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++.

O problema é baseado em um dos exercícios de modelagem UML feitos no início do curso. Leia esta especificação com atenção, pois a descrição do problema não é idêntica ao exercício de modelagem, foi modificada para o trabalho prático.

1. Descrição do problema

A empresa de entrega de refeições à domicílio Disque-Rango deseja um sistema de informação para melhor organizar seus negócios. Clientes fazem pedidos, discriminando um ou mais itens de cardápio e suas respectivas quantidades (por exemplo, João faz um pedido para receber em casa 2 lasanhas, 1 filé com fritas e 3 latas de cerveja). De um cliente deseja-se saber: nome, endereço, telefone e ponto de referência.

Itens de cardápio podem ser de três tipos: refeições, sobremesas e bebidas. Das refeições é necessário saber nome, preço e se é refeição quente ou fria. Das sobremesas deseja-se saber nome e preço. Das bebidas deseja-se saber nome, preço e volume em mililitros.

Uma vez que um pedido é concluído pelo atendente, ele é passado para um entregador para entrega em domicílio. A empresa possui vários entregadores e deseja registrar no sistema seus nomes e as placas de seus veículos. Todo dia no início de suas operações, a empresa verifica quais entregadores estão presentes. As entregas são passadas aos entregadores seguindo sempre a ordem alfabética das placas dos seus veículos. Esta ordem é sempre respeitada, mesmo que o entregador da vez não tenha ainda voltado da última entrega. Um entregador entrega sempre um pedido de cada vez.

No final do dia, a gerência gostaria que o sistema produzisse alguns relatórios, a saber:

- Pedidos do dia: inclui nome e telefone do cliente[1], data, hora e valor total do pedido para cada pedido feito naquele dia. Ordenado pela data do pedido (crescente);
- Entregadores: mostra, para cada entregador, quantas entregas ele fez, a soma dos valores totais dos pedidos entregues e quanto ele deve receber de comissão (entregadores recebem 5% do valor dos pedidos entregues). Em ordem alfabética (crescente) do nome do entregador;
- Itens vendidos: relata, para cada item de cardápio, quantas unidades foram vendidas naquela noite e se houve pico de procura, ou seja, se em um único pedido foram incluídas mais do que 5 unidades daquele item de cardápio.

[1] Como será visto nas seções seguintes, quando pessoas diferentes usam o mesmo número de telefone, o cliente que liga depois substitui o nome do cliente que ligou anteriormente. Para o relatório "Pedidos do dia", deve ser usado o nome do cliente que ligou por último. Em outras palavras, o sistema deve esquecer os clientes que ligaram anteriormente quando várias pessoas usam o mesmo número.

Ordenado primeiro pelo número de unidades vendida do item (decrescente) e, em caso de empate, pelo nome do item (alfabético crescente);

A seção a seguir especifica detalhes do funcionamento do sistema de informação contratado pela Disque-Rango.

2. Detalhamento

O sistema pode ser dividido em três partes: (1) carregamento de dados; (2) atendimento a pedidos; e (3) geração de relatórios.

2.1. Carregamento de dados

Os dados dos itens de cardápio da Disque-Rango e dos entregadores que estão presentes e disponíveis são fornecidos em dois arquivos em formato CSV (*Comma Separated Values*, ou Valores Separados por Vírgula)[2], porém utilizando ponto-e-vírgula como separador. Os nomes destes arquivos devem ser especificados na linha de comando ao executar a aplicação, da seguinte forma:

```
java nome.completo.da.classe.Principal -e entregadores.csv -c cardapio.csv
```

O arquivo especificado após a opção -e deve conter os dados dos entregadores em formato descrito anteriormente, com o nome do entregador na primeira "coluna" e a placa do seu veículo na segunda. A seguir, é mostrado um exemplo de arquivo de entregadores:

```
Entregador;Placa
Alex;XYZ 0001
Breno;ABC 0002
Cid;KKK 0003
Douglas;XPT 0004
Estêvão;BRB 0005
```

