UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0504/SCC0204 – Programação Orientada a Objetos
Prof. José Fernando Rodrigues Júnior

Lista 09 - Collections

1. (Fila)
Implemente um sistema de controle de atendimento de pacientes em uma clínica médica utilizando uma Queue<String>. Cada novo paciente deve ser adicionado ao final da fila utilizando add. O atendimento ao paciente deve ser feito retirando o paciente da frente da fila usando poll, mostrando qual paciente foi chamado. O sistema deve ainda permitir listar todos os pacientes que estão aguardando, seguindo a ordem da fila. Antes de chamar um paciente, o programa deve verificar se a fila não está vazia. Sabendo que é comum que pacientes cancelem o atendimento a qualquer momento do andamento da fila, determine qual é a melhor versão de Queue a ser usada ArrayList ou LinkedList.

2.(Pilha)
Implemente um programa que verifique se uma sequência de parênteses em uma expressão matemática está balanceada. Para isso, utilize uma Deque<Character> implementada com LinkedList<Character>, tratando-a como uma pilha (usando os métodos push, pop e peek). Cada vez que encontrar um parêntese de abertura (, empilhe-o; cada vez que encontrar um parêntese de fechamento), desempilhe. Se ao final da leitura a pilha estiver vazia e não tiver ocorrido erro de desempilhamento (tentativa de desempilhar uma pilha vazia), a expressão está corretamente balanceada. Caso contrário, a expressão está incorreta. Teste com algumas expressões algébricas e/ou sentenças de linguagem de programação.

3. (Deque)
Implemente um simulador de atendimento de chamadas de emergência. Cada chamada deve ser representada por uma classe Chamado, contendo a descrição do incidente e um nível de prioridade (crítico ou comum). Utilize um Deque<Chamado> (como ArrayDeque) para organizar a fila de atendimento. Chamados críticos devem ser adicionados no início da fila utilizando push, enquanto chamados comuns devem ser adicionados no final da fila com add. O atendimento deve sempre remover do início da fila (remove), de modo que chamados críticos recém-chegados tenham prioridade sobre chamados comuns, mesmo que tenham chegado depois. O sistema deve listar o estado atual da fila e a ordem em que os chamados serão atendidos.

4. (Array + Set)
Escreva uma classe que herda de ArrayList<E>. Além de estender de ArrayList (e herdar suas funcionalidades), sua classe deverá ter um atributo do tipo TreeSet<E>. Toda vez que operações de inserção e remoção na lista forem realizadas, será necessário realizar estas operações também no TreeSet. Desta maneira, sua classe não aceitará elementos duplicados e será capaz de listar os elementos da lista em ordem alfabética.

5.(Set)
Alunos da graduação podem se matricular em diversas disciplinas.
Dadas duas disciplinas, POO e BD, escreva um software que responde às seguintes perguntas:

- Quais alunos estão matriculados simultaneamente nas duas disciplinas?
- Quais alunos estão matriculados somente em uma delas?
- Quais alunos estão matriculados em pelo menos uma das disciplinas?

Resolva com uma das implementações de Set<E> vistas em aula.

6.(Map)
a) Crie um programa que gerencie uma agenda telefônica. Associe um nome (String) a um número de telefone (Long). Permita:

- Adicionar contato;
- Buscar telefone de um contato pelo nome;
- Listar todos os contatos.

Use a classe HashMap<String, Long>.

b) Modifique o código para que a listagem dos contatos ocorra em ordem alfabética. Use a classe TreeMap<String, Long>.

7. (Ordenação)
Considere uma lista de Produto(String desc, float preco, int popularidade). Escreva um método que permite ordenar esta lista por qualquer destes atributos. Use a classe Comparator<E>.