Exercícios sobre TADs e Coleções

Lista05

Informações

- Atividade em duplas;
- Entrega pelo Moodle em <http://trab.dc.unifil.br/moodle/>;

Antes de iniciar as atividades dessa lista de exercícios, é necessária a leitura dos seguintes capítulos de livros:

- Capítulo 10 de CORMEN, Thomas H. et al. Algoritmos: teoria e prática. Rio de Janeiro: Campus, 2012.
- Capítulo 6 de GOODRICH, Michael T.; TAMASSIA, Roberto. Estruturas de dados e algoritmos em Java. 4. ed. Porto Alegre: Bookman, 2007.

Para fazer esta lista, parta do projeto `Colecoes`, incluso no pacote deste roteiro.

1. Abra os fontes das interfaces `Pilha`, `Fila` e `Lista`, e escreva uma especificação adequada em formato *javadocs* para cada operação existente.
2. Escreva um método que utilize a classe `PilhaContigua` para empilhar 10 `Integer` aleatórios, escrevendo-os na tela na ordem em que foram empilhados. Em seguida, desempilhar os 10 elementos, escrevendo-os na tela na ordem de desempilhamento.
3. Escreva um método que utilize a classe `FilaContigua` para enfileirar 10 `Integer` aleatórios, escrevendo-os na tela na ordem em que foram enfileirados. Em seguida, desenfileirar os 10 elementos, escrevendo-os na tela na ordem de desenfileiramento.
4. Escreva um método que crie uma instância de `ListaContigua` com capacidade para armazenar 10 objetos. A seguir, insira 20 `Integer` aleatórios nessa lista. O Java acusou erro de execução? Explique!
5. Implemente a classe `No`, que possui como atributos apenas uma referência a um `Object valor` e a um `No proximo`. Essa classe é utilizada para construir nós encadeados.
6. O projeto `Colecoes` já possui uma implementação funcional para `Pilha`, `Fila` e `Lista`, utilizando arranjos internamente em todos eles. Recrie essas mesmas estruturas de dados abstratas (ou seja, contém exatamente as mesmas operações visíveis externamente) utilizando nós encadeados internamente[1], nomeando-as `PilhaEncadeada`, `FilaEncadeada` e `ListaEncadeada`, de forma que implementem as interfaces adequadas e mantenham a mesma funcionalidade.

[1] Consulte o Cap. 21 de Deitel e Deitel, Java: Como Programar, 10ª Edição, disponível na biblioteca virtual Pearson da UniFil.

7. Implemente a classe `ListaOrdenada`, que possui operações similares às de `Lista`. A todo e qualquer momento que `ListaOrdenada` possuir elementos, eles estarão ordenados. Por causa disso, a operação `inserir` não pode escolher o índice do elemento a ser armazenado, pois o próprio TAD é quem define isso.

   Outra diferença é o fato de que objetos da classe `Object` não podem ser ordenados, já que não são comparáveis por grandeza. Isso quer dizer que é impossível decidir qual `Object` é maior ou menor entre dois deles, operação fundamental em toda ordenação. Por conta disso, a classe `ListaOrdenada` não pode armazenar `Object`, portanto substitua-os por objetos da classe `Comparable`[2], que possuem o método `compareTo(Comparable other)`.

8. Faça uma análise assintótica do desempenho de tempo de execução para cada método público de cada uma das classes TAD de Coleção implementadas, inclusive as versões alternativas com lista encadeada. Coloque essa informação no arquivo texto do projeto e responda também:
   (a) Quais as vantagens e desvantagens das coleções que utilizam arranjos internamente frente às que utilizam listas encadeadas?
   (b) Qual operação é mais eficiente assintoticamente e do ponto de vista do tempo de execução: inserir em uma `Lista` com arranjo ou em uma `Lista` com lista encadeada?
   (c) Se você tiver que fazer muitas buscas por objetos armazenados em uma lista, qual é melhor utilizar, `Lista` ou `ListaOrdenada`? Por quê?

[2]Documentação em: `http://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`. Se precisar, procure também em outras páginas da Internet ou livros de Java sobre a utilização deste recurso.