Sistema FuraFila de Atendimentos
Proj2

Informações

- Entrega pelo Portal COMP em <http://trab.dc.unifil.br/moodle/>;

1 Introdução

Neste trabalho, vamos implementar um sistema de fila de chamadas com senhas, como os que encontramos em *fast-foods*, agências bancárias e serviços públicos. Em alguns desses casos, há senhas diferentes para clientes distintos, o que permite priorizar alguns atendimentos em detrimento de outros, de acordo com leis ou regras próprias das instituições. Implementar essa regra de negócio vai ser o principal foco deste trabalho, sem esquecer de considerar a melhor eficiência assintótica para nossos objetivos.

O trabalho é ambicioso mas muitas das funcionalidades já estão pré-implementadas no projeto `FuraFilaCmd`, disponível no Portal COMP. Por isso, recomendo que utilize-o como ponto de partida. Nele, toda a lógica de interface de linha de comando já está implementada[1], além de incluir a estrutura e as classes onde iremos implementar os nossos algoritmos necessários para o programa funcionar.

De maneira geral, caberá ao aluno implementar:

- As TADs FilaSimples, FilaOrdenada, FilaPrioritaria;
- A lógica de priorização entre senhas emitidas pelo sistema, para as diferentes classes de clientes;
- Implementação do padrão de projeto *Iterator* para cada TAD supracitada.

2 Roteiro de Trabalho

Para facilitar a construção desse trabalho ambicioso, separei as atividades em etapas. É necessário ter as etapas anteriores inteiramente prontas antes de iniciar as conseguintes.

2.1 Explorando o projeto

1. Abra o projeto em sua instalação de IntelliJ IDEA e rode o programa. Teste as diferentes funcionalidades oferecidas e procure entender o funcionamento conceitual do programa.

[1] Há um projeto com interface gráfica, mas esta implementação não está terminada, seu uso fica por risco do aluno.

2. Antes de explorar o código-fonte do programa, abra a interface `furafila.colecoes.Fila` e leia sua documentação. Essa interface é uma especialização da interface `Collections` do próprio Java. Por esse motivo, procure sua documentação Javadocs e entenda o contrato de implementação dessa interface como um todo, e as necessidades específicas de cada método por ela prevista.

3. Tendo aprendido o contrato de `Collections` e `Fila`, implemente a classe `FilaSimples`, que se trata de uma TAD de fila comum, seguindo a lógica FIFO (*first-in first-out*). *Sugiro* a utilização de testes unitários para confirmar a correta implementação da classe.

4. Classifique assintoticamente as operações da classe `FilaSimples`, escrevendo no comentário javadocs.

5. Se tiver implementado `FilaSimples` corretamente, todas as funcionalidades do programa deixarão de lançar exceções. A partir de agora, você deve comentar com Javadocs todos os métodos das classes do projeto, exceto aquelas do pacote `furafila.colecoes`. O objetivo é que você entenda o funcionamento do programa.

2.2 Senhas com prioridades

6. Neste ponto, o programa está quase totalmente funcional, mas os tipos distintos de senha não têm prioridades de atendimento entre si que não a própria ordem de emissão dos bilhetes. Esse funcionamento não atende às necessidades de nosso principal *stakeholder*, que tem três tipos de clientes:

   Especiais: são clientes que devem ser atendidos o quanto antes. Entre dois clientes especiais, o que chegou primeiro deve ser atendido antes.

   Idosos: clientes com idade superior a 59 anos, são atendidos com até vinte minutos de antecedência que clientes comuns. Por exemplo, se o cliente comum tirou sua senha às 18:20, e o idoso tirou a sua às 18:40, este é atendido primeiro que aquele. Agora, se o idoso se atrasasse mais um segundo (18:40:01), o cliente comum seria atendido antes. Entre dois idosos, quem chegar primeiro é atendido antes.

   Comuns: compõem a maioria dos clientes, e são atendidos em ordem de chegada.

   Com base nessas informações, estenda a classe `SenhaAtendimento` para que ela esteja em conformidade com a interface `Comparable`. *Sugiro*, novamente, a utilização de testes unitários certificar o correto funcionamento de `compareTo`.

7. Implemente agora a TAD `FilaOrdenada`, utilizando internamente um arranjo, ou lista, que está sempre ordenado, independente das operações realizadas. Teste o programa e veja se as prioridades estão sendo corretamente respeitadas entre as senhas.

8. Classifique assintoticamente as operações da classe `FilaOrdenada`, escrevendo no comentário javadocs.

9. Como tratado em sala de aula, a `FilaOrdenada` tem uma lógica de funcionamento que não propicia o melhor tempo assintótico possível para filas com prioridades. Na classe `FilaPrioritaria`, implemente uma fila de prioridades utilizando a lógica de uma heap para garantir o correto funcionamento bem como o bom desempenho.

10. Classifique assintoticamente as operações da classe `FilaPrioritaria`, escrevendo no comentário javadocs.

2.3 Iteradores de coleções

Iteradores são *design patterns* que permitem a um programador-usuário acessar todos os elementos de qualquer coleção, um de cada vez, sem conhecer especificamente a implementação da coleção, sua estrutura de dados ou sua organização conceitual. Por isso, trata-se de um recurso valioso a qualquer linguagem de programação, e toda coleção que você venha a implementar futuramente deverá prover pelo menos um iterador.

11. Pesquise em livros e na internet sobre iteradores, o conceito, como utilizar em Java e exemplos de implementação de iteradores.

12. Implemente um iterador para cada classe de coleção desse trabalho.

13. Após implementar os iteradores, modifique a lógica de exibição das próximas cinco senhas a serem chamadas, de forma que o programa mostre quais senhas serão chamadas, na ordem correta, mas sem alterar o estado das filas. A utilização de iteradores corretamente implementados é fundamental!