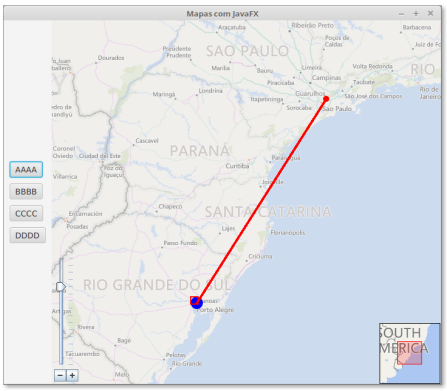
Pontifícia Universidade Católica do Rio Grande do Sul
Escola Politécnica - Programação Orientada a Objetos

Trabalho Final

Sistema para Consultas sobre Informações Aéreas



1 Descrição Geral do Trabalho

Leia atentamente o enunciado do problema, identifique os elementos e estude as classes que deverão ser criadas para a implementação do trabalho.

A partir da modelagem realizada em aula do sistema *MyFlight*, o objetivo deste trabalho é implementar diversas consultas sobre a estrutura de dados, com a possibilidade de visualizar os resultados de forma gráfica, isto é, desenhados em um mapa. Um objetivo secundário é promover a utilização de controle de versão (*git*), gerenciando as contribuições dos integrantes do grupo de trabalho.

2 Leitura dos arquivos e Modelagem

A primeira tarefa para a realização deste trabalho é entender os dados de entrada, e implementar a leitura dos arquivos de dados. Relembrando as estruturas criadas no sistema *MyFlight*:

- Companhia Aérea: possui um código e um nome (*airlines.dat*);
- Aeronave: possui um código, uma descrição e uma capacidade de passageiros (*equipment.dat*);
- Aeroporto: possui um código, um nome, uma localização geográfica e um identificador de país (*airports.dat*);
- Rota: é identificada por uma aeronave, aeroporto de origem, aeroporto de destino e companhia aérea (*routes.dat*);

As classes *Voo* e *GerenciadorVoos* não serão utilizadas.

Além desses dados, há também um arquivo com os identificadores e nomes dos países do mundo (*countries.dat*). Deve-se criar uma classe para representar um país.

Relembrando, cada classe é acompanhada por um gerenciador, capaz de incluir itens e fazer buscas diversas sobre os dados. Nesse sentido, também deve-se criar um gerenciador para países, a exemplo dos demais.

Inicialmente, é necessário analisar os dados disponíveis nestes arquivos para depois fazer a segunda tarefa: a modelagem do conjunto de classes para a realização de consultas. Estruturas de lista linear poderão ser utilizadas, ou qualquer outra coleção da API Java. Ou mesmo alguma classe que implemente uma estrutura diferente.

Observação: Não deve ser utilizado um banco de dados, e todos os dados dos arquivos devem ser lidos para a memória.

3 Consultas sobre a Estrutura

O aplicativo deverá realizar diversas consultas. A maior parte das consultas parte de um ponto no mapa, geralmente um aeroporto. Para tanto, a partir da posição do *mouse* (veja o método *consulta* na classe *JanelaConsulta* para entender) deve-se procurar a localização mais próxima dentro de uma determinado raio (distância). Por exemplo, para selecionar o Aeroporto Salgado Filho, deve ser possível clicar em um local próximo à localização real (por exemplo, num raio de 5 km).

O aplicativo deverá oferecer as seguintes consultas:

1. Desenhar todos os aeroportos onde uma determinada companhia aérea opera. Mostre também as rotas envolvidas.

2. Exibir uma estimativa de volume de tráfego de todos os aeroportos ou de um país específico. Deve-se explorar o tamanho e cor do ponto correspondente, para destacar aeroportos com maior ou menor tráfego.

3. Selecionar um aeroporto de origem e um de destino e mostrar todas as rotas possíveis entre os dois, considerando que haverá no máximo 2 conexões entre eles. Por exemplo, se houver as rotas POA->GRU->LHR e POA->GIG->LHR, esta consulta deve desenhar todas elas. É importante evitar *loops*, ou seja, não voltar para um aeroporto já percorrido. Ao selecionar uma rota, mostrar o tempo aproximado total de vôo e destacá-la no mapa com outra cor.

4. Selecionar um aeroporto de origem e mostrar todos os aeroportos que são alcançáveis até um determinado tempo de vôo (ex: 12 horas), com no máximo duas conexões.

Quando uma rota é exibida, deve-se mostrar também a distância entre os pontos e a aeronave sendo utilizada.

Estas consultas podem usar outras estruturas de dados estudadas em aula para auxiliar no processamento, ou qualquer outra estrutura criada especificamente para o trabalho.

4 GUI

Para tornar mais interessante a visualização dos dados, utilizaremos uma biblioteca capaz de exibir um mapa, denominada *JXMapViewer*. Essa biblioteca acessa automaticamente servidores de mapas e exibe as imagens correspondentes. Também é possível acrescentar pontos de referência e linhas ao mapa exibido, ou seja, o que utilizaremos para visualizar a localização dos aeroportos e do traçado das rotas.

Faça um *clone* do projeto *MyFlight* e selecione o *branch* tf: ele contém os arquivos e alterações necessárias para o desenvolvimento deste trabalho. O exemplo de aplicação exibe um mapa, e também demonstra como desenhar pontos e traçados (linhas). O exemplo também apresenta como obter as coordenadas de uma localização clicada pelo usuário.

Observação: não deve haver nenhuma interação pelo terminal.

5 Critérios de Avaliação

Leia com atenção os critérios de avaliação e siga corretamente as instruções para entrega e apresentação:

- Leitura de arquivos e correto armazenamento dos dados: 1 ponto
- Interface gráfica com o usuário: 1 ponto
- Consultas sobre os dados: 7 pontos
    - Consulta 1: 1,5 pontos

    - Consulta 2: 1,5 pontos
    - Consulta 3: 2 pontos
    - Consulta 4: 2 pontos
- *Commits* no *git* de cada integrante do grupo: 1 ponto

Observações:

- A implementação deve seguir as orientações dadas em aula quanto a convenções Java para nomes de identificadores e estrutura das classes.
- Não serão aceitos trabalhos com erros de compilação. Programas que não compilarem corretamente terão nota ZERO.

5.1 Entrega e Apresentação

- Os trabalhos são individuais ou em duplas.
- O projeto deve ser obrigatoriamente desenvolvido através de controle de versão (*git*), e o repositório deve ser informado ao professor assim que for criado, para acompanhamento. Se for utilizado o *BitBucket*, o usuário do professor (*mflash*) poderá ser incluído como colaborador no projeto. A versão final deve ser publicada (*commit*) até a data e hora especificada pelo professor. É fundamental a divisão de tarefas e o trabalho colaborativo: todos os integrantes do grupo devem ter contribuído durante o desenvolvimento do trabalho.
- Trabalhos entregues, mas não apresentados, terão sua nota anulada pelo professor. Durante a apresentação será avaliado o domínio da resolução do problema, podendo inclusive ser possível invalidar o trabalho quando constatada a falta de conhecimento sobre o código implementado.

- **A cópia parcial ou completa do trabalho terá como consequência a atribuição de nota ZERO ao trabalho dos alunos envolvidos. A verificação de cópias é feita inclusive entre turmas.**

Document generated by eLyXer 1.2.5 (2013-03-10) on 2018-05-21T12:37:48.820256