de 2 semanas);
- O sistema bloqueia automaticamente horários já reservados ou em que o professor tenha outros compromissos;

[8] Idem seção 3.

- É de responsabilidade do aluno achar um horário disponível. Planeje-se com antecedência para evitar problemas de última hora (ex.: falta de horários adequados).

Uma vez agendada a reunião, os alunos devem comparecer à sala do professor (Ufes, CT-7, térreo, sala 17) para a entrevista pontualmente no dia e hora marcados. A apresentação do trabalho pode ser feita em computador portátil trazido pelos alunos ou no computador do professor. Em qualquer caso, os alunos devem também trazer o código-fonte do trabalho em disco removível (*pen-drive*) para o professor (ou tê-lo enviado por e-mail anteriormente).

A entrevista consiste em uma apresentação do código do trabalho feita pelos alunos. Durante esta apresentação, os alunos serão questionados **individualmente** sobre detalhes do trabalho e serão avaliados com relação às respostas fornecidas. Os critérios de avaliação são descritos na seção a seguir.

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 foi feito em dupla (exceto casos previamente autorizados)	-2
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

Situação	Modificação
Opções `--read-only` e `--write-only` funcionaram no teste automático.	+2

Para avaliação subjetiva, novamente os trabalhos começam com nota 10 e perdem pontos (que variam de acordo com a avaliação feita pelo professor e pelo monitor) 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 pelo professor durante a entrevista sobre o código-fonte escrito pela dupla.

6. Pontos extra[9]

Para incentivar alunos que desejarem aprender conteúdos avançados da linguagem Java por conta própria[10], são oferecidos pontos extra para os alunos que demonstrarem na entrevista do trabalho 2 que adicionaram uma ou mais das seguintes funcionalidades ao programa do trabalho 1:

Funcionalidade opcional	Pontos extra
Implementação de uma interface gráfica (janelas) que permita indicar onde encontram-se os arquivos de entrada e onde salvar os arquivos de saída e gerar os relatórios a partir de um clique.	Até 3 pontos
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

[9] 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.

[10] Os alunos devem buscar recursos próprios para aprender tais tecnologias, visto que não há tempo hábil durante o semestre para aulas destes assuntos. O professor, no entanto, possui alguns materiais que pode indicar para os alunos interessados.

Substituir a serialização descrita na seção 3 por armazenamento em banco de dados relacional. Podem ser utilizadas soluções de mapeamento objeto/relacional.	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 pelo professor 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 1. 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.

Esta opção não é oferecida para os trabalhos C++.

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.