PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Disciplina: Algoritmos e Estruturas de Dados
Entrega: **A lista deverá ser entregue até dia 13/04 às 18h.**

Obs: Cópias serão desconsideradas, ou seja, a nota será igual a 0 (zero).

Lista 7

1) Implemente um programa que permita que uma clínica gerencie sua fila de espera de clientes (i.e., nomes dos clientes). Para tanto, o programa deverá apresentar para o usuário um menu com as seguintes opções:

Menu:
1) Inserir cliente na fila de espera
2) Remover um cliente da fila de espera (Deve ser removido o primeiro cliente que foi inserido na fila)
3) Listar os nomes dos clientes que estão na fila de espera
4) Pesquisar se o cliente está na fila de espera
5) Verificar se a fila está vazia
6) Encerrar o programa

O programa deverá ler a opção informada pelo usuário e executar a operação selecionada. Em seguida o programa deverá apresentar novamente o menu para o usuário, ler e executar a operação selecionada. Esse processo deverá ser repetido até que o usuário digite a opção 5.

Obs: para o desenvolvimento desse programa deve ser utilizada a estrutura de dados Fila. Devem ser criados dois novos métodos:

- pesquisar: recebe como parâmetro uma string. O método deve retorna True caso a string conste na fila e False, caso contrário.
- isVazia: método sem parâmetros. O método deve retornar True caso a fila esteja vazia e False caso contrário.

2) A seguir é apresentado o código de uma Fila de Alunos. Analise, entenda e execute o código para ver os resultados (Para essa questão não é necessário enviar nenhuma resposta)

```
public class Aluno {
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;

    public Aluno(String nome, int idade, double peso, char sexo) {
        this.nome = nome;
        this.idade = idade;
        this.peso = peso;
        this.formando = false;
        this.sexo = sexo;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
```

```
    }

    public double getPeso() {
        return peso;
    }

    public void setPeso(double peso) {
        this.peso = peso;
    }

    public boolean isFormando() {
        return formando;
    }

    public void setFormando(boolean formando) {
        this.formando = formando;
    }

    public char getSexo() {
        return sexo;
    }

    public void setSexo(char sexo) {
        this.sexo = sexo;
    }

    @Override
    public String toString() {
        return "[nome=" + nome + ", idade=" + idade + ", peso=" + peso + ", formando="
+ formando + ", sexo=" + sexo + "]";
    }
}
```

```
   class Fila {

    private Aluno[] array;
    private int primeiro;
    private int ultimo;



    /**
     * Construtor da classe.
     */
    public Fila () {
       this(6);
    }



    /**
     * Construtor da classe.
     * @param tamanho Tamanho da fila.
     */
    public Fila (int tamanho){
       array = new Aluno[tamanho+1];
       primeiro = ultimo = 0;
    }
```

```java
    /**
     * Insere um elemento na ultima posicao da fila.
     * @param x elemento a ser inserido.
     * @throws Exception Se a fila estiver cheia.
     */
    public void inserir(Aluno x) throws Exception {

       //verifica se a fila esta cheia
       if (((ultimo + 1) % array.length) == primeiro) {
          throw new Exception("Erro ao inserir!");
       }

       array[ultimo] = x;
       ultimo = (ultimo + 1) % array.length;
    }

    /**
     * Remove um elemento da primeira posicao da fila
     * @return resp elemento removido.
     * @throws Exception Se a fila estiver vazia.
     */
    public Aluno remover() throws Exception {
       //verifica se a fila esta vazia
       if (primeiro == ultimo) {
          throw new Exception("Erro ao remover!");
       }
       Aluno resp = array[primeiro];
       primeiro = (primeiro + 1) % array.length;
       return resp;
    }

    /**
     * Mostra os nomes dos alunos que estao na Fila.
     */
    public void mostrarNomes (){
       System.out.print("\nNomes alunos na fila:");

       for(int i = primeiro; i != ultimo; i = ((i + 1) % array.length)) {
          System.out.print("(Aluno: "+ array[i].getNome() + ") ");
       }

       System.out.println("\n");
    }

     /**
     * Mostra todos os atributos dos alunos que estao na fila
     */
    public void mostrar(){
        System.out.println("\nFila:");

        for(int i = primeiro; i != ultimo; i = ((i + 1) % array.length)) {
           System.out.println(array[i].toString());
        }

        System.out.println("\n");
     }
}
```

Classe de Teste que gera um objeto Fila e mostra exemplo de inserção, remoção e impressão dos elementos da Fila.

```
public class Teste {
    public static void main(String[] args) throws Exception {
        Fila filaEspera = new Fila(60);

        Aluno a1 = new Aluno("Maria", 30, 70, 'f');
        Aluno a2 = new Aluno("Jose", 20, 60, 'm');
        Aluno a3 = new Aluno("Joao", 40, 80, 'm');

        filaEspera.inserir(a1);
        filaEspera.mostrar();

        filaEspera.inserir(a2);
        filaEspera.mostrarNomes();

        filaEspera.inserir(a3);
        filaEspera.mostrar();

        Aluno temp = filaEspera.remover();
        System.out.println("Aluno removido:"+ temp.getNome());
        filaEspera.mostrar();

        temp = filaEspera.remover();
        System.out.println("Aluno removido:"+ temp.getNome());
        filaEspera.mostrar();
    }
}
```

3) Refaça o programa da questão 1, considerando que será gerada uma lista de Clientes. Considere que cada Cliente terá nome, CPF, endereço e telefone (determine o tipo mais adequado para cada atributo).
Obs: O método pesquisar deve continuar recebendo como parâmetro uma string. Esse método vai pesquisar se tem na Fila algum Cliente com o nome informado como parâmetro.

4) Faça um programa que leia um valor inteiro positivo n, e imprima na tela os n primeiros elementos da sequência de Fibonacci em ordem decrescente. Use uma estrutura de dados pilha.

   Exemplo:
   n = 11
   89, 55, 34, 21, 13, 8, 5, 3, 2, 1, 1

5) Na notação tradicional de expressões aritméticas pode-se usar parênteses para eliminar ambiguidade
   Exemplo:
   A + B * C
   A + (B * C)
   (A+B) * C

   A notação polonesa reversa, dispensa o uso de parênteses. Nessa notação os operadores aparecem após os operandos. Ela é utilizada em vários equipamentos eletrônicos, como calculadores e computadores.

   Exemplo:
   Notação tradicional: A * B – C/D
   Notação polonesa reversa: A B * C D / -

   Notação tradicional: A * ((B-C)/D)
   Notação polonesa reversa: A B C – D / *

Faça um programa que leia uma expressão matemática no formato da notação polonesa reversa, e imprima o resultado da expressão. Utilize a estrutura de dados pilha. Considere que a expressão poderá ter apenas as operações básicas: soma, subtração, multiplicação e divisão. A expressão terá no máximo 100 caracteres.

Exemplo:

Expressão lida: 3572-*4/+

Impressão esperada: 9,25

Para avaliar a expressão deve-se seguir esses passos:

- Percorrer a expressão:
  - Se encontrar um operando, empilhar
  - Se encontrar um operador, desempilhar os dois operandos, aplicar a operação e empilhar o resultado
  - Ao final, o resultado estará no topo da pilha