O arquivo especificado após a opção -c deve conter os dados dos itens do cardápio do Disque-Rango. As "colunas" deste arquivo devem mostrar, nesta ordem: (1) o código do item; (2) o tipo do item (R para refeição, B para bebida, S para sobremesa); (3) o nome do item; (4) o preço do item em R$; (5) no caso de refeições, um x representa uma refeição quente; (6) no caso de bebidas, o volume da bebida em mililitros. A seguir é mostrado um exemplo de arquivo de cardápio:

```
Código;Tipo;Item;Preço (R$);Quente [R];Volume [B]
1;R;Hamburguer;3;x;
12;R;Pizza Brotinho;3;x;
17;R;Salada pequena;2,5;;
21;B;Cola-cola 600ml;3;;600
25;B;Suco de laranja;2,5;;500
26;S;Salada de fruta;2,9;;
29;S;Açaí;5,5;;
```

[2] Este formato de arquivo, apesar de ser um formato puro texto, pode ser aberto por softwares de planilha eletrônica como Microsoft Excel e LibreOffice Calc, organizando os valores em colunas. Cada linha do arquivo é uma linha da planilha e o separador é utilizado para indicar uma mudança de coluna.

Erros a serem tratados

Durante o carregamento de dados, dois erros devem ser tratados:

1. Se um dos (ou ambos) arquivos não forem especificados na linha de comando, o programa deve imprimir a seguinte mensagem:

   ```
   Arquivos de entrada não especificados. Use: -e <entregadores> -c
                                   <cardápio>%n
   ```

2. No caso de exceções causadas por erros na leitura dos arquivos (arquivo não encontrado, arquivo não segue o formato especificado acima, etc.), seu programa deve imprimir a mensagem abaixo e, em seguida, ser terminado:

   ```
   Não foi possível processar os arquivos especificados: {0}, {1}%n
   ```

Onde:

- Em ambas as mensagens, `%n` deve ser substituído por uma quebra de linha;
- Na segunda mensagem, `{0}` deve ser substituído pelo arquivo especificado após a opção -e (arquivo de entregadores), enquanto `{1}` deve ser substituído pelo arquivo especificado após a opção -c (arquivo de cardápio).

A convenção acima para especificação das mensagens a serem utilizadas pelo programa serão usadas em todo este documento.

2.2. Atendimento a pedidos

A Disque-Rango possui um protocolo específico para atendimento a pedidos, explicado por meio do fluxograma da **Figura 1**. A tabela a seguir explica os símbolos mostrados na figura.

Símbolo	Significado
	Operação de saída de dados. O elemento traz entre aspas o formato exato que a mensagem deve ser impressa na tela. Como já mencionado anteriormente, `%n` representa quebra de linha e `{0}` deve ser substituído pela informação adequada.
	Operação de entrada de dados. O nome dentro do elemento funciona como uma variável. Outros elementos (ex.: decisões) podem se referir a este mesmo termo.
	Outra tarefa realizada pelo programa.
	Decisão. O caminho a seguir depende da resposta à pergunta que encontra-se dentro deste elemento.

