Universidade Federal do Espírito Santo
Centro Tecnológico
Departamento de Informática

Prof. João Paulo A. Almeida (adaptado do Prof. Vítor Silva Souza)

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++.

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. Assim 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 (opcional), gênero, duração total da temporada, 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 uma data de 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-

- 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 cabeçalho 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:

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

- Cadastro de gêneros;
- 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. Data de empréstimo e data prevista de devolução são datas, no formato dd/mm/yyyy.

2.2. Processamento e Escrita dos Relatórios

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.

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-pessoas.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, 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$ ".

2.3. 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. Inconsistências nos dados de entrada: por exemplo, uso de código de mídia no arquivo de empréstimos que não corresponda a código de mídia no arquivo de mídias; uso de código de gênero ou pessoa inexistente nos respectivos cadastros, 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).
4. 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 possíveis 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
```

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 -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

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`.

4. Condições de entrega

O trabalho deve ser feito obrigatoriamente em dupla[3] 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 Labgrad 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 professor (jpalmeida@inf.ufes.br). Serão aceitos trabalhos entregues até as 23h59 da data limite. O assunto do e-mail deverá ser o seguinte:

`PROG3 - 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`.

[3] Exceto quando permitido extraordinariamente pelo professor, acordado com antecedência.

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. 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 .zip deverá também incluir 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 5 gêneros, 30 pessoas, 20 mídias e 10 empréstimos;

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.

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.

Para que o *build* 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*).

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_2015_2" default="compile" basedir=".">
    <description>Arquivo de build do trabalho de PROG3, 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>
```

4.2. Preparação e execução do script de testes

Será 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 funcionará somente em ambientes Linux/MacOS e será 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.

5. Critérios de avaliação

Os trabalhos serão avaliados em duas etapas:

- Avaliação objetiva (com testes automáticos), e
- Avaliação subjetiva.

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.);

6. 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.