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

A escola de esportes **UFESports** oferece aulas de futebol, basquetebol, voleibol e tênis. Atualmente controla os alunos inscritos, as turmas oferecidas e os professores alocados em um caderno, e deseja um sistema automatizado para processamento das inscrições e geração de relatórios.

1.1. As turmas

A escola oferece aulas de quatro modalidades esportivas: futebol, basquetebol, voleibol e tênis. As três primeiras são dadas a turmas de alunos, enquanto a última é individual. Todas as aulas são semanais e tem duração de uma hora. Seu sistema deve construir classes que represente estas aulas, sendo que de cada turma deseja-se saber:

- Aulas de futebol: dia da semana e horário da aula, professor, faixa etária, número mínimo e número máximo de alunos;
- Aulas de basquetebol: dia da semana e horário da aula, professor, número mínimo e número máximo de alunos;
- Aulas de voleibol: dia da semana e horário da aula, professor, estatura máxima para inscrição na turma, número mínimo e número máximo de alunos;
- Aulas de tênis: dia da semana e horário da aula, professor e se o aluno treina para lazer ou para campeonato.

Para a leitura dos dados, cada aula possui ainda um identificador numérico único.

1.2. Alunos e professores

Professores são cadastrados no sistema e associados às turmas que dão aula. De um professor deseja-se registrar: nome, CPF, telefone e e-mail, sendo este último opcional.

Alunos cadastram-se no sistema e em seguida inscrevem-se em aulas disponíveis. De um aluno deseja-se saber o mesmo que o professor: nome, CPF, telefone e e-mail, sendo este último opcional. Professores e alunos possuem também um identificador numérico único para efeito de leitura de dados.

1.3. Relatórios

Após registrados todos os alunos, professores e aulas disponíveis, é preciso processar as inscrições. Durante este processamento, alguns problemas podem ocorrer:

- Algumas turmas podem não ter o mínimo de alunos necessários e, portanto, serem canceladas. Todas as inscrições para estas turmas devem também ser canceladas;
- Algumas turmas podem lotar e as inscrições que chegarem após sua lotação devem ser canceladas. Aulas de tênis são individuais e, portanto, somente a primeira inscrição será confirmada;

Pede-se que o programa gere os seguintes relatórios:

1. Relatório de aulas confirmadas;
2. Relatório de aulas canceladas;
3. Relatório de inscrições canceladas.

Tais relatórios são utilizados pelos funcionários da **UFESports** para contatar alunos e professores para confirmar ou cancelar as aulas. Na próxima seção serão detalhados os formatos dos arquivos de entrada e saída de dados.

2. Formatos de entrada e saída

Na **UFESports**, todos os cadastros são feitos usando planilhas eletrônicas. Para integração com o sistema, as planilhas serão convertidas em texto puro usando o formato CSV (*Comma Separated Values* – Valores Separados por Vírgulas) para leitura pelo seu programa. Da mesma forma, os relatórios gerados pelo seu programa devem também seguir o padrão CSV para que possam ser lidos pelo software de planilha eletrônica, facilitando o trabalho dos funcionários da escola.

Sendo assim, as próximas seções explicam como são formatados os arquivos de entrada e saída de dados. Muita atenção a estes formatos, pois eles tem papel fundamental na correção do software.

2.1. Entrada de dados

São quatro os arquivos de entrada de dados:

- `professores.csv`: contém o cadastro dos professores;
- `alunos.csv`: contém o cadastro dos alunos;
- `aulas.csv`: contém as aulas disponíveis para o próximo período;
- `inscricoes.csv`: contém as inscrições dos alunos.

