GCC178 – Práticas de Programação Orientada a Objetos

Trabalho Prático

Objetivo do Trabalho

O principal objetivo deste trabalho é praticar os conceitos aprendidos na disciplina. Espera-se com isso que os alunos possam revisar os conteúdos vistos em sala e entendê-los melhor.

Um segundo objetivo é manter os alunos praticando a programação orientada a objetos, pois o aprendizado do conteúdo da disciplina só acontece ao se colocar em prática os conceitos estudados.

Proposta do Trabalho

O trabalho a ser desenvolvido corresponde a um Simulador de Eventos Discretos para um determinado cenário. O simulador deverá ser implementado atendendo os requisitos apresentados nesta proposta.

A realização deste trabalho deve começar pelo entendimento do tema “Simulação de Eventos Discretos”. Para isso, a seguir apresento uma breve introdução sobre esse assunto. No entanto, sugiro que os alunos procurem outros materiais que os ajudem no entendimento desse tema, tal como o material disponível em: http://www.ime.usp.br/~reverbel/mac212/eps/ep2.pdf

Introdução a Simulação de Eventos Discretos

A simulação de eventos discretos é utilizada para simular eventos da vida real, tais como eventos relacionados com filas de execução de tarefas. Nesse tipo de simulação utiliza-se o conceito de “tempo simulado”, que é diferente do tempo real (de relógio). A partir de dados amostrados ou fictícios o sistema simula o atendimento a tarefas, gerenciando filas de espera e permitindo cálculos que demonstram o comportamento do cenário que está sendo simulado.

Para melhor entendimento dos conceitos, vamos utilizar como exemplo a rotina de atendimento em uma agência bancária. Nesse cenário, poderíamos utilizar a simulação de eventos discretos para estimar quanto tempo um cliente ficará na fila até ser atendido, o tamanho médio das filas etc.

Nesse cenário da agência bancária podemos ter diferentes tipos de clientes (entidade atendida), como por exemplo, o cliente normal e o preferencial (idosos, gestantes e portadores de deficiência). Além disso, cada cliente pode demandar uma quantidade de tempo diferente para ser atendido, dependendo da quantidade de ações necessárias para ele (número de contas a pagar, número de saques, número de depósitos, número de transferências etc).

Do outro lado temos os atendentes (entidade que fará o atendimento), que também podem ser de diferentes tipos, como por exemplo, atendente experiente e atendente em treinamento. O tipo da atendente também poderá influenciar no tempo que o cliente levará para ser atendido.

Nesse tipo de simulação podemos ter diferentes tipos de eventos, tais como: (1) chegada de um cliente e (2) término do atendimento de um cliente. No começo da simulação os eventos iniciais são colocados em uma fila. Os dados desses eventos devem ser disponibilizados como entrada para a simulação. Além disso, ao longo da simulação novos eventos podem ser gerados e, portanto, incluídos na fila de eventos. Por exemplo, quando um cliente começa a ser atendido pelo caixa do banco, um evento “fim de atendimento do cliente” deverá ser incluído na fila de eventos para que o término do atendimento possa ser executado pelo simulador no futuro. De modo geral, a simulação é realizada a partir de um laço de eventos (*event loop*) que consiste em:

- retirar o elemento com maior prioridade da fila de eventos.
- atualizar o tempo atual da simulação de acordo com o elemento retirado.
- dar o devido tratamento ao evento retirado da fila (podendo gerar a inclusão de novos eventos na fila).

Observe que, a cada iteração do *loop*, o simulador remove um evento da fila de eventos e trata-o. No caso do exemplo em questão, o tratamento do evento pode ser o atendimento do cliente pelo caixa do banco ou, se todos atendentes estiverem ocupados, a colocação do cliente em uma fila de espera. Note que essa fila de espera de atendimento não é a mesma coisa do que a fila de eventos da simulação.

Ao final da simulação um relatório em arquivo texto deve ser gerado com informações como:

- Tempo total simulado.
- Número de eventos tratados.
- Tempo médio de espera na fila de atendimento.
- Tamanho médio da fila de atendimento.
- Tamanho máximo da fila de atendimento.
- Tempo médio de atendimento de cada tipo de cliente etc.

Além disso, ao final da simulação, uma interface gráfica contendo um gráfico que apresente a evolução temporal de algumas das informações mencionadas anteriormente deve ser apresentada.

Por fim, a implementação deve atender os seguintes requisitos:

Requisito 1 – Classes de Entidades Atendidas

- Deve existir mais de um tipo de entidade atendida.
- Deve existir uma classe para cada tipo de entidade atendida.
- Se necessário, elas devem conter o método to*String*.