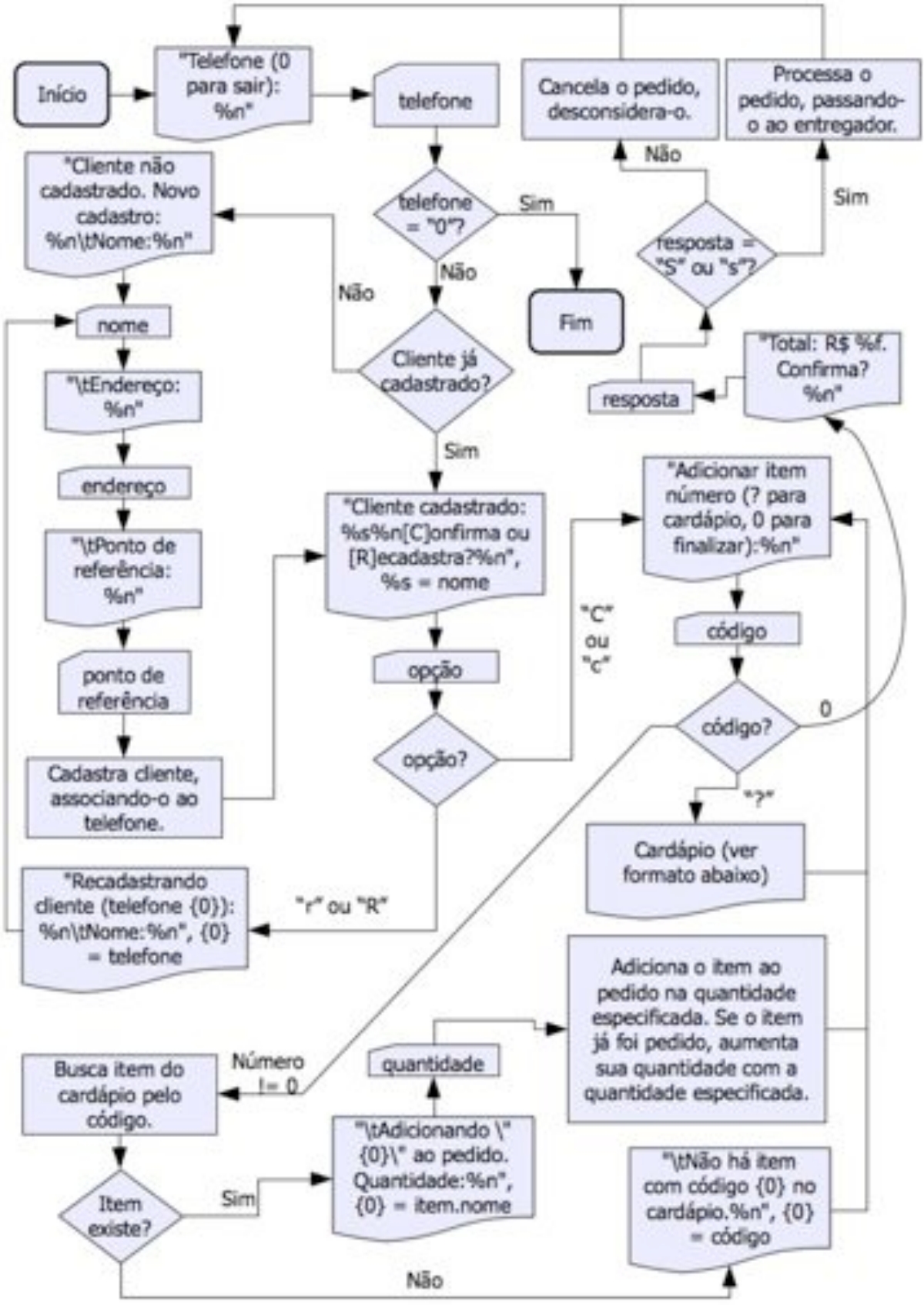

Figura 1. Fluxograma que descreve o atendimento a pedidos.

O processo acima deve ser feito após o carregamento dos dados do cardápio e dos entregadores. Ao chegar ao fim, o programa deve gerar os relatórios como descrito a seguir.

É importante ressaltar que as operações de entrada e saída (escrita de texto na tela e leitura dos dados informados pelo usuário pelo teclado) deve ser feita exatamente como especificado no processo da **Figura 1**. Do contrário, a correção automática (descrita na Seção 3) não funcionará, resultando em perda de pontos do trabalho.

Formato do cardápio

Durante a composição do pedido, o caractere "?" indica ao programa para imprimir o cardápio. O cardápio deve ser impresso por ordem de código de item no formato `"\t{0}\t{1} \t{2}%n"`, sendo:

- `{0}` = código do item;
- `{1}` = preço do item (precedido por "R$ ", com duas casas decimais, separando-as com vírgula);
- `{2}` = Nome do item.

