Universidade Federal do Ceará - Campus Russas
Programação Orientada a Objetos – 2018.2
Professor: Marcio Costa Santos
Trabalho 2 - The choosen one!

Apresentação e Motivação

Aplicativos de compartilhamento de carros e de carona conquistaram um imenso mercado nos últimos anos. Apesar de sua simplicidade de desenvolvimento, estes aplicativos fazem uso de uma serie de conceitos de orientação à objetos em sua concepção.
Em todos os projetos descritos aqui será cobrado dos alunos o desenvolvimento de um sistema relacionado com aplicativos de compartilhamento de transporte. Cada um dos projetos a seguir contém as informações básicas necessárias a seu entendimento, assim como uma explicação de como as notas serão atribuídas e da quantidade máxima de alunos em cada um dos projetos.
Todos os projetos devem ser entregues na data prevista no SIGAA, pelo SIGAA. A entrega se dará com um arquivo zipado, nomeado com o nome dos alunos em letras minúsculas e sem caracteres especiais com suas matriculas separadas por underline. por exemplo: joaodasilva_000001_mariadorosario_00002.zip
Devem ser entregues um relatório explicativo do sistema em pdf, o diagrama de classes UML do sistema em pdf e o código fonte (arquivos .java) organizados em uma pasta nomeada **src**.
Todos os trabalhos serão apresentados pelas equipes para o professor em data definida no SIGAA de acordo com sorteio realizado após a entrega dos trabalhos.

Uber POOL

A maneira mais rápida de passar em POO este semestre

Número de Alunos por Equipe: no máximo 2 alunos

Nota Máxima: 10,0

Descrição: Neste trabalho os alunos devem desenvolver um sistema que simule a interação dos clientes e dos motoristas de aplicativos de transporte. O programa deve ler um arquivo contendo uma representação abstrata de uma cidade (descrição detalhada no anexo 1), um outro arquivo contendo as posições de clientes assim como seus destinos finais e um terceiro arquivo contendo as posições iniciais dos motoristas. O programa deve então simular o que acontece neste ambiente com o aplicativo de transporte seguindo as seguintes regras:

1. A empresa recupera 25% dos valores das corridas dos motoristas
2. Os carros se movimentam à 50 km/h.
3. As requisições dos clientes são feitas aos 3 motoristas mais próximos (de acordo com suas distancias pelas ruas da cidade) em ordem. Motoristas com passageiros recusam todas as requisições que recebem e motoristas livres aceitam todas as requisições que são feitas a eles.
4. Se uma requisição do cliente não é aceita por nenhum dos 3 motoristas próximos a ele esta é perdida e o cliente faz uma nova requisição, para o mesmo destino após 10 minutos. Após 10 tentativas o cliente desiste e sai do aplicativo.
5. A simulação termina quando todos os clientes ou foram atendidos ou desistiram de usar o aplicativo.

O programa deve imprimir na tela quando um cliente é atendido e quando ele desiste de pedir um carro. Ao final da execução o programa deve listar todos os motoristas e seus respectivos lucros, assim como o lucro da empresa.

Barema de Notas: .

Corretude - 2,0 Diz respeito a execução correta do programa, neste quesito será verificado se o raciocínio dos alunos para resolver o problema foi correto.

Modelagem conceitual - 3,0 Diz respeito ao uso dos conceitos de programação orientada à objetos no código do programa.

Tratamento de Execeções e Problemas - 0,5 Diz respeito a robustez do código, como este reage a problemas nos dados de entrada.

Documentação - 0,5 Diz respeito a facilidade ou dificuldade de leitura do código, aos comentários em classes e funções.

UML - 2,0 Diz respeito a correspondência entre o diagrama de classes apresentado e a implementação feita.

Relatório - 2,0 Diz respeito a explicação do desenvolvimento e da utilização de das funções propostas no programa.