Requisito 2 – Classes de Atendentes

- Deve existir mais de um tipo de entidade atendente.
- Deve existir uma classe para cada tipo de entidade atendente.
- Se necessário, elas devem conter o método to*String*.

Requisito 3 – Classe Geral de Eventos

- Deve ser criada uma classe geral (abstrata) para todos seus eventos.
- Se necessário, essa classe deverá implementar a interface Comparable (da API padrão do Java).
- Se necessário, ela deve conter o método to*String*.

Requisito 4 – Classes Específicas e Eventos

- Devem existir pelo menos dois tipos de eventos.
- Implemente uma classe específica para cada tipo de evento.
- Se necessário, elas devem conter o método to*String*.

Requisito 5 – Classe de Acesso a Dados

- Essa classe deve permitir a leitura dos dados de entrada da simulação a partir de um arquivo texto denominado “dadosEntrada.txt”.
- O arquivo de entrada terá informações como:
  - Quantidade de atendentes por tipo.
  - Estratégia de atendimento das filas.
  - Informações das entidades atendidas (tempo da chegada, tipo de entidade etc.).

Requisito 6 – Classe da Simulação

- Classe que conterá toda a lógica da simulação propriamente dita.

Requisito 7 – Estatísticas da Simulação

- A simulação deverá gerar estatísticas por atendente.
- A simulação deverá gerar estatísticas por unidade de tempo para permitir a plotagem do gráfico.

- Estatísticas gerais, ou seja, considerando todos os postos de atendimentos, também devem ser geradas.
- Os dados das estatísticas devem ser salvos em um arquivo texto denominado “estatisticas.txt”

Requisito 8 – Classe Principal
- Classe a partir da qual a simulação é inicializada.
- Nenhum dado deve ser solicitado ao usuário, ou seja, todas as informações necessárias para a simulação devem estar contidas no arquivo de entrada de dados “dadosEntrada.txt”.

Requisito 9 – Tratamento de Exceções
- Deve-se utilizar tratamento de exceções na simulação.

Requisito 10 – Documentação Javadoc
- Toda a implementação deve estar devidamente comentada no padrão Javadoc.

Requisitos Não-Funcionais

- O trabalho deverá usar corretamente os conceitos de Orientação a Objetos.
- O trabalho deverá fazer uso de composição e/ou agregação.
- O trabalho deverá fazer uso de herança.
- O trabalho deverá fazer uso de polimorfismo (variável polimórfica e polimorfismo de método).
- O trabalho deverá ter um bom *Design* de Classes.
- Deve ser entregue junto com o trabalho o Diagrama de Classes UML da modelagem do sistema.
- Serão avaliados a legibilidade do código, organização e uso de comentários.

Pontuação e Entrega

Conforme previsto no Plano de Curso, este trabalho vale 25% da nota da disciplina.

Para a entrega final do trabalho deve haver uma pasta raiz contendo todos os códigos do trabalho e um arquivo **PDF** com o diagrama de classes UML. Essa pasta deve ter o nome “*TrabalhoPratico_Grupo@*”, onde @ corresponde à identificação (número) do grupo. A pasta deve ser **compactada** (.zip) **em um único arquivo** com o mesmo nome da pasta (ex: “*TrabalhoPratico_Grupo@.zip*”) e esse arquivo deve ser enviado **até dia dd/mm/2020**, **às 23h50**, via atividade que será aberta para essa finalidade no **Campus Virtual**.

Temas de Trabalho

Por meio de um sorteio foi definido o seguinte tema de trabalho para cada um dos grupos:

Grupo 1: Fila clientes em uma agência bancária

Grupo 2: Fila de estudantes no restaurante universitário

Grupo 3: Fila de pacientes em um hospital

Grupo 4: Fila de trabalhadores no INSS

Grupo 5: Fila de veículos em pedágio rodoviário

Grupo 6: Fila de pessoas para comprar ingressos para o cinema

Grupo 7: Fila de clientes em uma farmácia

Grupo 8: Fila de caminhões para descarregar em um porto

Grupo 9: Fila para votação em seções eleitorais

Grupo 10: Fila de carros em um *drive-thru* de um *fast food*

Grupo 11: Fila de passageiros para embarcar nas balsas RJ-Niterói

Grupo 12: Fila de clientes em um supermercado

Grupo 13: Fila de veículos para abastecimento em um posto de combustíveis

Grupo 14: Fila de pessoas para vacinação em um posto de saúde

Grupo 15: Fila de clientes aguardando atendimento telefônico em um 0800.