Abaixo, exemplo de impressão dos itens do cardápio montado a partir do exemplo de cardápio da seção 2.1:

```
1	R$ 3,00	Hamburguer
12	R$ 3,00	Pizza Brotinho
17	R$ 2,50	Salada pequena
21	R$ 3,00	Cola-cola 600ml
25	R$ 2,50	Suco de laranja
26	R$ 2,90	Salada de fruta
29	R$ 5,50	Açaí
```

2.3. Geração de relatórios

Como falado na Seção 1, o programa deve gerar 3 relatórios: pedidos, entregadores e itens vendidos. Seu programa deve solicitar que o usuário digite o nome dos arquivos onde estes três relatórios serão salvos, da seguinte maneira:

Relatório	Mensagem
Pedidos do dia	`Relatório de pedidos do dia:`
Entregadores	`Relatório de entregadores:`
Itens vendidos	`Relatório de itens vendidos:`

Após cada mensagem, seu programa deve ler uma *string*. As *strings* informadas pelo usuário em resposta a estas mensagens devem ser utilizadas como nomes dos arquivos onde serão escritos os relatórios.

Os relatórios devem ser formatados da seguinte maneira:

Relatório	Formato

Pedidos do dia	`Nome;Telefone;Data;Hora;Valor total`
	`{0};{1};{2};{3};{4}`
	
	A primeira linha é o cabeçalho, a segunda deve ser repetida para cada pedido feito durante a execução do programa, sendo:
	• {0}: nome do cliente (como informado);
	• {1}: telefone do cliente (como informado);
	• {2}: data em que o pedido foi feito (no formato dd/MM/yyyy);
	• {3}: hora em que o pedido foi feito (no formato HH:mm);
	• {4}; valor total do pedido (precedido por "R$ ", com duas casas decimais, separando-as com vírgula).

Entregadores	`Entregador;Qtd. entregas;Valor total;Comissão`
	`{0};{1};{2};{3}`
	
	A primeira linha é o cabeçalho, a segunda deve ser repetida para cada entregador, sendo:
	• {0}: nome do entregador (como no arquivo);
	• {1}: número de entregas feitas;
	• {2}: valor total entregue, ou seja, soma do valor total de todos os pedidos entregues pelo entregador (mesmo formato do valor total do pedido, descrito acima);
	• {3}: comissão a receber, ou seja, 5% do valor mostrado em {2} (mesmo formato de {2}).

Itens vendidos	`Item;Qtd. vendida;Pico de procura`
	`{0};{1};{2}`
	
	A primeira linha é o cabeçalho, a segunda deve ser repetida para cada item de cardápio, sendo:
	• {0}: nome do item (como no arquivo);
	• {1}: quantidade de unidades deste item que foram vendidas, considerando todas as vendas;
	• {2}: "Sim" ou "Não", indicando, respectivamente, se houve ou não pico de procura deste item (5 ou mais unidades pedidas em um único pedido).

Novamente, é muito importante que seus relatórios sigam estritamente o formato descrito acima. Juntamente com esta especificação, encontram-se arquivos de exemplo que podem ser utilizados para comparação com o resultado produzido pelo seu programa. Discutiremos estes arquivos na Seção 3.

2.4. Persistência (somente em Java)

A versão Java do trabalho prático deve dar suporte a persistência dos dados. Seu programa deve:

- Ao finalizar, seu programa deve serializar os dados no arquivo `diskrango.dat`;
- Ao iniciar, se for especificada a opção `-p` na linha de comando, seu programa deve recuperar os dados serializados, continuando o programa do ponto em que parou. Note que neste caso não é necessário carregar os dados do cardápio e dos entregadores;
- Ao iniciar, se não for especificada a opção `-p` na linha de comando, seu programa deve funcionar como explicado anteriormente, com as opções `-c` e `-e`, carregando o cardápio e os entregadores e começando com nenhum pedido feito.

Novamente, ressaltamos que esta funcionalidade não precisa ser implementada na versão C++ do programa.

3. Execução automática

Os trabalhos passarão por uma bateria de testes automáticos preparados pelo professor. Um destes testes é disponibilizado aos alunos no site da disciplina para que possam verificar que o trabalho está funcionando ao menos parcialmente. No arquivo de testes disponibilizado, encontram-se:

- Os arquivos `cardapio.csv` e `entregadores.csv`, com os dados descritos na Seção 2.1;
- O arquivo `in.txt`, com uma sequência de comandos que simula a utilização do programa de forma automática;
- O arquivo `prof-out.txt`, com a saída esperada do programa;
- Os arquivos `prof-relatped.csv`, `prof-relatent.csv` e `prof-relatven.csv`, que possuem o conteúdo esperado dos relatórios que devem ser gerados pelo programa.