Uber X

Sua POO de todos os dias por um tempo de programação razoável

Número de Alunos por Equipe: no máximo 2 alunos

Nota Máxima: 15,0

Descrição: Neste trabalho os alunos também devem desenvolver um sistema que simule a interação dos clientes e dos motoristas de aplicativos de transporte, entretanto temos uma componente incerta no comportamento do motorista. O programa deve ler um arquivo contendo uma representação abstrata de uma cidade (descrição detalhada no anexo 1), um outro arquivo contendo as posições de clientes assim como seus destinos finais e um terceiro arquivo contendo as posições iniciais dos motoristas assim como suas taxas de cancelamento com passageiros e sem passageiros. O programa deve então simular o que acontece neste ambiente com o aplicativo de transporte seguindo as seguintes regras:

1. A empresa recupera 25% dos valores das corridas dos motoristas
2. Existem três tipos de motoristas: UberX, Uber Vip e Uber Select.
3. A tarifa fixa para os motoristas é de 1 real por quilômetro rodado para uberX, de 2 reais para Uber Vip e de 3 reais para uber Select.
4. Os carros se movimentam com velocidade variadas. Uber X viaja à 50km, uber Vip à 40km e uber Select à de 30km/h.
5. As requisições dos clientes são feitas aos 3 motoristas mais próximos (de acordo com suas distancias pelas ruas da cidade) em ordem. Motoristas aceitam as requisições de acordo com suas taxas de aceitação de maneira aleatória. Em suas requisições os clientes indicam o tipo de carro que eles desejam(X, VIP ou Select), caso não haja em um raio de 10km um motorista daquela categoria, o pedido é atualizado para a categoria superior, com o ônus da diferença de preço sendo pago pela empresa, ou seja, o motorista recebe o valor da corrida de acordo com sua categoria e o cliente paga de acordo com a sua requisição e a empresa paga a diferença.
6. Se o cliente faz uma requisição e está é aceita por um motorista que está a mais de 10 minutos distante do cliente, ele cancela a requisição.

7. Se uma requisição do cliente não é aceita por nenhum dos 3 motoristas próximos, se não existem motoristas próximos de nenhuma categoria ou se o cliente cancela ela é perdida e o cliente faz uma nova requisição, para o mesmo destino após 10 minutos. Após 10 tentativas o cliente desiste e sai do aplicativo.
8. A simulação termina quando todos os clientes ou foram atendidos ou desistiram de usar o aplicativo.

O programa deve imprimir na tela quando um cliente é atendido e quando ele desiste de pedir um carro. Ao final da execução o programa deve listar todos os motoristas e seus respectivos lucros.

Barema de Notas: .

Corretude - 2,5 Diz respeito a execução correta do programa, neste quesito será verificado se o raciocínio dos alunos para resolver o problema foi correto.

Modelagem conceitual - 4,0 Diz respeito ao uso dos conceitos de programação orientada à objetos no código do programa.

Tratamento de Execeções e Problemas - 1,0 Diz respeito a robustez do código, como este reage a problemas nos dados de entrada.

Documentação - 1,5 Diz respeito a facilidade ou dificuldade de leitura do código, aos comentários em classes e funções.

UML - 3,0 Diz respeito a correspondência entre o diagrama de classes apresentado e a implementação feita.

Relatório - 3,0 Diz respeito a explicação do desenvolvimento e da utilização de das funções propostas no programa.

Uber SELECT

Os tópicos mais seletos de POO e um tempo provavelmente aceitável

Número de Alunos por Equipe: no máximo 3 alunos

Nota Máxima: 20,0

Descrição: Um dos grandes problemas quando executamos simulações computacionais é a falta de recursos (memória, processamento, etc...). Por isso muitas simulações são executadas em servidores de cálculos e executam continuamente. Neste projeto desenvolveremos uma arquitetura cliente servidor para realizar a simulação de uma rede de transportes passeados em aplicativos. A simulação do procedimento deve respeitar as regras e procedimentos descritos para o trabalho **uberX**, entretanto deve ser realizada inteiramente no servidor. A aplicação cliente apenas lê os dados e os transmite para o servidor, este realiza todos os cálculos e retorna os resultados para a aplicação cliente. Importante: o servidor deve suportar pelo menos 3 requisições simultâneas e deve realizar as computações em paralelo para as requisições que recebe.

Barema de Notas: Corretude - 3,0 Diz respeito a execução correta do programa, neste quesito será verificado se o raciocínio dos alunos para resolver o problema foi correto.

Modelagem conceitual - 4,5 Diz respeito ao uso dos conceitos de programação orientada à objetos no código do programa.

Tratamento de Execeções e Problemas - 1,5 Diz respeito a robustez do código, como este reage a problemas nos dados de entrada.