Cadastro de professores e alunos

Os arquivos `professores.csv` e `alunos.csv` possuem, respectivamente, os cadastros de professores e alunos. Cada linha do arquivo representa um cadastro e os dados são dispostos no seguinte formato:

`<id>,<nome>,<cpf>,<telefone>,<email>`

Dado	Significado	Formato
`id`	Identificador único	Número inteiro
`nome`	Nome completo	String
`cpf`	Número do CPF	String no formato `###.###.###-##` (14 caracteres)
`telefone`	Telefone de contato	String
`email`	E-mail de contato	String

Aulas disponíveis

As aulas disponíveis são registradas no arquivo `aulas.csv`. Cada linha do arquivo representa uma aula, sendo que a primeira informação indica qual é a modalidade esportiva daquela aula (`F`, `B`, `V` ou `T`). As demais informações dependem do tipo da aula, como mostrado abaixo:

```
F,<id>,<dia>,<horário>,<id-prof>,<mínimo>,<máximo>,<faixa-etária>
B,<id>,<dia>,<horário>,<id-prof>,<mínimo>,<máximo>,
V,<id>,<dia>,<horário>,<id-prof>,<mínimo>,<máximo>,<estatura>
T,<id>,<dia>,<horário>,<id-prof>,<propósito>,,
```

Dado	Significado	Formato
`id`	Identificador único	Número inteiro
`dia`	Dia da semana no qual a aula ocorre	1 = domingo, 2 = segunda-feira, etc.
`horário`	Horário no qual a aula ocorre	HH:MM (24h)
`id-professor`	Identificador do professor que dá a aula	Número inteiro
`mínimo`	Número mínimo de alunos	Número inteiro
`máximo`	Número máximo de alunos	Número inteiro
`faixa-etária`	Faixa-etária dos alunos da turma	String
`estatura`	Estatura máxima dos alunos da turma, em metros	Número real
`propósito`	Indica se o aluno treina para lazer ou campeonato	L ou C

Repare que as linhas que não possuem oito informações são completadas automaticamente com valores vazios, por isso sobram vírgulas ao final de uma linha que representa uma aula de tênis ou basquetebol. Basta ler e descartar a informação vazia.

Inscrições

Por fim, o arquivo `inscricoes.csv` contém as inscrições dos alunos para as turmas. O formato de cada linha, que representa uma inscrição, é o seguinte:

```
<id-aluno>,<id-aula>
```

Dado	Significado	Formato
`id-aluno`	Identificador do aluno a ser inscrito	Número inteiro
`id-aula`	Identificador da aula na qual está se inscrevendo	Número inteiro

2.2. Saída de dados

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

Relatório	Nome do Arquivo
Relatório de aulas confirmadas	`aulasconfirmadas.csv`
Relatório de aulas canceladas	`aulascanceladas.csv`
Relatório de inscrições canceladas	`inscricoescanceladas.csv`

Os dois primeiros relatórios exibem, em cada linha, informações sobre uma aula (que foi confirmada ou cancelada) e seu professor. O último exibe em cada linha dados da inscrição cancelada e do aluno inscrito.

Todos os relatórios devem estar ordenados primeiramente por dia e horário da aula, em segundo lugar pelo identificador da aula e, no caso de inscrições, em terceiro lugar pelo identificador do aluno. O padrão para escrita das linhas é detalhado nas tabelas abaixo:

Relatório	Padrão de Saída de Dados
`aulasconfirmadas.csv`	`<tipo>,<dia>,<horário>,<professor>,<qtd-alunos>,<X>`
`aulascanceladas.csv`	`<tipo>,<dia>,<horário>,<professor>,<telefone>,<email>`
`inscricoescanceladas.csv`	`<tipo>,<dia>,<horário>,<aluno>,<telefone>,<email>`

Dado	Significado	Formato
`tipo`	Modalidade esportiva	F, B, V ou T
`dia`	Dia da semana no qual a aula ocorre	1 = domingo, 2 = segunda-feira, etc.
`horário`	Horário no qual a aula ocorre	HH:MM (24hs)
`professor`	Nome do professor que dá a aula	String
`qtd-alunos`	Quantidade de alunos inscritos na aula	Número inteiro
`X`	Se a modalidade for: • Futebol: exibir faixa etária; • Basquetebol: não exibir nada; • Voleibol: exibir estatura máxima; • Tênis: informar se é lazer ou campeonato.	F: número inteiro V: número real com duas casa decimais (separe com ponto) T: L ou C
`telefone`	Telefone de contato (do professor ou do aluno)	String
`email`	E-mail de contato (do professor ou do aluno)	String