Para testar seu programa de forma automática, basta chamá-lo pela linha de comando especificando que a entrada de dados deve vir do arquivo in.txt e, opcionalmente, a saída de dados deve ser feita no arquivo out.txt, como no exemplo abaixo:

```
java classe.Principal -e entregadores.csv -c cardapio.csv < in.txt > out.txt
```

Dado o conteúdo de `in.txt`, seu programa deve gerar os arquivos `relatped.csv`, `relatent.csv` e `relatven.csv`, além de sua saída padrão ser armazenada em `out.txt`. Você pode comparar todos estes arquivos com a versão deles fornecida pelo professor utilizando a ferramenta `diff` (http://pt.wikipedia.org/wiki/Diff) ou algum de seus *front-ends* com interface mais amigável:

- No Windows: WinMerge (http://winmerge.org);
- No Linux: Meld (http://meldmerge.org);
- No Mac: FileMerge (https://developer.apple.com/technologies/tools/features.html).

Data e hora dos pedidos

Ao seguir as instruções acima e comparar o resultado do seu programa com o resultado esperado (gerado pelo professor), certamente o relatório de pedidos estará incorreto, mesmo que o seu programa tenha feito tudo corretamente. O motivo é simples: as datas e horários dos pedidos feitos na sua execução do programa certamente são diferentes das datas e horários dos pedidos feitos na execução do programa por parte do professor. Para que a comparação seja feita com sucesso, é preciso executar o programa em "modo de teste".

Seu programa deve, portanto, responder a um último parâmetro de linha de comando: `-teste`. Por exemplo:

```
java classe.Principal -teste -e entregadores.csv -c cardapio.csv
```

Quando este parâmetro for especificado, seu programa deve:

- Registrar o primeiro pedido sempre com data 20/12/2013 e hora 23:59:59 (que, a propósito, é o prazo para entrega do trabalho de Java);
- Registrar cada pedido subsequente 10 minutos após o anterior. Ou seja, o segundo pedido será registrado como feito em 21/12/2013 00:09:59, o terceiro pedido em 21/12/2013 00:19:59, e assim por diante;
- Note que: se um pedido não for confirmado (última verificação no fluxograma de pedidos) e for desconsiderado, o pedido seguinte deverá ser registrado 10 minutos após o pedido que foi desconsiderado e não 10 minutos após o último pedido confirmado.

Segue abaixo exemplo de relatório de pedidos em que 5 pedidos foram feitos, todos eles confirmados. Note que cada pedido ocorre 10 minutos após o anterior.

```
Nome;Telefone;Data;Hora;Valor total
Vítor;555-2131;20/12/2013;23:59;R$ 15,50
Ricardo;555-2167;21/12/2013;00:09;R$ 20,00
Giancarlo;555-2189;21/12/2013;00:19;R$ 24,00
Renata;555-2196;21/12/2013;00:29;R$ 8,50
Monalessa;555-2140;21/12/2013;00:39;R$ 20,00
João Paulo;555-2132;21/12/2013;00:49;R$ 23,00
```

Este comportamento deve ocorrer somente quando a opção -teste for especificada na linha de comando. Do contrário, seu programa deve registrada data e hora atuais para cada pedido.

Ponto extra

Ganhará um ponto extra o aluno ou grupo que preparar um teste automático similar ao fornecido pelo professor, contendo arquivo de cardápio, de entregadores, de entrada de dados, saída de dados esperada e relatórios esperados. Para ganhar ponto, o arquivo deve ser entregue em até uma semana antes do prazo de entrega do trabalho Java.

Os arquivos não podem ser copiados do professor ou de outros alunos e devem conter um cenário de uso do sistema não muito simples: devem ter ao menos 10 pedidos feitos, a maioria dos quais com mais de um produto no pedido e deve passar por todos os caminhos possíveis do fluxograma da **Figura 1**.