Documentação - 2,0 Diz respeito a facilidade ou dificuldade de leitura do código, aos comentários em classes e funções.

UML - 3,5 Diz respeito a correspondência entre o diagrama de classes apresentado e a implementação feita.

Relatório - 3,5 Diz respeito a explicação do desenvolvimento e da utilização de das funções propostas no programa.

Uso de Sockets e Threads - 2,0 Diz respeito ao uso correto das estruturas de paralelismo e conexão em Java.

Uber VIP

Os tópicos mais seletos de POO, as API's mais famosas e um tempo certamente insuficiente

Número de Alunos por Equipe: no máximo 3 alunos

Nota Máxima: 30,0

Descrição: Melhor que realizar simulações de um sistema é desenvolver um sistema real e implanta-lo. Neste projeto nós desenvolveremos um conjunto de aplicações que unidas funcionam como um aplicativo de transporte. Este sistema tem 3 partes:

Aplicação do Cliente: A aplicação do cliente deve conter uma interface gráfica que permita ao cliente fazer a requisção de um motorista, cancelar a requisição feita e ao final da corrida saber o valor da corrida.

Aplicação do Motorista: A aplicação do motorista deve conter uma interface gráfica que permita ao motorista aceitar ou recursar uma corrida, ver o endereço onde buscar o cliente, ver o endereço final do cliente e saber o valor da corrida.

Servidor: O servidor deve gerenciar as requisições dos clientes e dos motoristas, assim como manter um histórico das corridas realizadas e manter o balanço financeiro da empresa.

O gerenciamento realizado no servidor deve satisfazer as premissas descrita no projeto *uberX*. Os alunos são livres para utilizarem as API's de desenvolvimento gráfico que desejarem, assim como a de localização. Uma recomendação é Java Swing + openstreetmap.

Barema de Notas: Corretude - 4,0 Diz respeito a execução correta do programa, neste quesito será verificado se o raciocínio dos alunos para resolver o problema foi correto.

Modelagem conceitual - 5,5 Diz respeito ao uso dos conceitos de programação orientada à objetos no código do programa.

Tratamento de Execeções e Problemas - 2,5 Diz respeito a robustez do código, como este reage a problemas nos dados de entrada.

Documentação - 3,0 Diz respeito a facilidade ou dificuldade de leitura do código, aos comentários em classes e funções.

UML - 4,5 Diz respeito a correspondência entre o diagrama de classes apresentado e a implementação feita.

Relatório - 5,0 Diz respeito a explicação do desenvolvimento e da utilização de das funções propostas no programa.

Uso de Sockets e Threads - 3,0 Diz respeito ao uso correto das estruturas de paralelismo e conexão em Java.

Uso correto das API's - 2,5

Anexo 1 - Representação da cidade, dos clientes e motoristas

A cidade será representada como um grafo. Um grafo é uma estrutura matemática composta por pontos, ligados por arestas. O arquivo de entrada da representação da cidade é formado da seguinte maneira, na primeira linha está o número de vértices e a partir da segunda linha temos cada aresta possível, representa pelos dois vértices que ele liga e pelo tamanho em km da mesma.
Por exemplo, a seguir temos um exemplo desse tipo de arquivo

```
4
1 2 40
1 3 30
1 4 40
2 3 50
2 4 40
3 4 40
```

Existe diferente maneiras de manter um grafo em um programa de computador, a maneira mais simples e mais útil para este trabalho é uma *matriz de adjacências*. Uma matriz de adjacências é uma matriz onde cada posição armazena a distância entre dois vértices. Veja que a nossa matriz de adjacências é completa, ou seja, existe uma aresta entre quaisquer dois vértices distintos. Por exemplo, para nosso exemplo, a matriz de adjacências é

0	40	30	40
40	0	50	40
30	50	0	40
40	40	40	0

Para os clientes, o arquivo contém um identificador para o cliente, seu ponto de partida e seu ponto de chegada de acordo com o grafo lido anteriormente, por exemplo

```
Gabriel 1 2
Samuel 3 4
```

E finalmente, para os motoristas temos o identificador do motorista, assim como sua posição inicial, sua taxa de cancelamento e sua probabilidade de cancelamento. Por exemplo

```
Rafael 1 5 40
Miguel 3 7 10
```