3. Opções de execução

A versão Java do seu programa deve suportar três modos de execução. Apesar do uso do pacote *default* não ser recomendado, para efeito dos exemplos abaixo vamos supor que a classe do seu programa que possui o método `main()` chama-se `Main` e encontra-se no pacote *default*. Neste caso, o programa deve poder ser chamado das três formas, como a seguir:

- `java Main`: 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 Main --read-only`: quando especificada esta opção, o programa deve ler os arquivos de entrada, montar as estruturas de objetos em memória e serializar esta estrutura em um arquivo chamado `ufesports.dat`;
- `java Main --write-only`: quando especificada esta opção, o programa deve carregar os objetos serializados no arquivo `ufesports.dat`, gerar os relatórios e escrevê-los nos arquivos de saída.

A versão C++ do programa não precisa implementar estas opções.

4. Condições de entrega

O trabalho deve ser feito obrigatoriamente em dupla e em duas versões: uma utilizando a linguagem Java, outra utilizando a linguagem C++. O primeiro deve ser entregue até o dia **05/08/2013** e o segundo até o dia **04/09/2013**, impreterivelmente.

Alunos que não fizerem o trabalho em dupla sofrerão penalidade de 2 pontos na nota do trabalho. No caso de haver um número ímpar de alunos matriculados, o professor indicará um aluno que poderá, a seu critério, fazer o trabalho sozinho ou juntar-se a uma dupla para formar um trio. As duplas para os trabalhos Java e C++ não precisam necessariamente ser as mesmas.

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 do LabGrad.

O recebimento dos trabalhos será feito por meio de um sistema automatizado. As seções a seguir detalham o formato preciso de entrega dos trabalhos Java e C++. É essencial que sejam seguidas à risca as instruções abaixo, caso contrário a correção automatizada não funcionará. Trabalhos que não passarem pela correção automatizada passarão por correção manual, porém sofrerão penalidade de 2 pontos.

4.1. Formato de entrega do trabalho Java

O código-fonte de sua solução deverá ser compactado e entregue por e-mail (anexo ao e-mail) para o endereço vsouza@ninfa.inf.ufes.br.

ATENÇÃO: este e-mail foi gentilmente criado pelos membros do laboratório NINFA para ser utilizado apenas para recebimento de trabalhos. Dúvidas devem ser enviadas para o e-mail do professor: vitorsouza@inf.ufes.br.

Serão aceitos trabalhos entregues até as 23h59 da data limite. O assunto do e-mail deverá ser o seguinte:

```
prog3:trab1:<nome1>:<nome2>:
```

O termo "<nome1>:<nome2>" deverá ser substituído pelos nomes dos alunos em ordem alfabética – somente um nome e um sobrenome de cada aluno, separados por vírgula e sem acentos, til ou cedilha, como no exemplo abaixo:

```
prog3:trab1:Flavio Varejao:Vitor Souza:
```

Repare, novamente, que os nomes Flávio e Vítor foram escritos em ordem alfabética de primeiro nome, sem acento e o sobrenome Varejão foi escrito sem til. É muito importante que o assunto do e-mail esteja correto, do contrário o sistema de correção automática não reconhecerá sua submissão e seu trabalho será considerado como não recebido. Caso um aluno tenha feito o trabalho sozinho (apesar da obrigatoriedade do trabalho em dupla), basta informar apenas `prog3:trab1:<nome1>:`.

Se tudo correr bem no recebimento do arquivo, você receberá um e-mail de confirmação do recebimento do trabalho. Neste e-mail haverá um *hash* MD5 (https://pt.wikipedia.org/wiki/MD5#Hashes_MD5) do arquivo recebido. Para garantir que o arquivo foi recebido sem ser corrompido, gere o *hash* MD5 do arquivo que você enviou 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 arquivo compactado enviado por e-mail deve estar no formato `tar.gz` com o nome `trab1.tar.gz` e conter os arquivos fonte e um arquivo de *build* do Ant. Ele não deve conter classes compiladas (.class). Estas devem ser geradas pelo software de *build* (construção de programas) Apache Ant (http://ant.apache.org). Um exemplo está disponível para download no site da disciplina.

