Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Programação Modular
1º Semestre de 2016
Prof. Douglas G. Macharet

Trabalho Prático 1 – Estacionamento Modular

Valor: 10 pontos
Data de entrega: 13/04/2016

O objetivo deste trabalho prático é familiarizar o aluno com os principais conceitos de Orientação a Objetos e com a programação utilizando a linguagem Java. Deverá ser implementada uma estrutura de classes que permita armazenar e manipular os dados referentes a um estacionamento.

1 Instruções:

Você foi contratado para desenvolver o sistema responsável pela automatização de um estacionamento vertical (prédio). O estacionamento possui quatro níveis (andares) e cada nível possui 10 vagas, divididas da seguinte forma:

- 4 vagas de veículos pequenos
- 2 vagas de motocicleta
- 2 vagas para veículos grandes (e.g. caminhonetes)
- 2 vagas para portadores de necessidades especiais

As vagas devem obrigatoriamente ser preenchidas sequencialmente do nível mais baixo até o nível mais alto. Além disso, elas podem ser ocupadas da seguinte forma:

- Motocicletas: podem parar em qualquer tipo de vaga (exceto deficientes)
- Veículos pequenos: podem parar em vagas de veículos pequenos ou grandes (exceto deficientes)
- Veículos grandes: podem parar apenas em vagas de veículos grandes

Ao chegar ao estacionamento, o cliente acessa um totem onde ele informa o tipo do seu carro. O sistema deve então informar se existe ou não vaga disponível que ele pode utilizar, em caso afirmativo, ele deve informar em qual nível e posição essa vaga fica, caso contrário, deve informar LOTADO. As vagas são numeradas sequencialmente de acordo com o nível (N1, N2, N3, N4) e tipo de vaga (VP, MT, VG, NE). Logo, por exemplo, no primeiro nível, a primeira vaga para veículos pequenos é identificada por N1VP1. A hora de entrada é salva caso exista vaga disponível.

Na saída do estacionamento, o cliente deve receber um ticket com o valor que deve ser pago. O custo é calculado de acordo com um valor associado ao tipo da vaga utilizada e à quantidade de tempo. O valor da hora para cada vaga é:

- Vagas de veículos pequenos: R$ 6,00 / hora
- Vagas de motocicleta: R$ 3,50 / hora
- Veículos grandes: R$ 8,00 / hora
- Portadores de necessidades especiais: R$ 6,00 / hora

O programa deverá ser feito baseado no JAVA SDK 7. Para guardar as informações relativas às coleções, você pode utilizar uma das classes que implementa a interface `Collection<E>`. Você pode obter mais informações em: http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html.

2 Arquivos de Entrada e Saída:

2.1 Exemplo de arquivo de entrada

O arquivo de entrada contém diversas linhas representando carros acessando o totem de atendimento. Cada uma dessas linhas possui as seguintes informações:

- Tipo do atendimento (Entrada ou Saída)
- Horário (6:00 – 21:00)
- Placa
- Tipo do veículo

entrada.txt

```
E;06:12;ABC-1234;VP
E;06:25;XYZ-4321;MT
E;06:30;XAB-2121;VG
E;07:31;CAB-2987;VG
E;08:37;RBA-9715;VG
E;08:41;OXN-1973;NE
S;09:15;XAB-2121;VG
S;09:25;XYZ-4321;MT
E;10:37;PZS-3179;VG
S;11:13;OXN-1973;NE
```

2.2 Exemplo de arquivo de saída

O arquivo de saída contém as informações do totem para cada uma das interações do arquivo de entrada. De maneira geral, a saída terá o seguinte comportamento

- Entrada
  - Em caso de vaga livre, informa a posição, por exemplo ‘N1VP1’.
  - Caso contrário, escreve ‘LOTADO’.
- Saída
  - Informa ao usuário o tipo de vaga utilizado, o tempo e o valor a ser pago.

saida.txt

```
N1VP1
N1MT1
N1VG1
N1VG2
N2VG1
N1NE1
VG;02:45;32,00
MT;03:00;10,50
N1VG1
NE;2:32;15,20
```

3 Documentação:

Entre outras coisas, a documentação deve conter:

1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
2. Implementação: descrição sobre a implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes que porventura estejam omissos no enunciado.
4. Testes: descrição dos testes realizados e listagem da saída (não edite os resultados). Você pode propor outros testes além dos fornecidos com o enunciado.
5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.

O que deve ser entregue:

Envie um arquivo ZIP com o nome no formato 'tp1-primeironome1-primeironome2.zip', contendo os seguintes arquivos:

- Arquivo README.txt com os nomes completos dos alunos da dupla.
- O código fonte do programa em, Java bem endentado e comentado. Deve ser fornecido junto com o fonte um arquivo Makefile com as opções 'make' e 'make run'.
- A documentação do trabalho bem escrita e detalhada.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.
- Clareza, endentação e comentários no programa também serão avaliados.
- O trabalho deverá ser feito em dupla.
- Trabalhos copiados serão penalizados conforme anunciado.
- Penalização por atraso: $(2^d - 1)$ pontos, onde d é o número de dias de atraso.

Critérios de avaliação:

- Funcionamento correto (3 pts).
- Uso correto dos conceitos de OO (5 pts).
- Documentação (2 pts).