Ao enviar o arquivo ao professor, os alunos concordam em disponibilizá-lo no site da disciplina para que outros alunos possam executar também o seu teste.

4. Condições de entrega

O trabalho deve ser feito de maneira individual ou em dupla e em duas versões: uma utilizando a linguagem **Java**, outra utilizando a linguagem **C++**. A primeira deve ser entregue até o dia **27/12/2013** e a segunda até o dia **25/02/2014**, impreterivelmente. 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.

Fica também determinada a codificação de caracteres Unicode (UTF-8) como obrigatória para a entrega dos arquivos de código-fonte. Usuários do sistema operacional Windows em especial devem prestar atenção neste aspecto, pois a utilização de caracteres especiais (ex.: acentuados) em uma codificação diferente pode atrapalhar o sistema de correção automática e exigir intervenção manual por parte do professor. Neste caso, haverá penalidade de 1 ponto.

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.

É responsabilidade do aluno garantir a entrega dos trabalhos sem penalidade alguma. **Fique atento!**

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 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, 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, especificando `cardapio.csv` como arquivo de cardápio e `entregadores.csv` como arquivo de entregadores (ver Seção 2.1).
`ant continue`	O programa deve ser executado com opção `-p` (ver Seção 2.4).
`ant clean`	Todos os arquivos gerados (classes compiladas, etc.) devem ser excluídos, sobrando somente o conteúdo original do arquivo compactado (ou seja, 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. Atenção 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 no modo normal, especificando `cardapio.csv` como arquivo de cardápio e `entregadores.csv` como arquivo de entregadores (ver Seção 2.1);
- Como citado na seção 2.4, o trabalho C++ não precisa ter as opção de execução `-p`;
- O comando `make clean` deve excluir os arquivos gerados (classes compiladas, etc.).

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.	-7 pontos
Trabalhos que compilarem mas não gerarem resultados corretamente.	Até -3 pontos, dependendo do número de erros
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
Trabalhos cujo código-fonte encontrar-se em uma codificação de caracteres diferente de UTF-8, exigindo intervenção manual.	-1 ponto

Em segundo lugar, os trabalhos serão avaliados segundo critérios subjetivos definidos pelo professor. Alguns dos critérios subjetivos avaliados serão:

- Uso dos princípios básicos da orientação a objetos, como encapsulamento, abstração e modularização, evitando variáveis globais (constantes globais OK) e excesso de métodos `static`;
- 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.;
- Uso eficaz das bibliotecas de função do C++ (ex.: uso de string ao invés de char*).

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. Recuperação de pontos perdidos[3]

Para incentivar alunos que desejarem aprender conteúdos avançados da linguagem Java por conta própria, poderão recuperar pontos eventualmente perdidos os alunos que agendarem uma reunião com o professor até o final do período letivo e demonstrarem que adicionaram uma ou mais das seguintes funcionalidades ao programa básico:

Funcionalidade opcional	Pontos recuperados
Implementação de uma interface gráfica (janelas) que ofereça uma interação com o usuário mais rica que a interface somente texto obrigatória do trabalho.	Até 3 pontos
Implementação de uma interface Web (exceto Applet) que ofereça uma interação com o usuário mais rica que a interface somente texto obrigatória do trabalho.	Até 3 pontos
Substituir a serialização descrita na seção 2.4 por armazenamento em banco de dados relacional. Podem ser utilizadas soluções de mapeamento objeto/relacional.	Até 2 pontos

A coluna "Pontos recuperados" 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. Receberão pontuação máxima somente os trabalhos que realmente impressionarem pela qualidade do resultado.

Note que o trabalho com funcionalidade opcional não deve ser entregue seguindo as instruções da Seção 4, mas sim apresentado ao professor em uma entrevista. Após esta apresentação, o código poderá ser entregue ao professor via pen drive, e-mail, etc.

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.

[3] Recuperar pontos perdidos significa que os pontos extra recebidos serão somados à nota recebida no trabalho, porém esta última não poderá ultrapassar o valor máximo de 10.