Os arquivos fonte podem estar organizados da forma que você achar melhor, desde que o Ant consiga compilá-los e executar as classes geradas. Para que isso seja feito de forma automatizada, o arquivo de *build* do Ant deve, obrigatoriamente, encontrar-se na raiz do arquivo compactado. 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 no modo normal.
`ant run-read-only`	O programa deve ser executado no modo `--read-only` (ver Seção 3).

`ant run-write-only`	O programa deve ser executado no modo `--write-only` (ver Seção 3).
`ant clean`	Todos os arquivos gerados (classes compiladas, etc.) devem ser excluídos, sobrando somente o conteúdo original do arquivo compactado (i.e., código-fonte e arquivo de *build*).

4.2. Formato de entrega do trabalho C++

A entrega do trabalho C++ é similar à entrega do trabalho Java. Fique atento às seguintes diferenças:

- O assunto do e-mail de entrega do trabalho deverá ser `prog3:trab2:<nome1>:<nome2>` (`trab2` ao invés de `trab1`);
- O arquivo compactado com o código-fonte deverá ter o nome `trab2.tar.gz` (novamente, `trab2` ao invés de `trab1`);
- Ao invés do `build.xml` do Ant, o arquivo deverá conter (além do código-fonte) um arquivo chamado `Makefile` com instruções para que o programa `make` possa compilar e executar o programa:
  - O comando `make` (ou `make all`) deve compilar o programa;
  - O comando `make run` deve executar o programa (lembre-se de tomar por base o sistema Linux instalado nas máquinas do LabGrad);
  - Como citado na seção 3, o trabalho C++ não precisa ter as opções de execução *read-only* e *write-only*.

Assim como o trabalho Java, o arquivo compactado enviado deve ter apenas o código-fonte e o arquivo de *build*, sem nenhuma classe compilada. Lembre-se de conferir o *hash* MD5 gerado pelo sistema de recebimento automático com o *hash* do arquivo enviado para verificar o correto recebimento do mesmo.

5. Critérios de avaliação

Os trabalhos serão avaliados primeiramente pela execução correta, seguindo os critérios objetivos abaixo:

Critério	Nota
Trabalhos que não compilarem.	De 0 a 3
Trabalhos que compilarem mas não gerarem resultados corretamente.	De 3 a 7
Trabalhos que compilarem e gerarem os relatórios corretamente.	De 6 a 10
Trabalhos entregues fora do prazo.	-1 por dia de atraso
Trabalhos que não seguirem as condições de entrega descritas nas seções 4.1 e 4.2.	-2 pontos

Em segundo lugar, os trabalhos serão avaliados segundo critérios subjetivos definidos pelo professor, variando a nota dentro da faixa de notas definida pelo critério anterior. Alguns dos critérios subjetivos avaliados serão:

- 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 e das funcionalidades das novas versões da plataforma, como tipos genéricos, laço *for-each*, *try* com recursos fecháveis, etc.

Alguns alunos poderão ser aleatoriamente escolhidos para explicar o trabalho ao professor em entrevista. Tal entrevista consiste em explicar trechos do trabalho escolhidos arbitrariamente pelo professor. A nota do aluno dependerá do resultado desta entrevista.

6. Pontos extra

Para incentivar alunos que desejarem aprender conteúdos avançados da linguagem Java por conta própria*, são oferecidos pontos extra para os alunos que agendarem uma reunião com o professor e demonstrarem que adicionaram uma ou mais das seguintes funcionalidades ao programa básico:

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.	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.	Até 2 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 avaliado o código, que deve ser explicado pelos alunos.

* O professor se coloca a disposição para ministrar cursos avançados de Java em horários alternativos, caso os alunos se organizem, formem uma turma e indiquem uma série de dias e horários nos quais tais aulas poderiam ocorrer. Caso os horários sejam compatíveis com os do professor, as aulas seriam realizadas.

Esta opção não será oferecida para os trabalhos C++ por não haver tempo hábil para realização das entrevistas com os alunos após o prazo de entrega do trabalho 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.