Exercício: Sistema de Priorização de Contato para Ofertas de Produtos Sustentáveis

Objetivo: Desenvolver um sistema para priorizar o contato de uma empresa que oferece produtos sustentáveis com seus clientes. O sistema organizará e filtrará os clientes com base no valor de compra de produtos sustentáveis e sua elegibilidade para ofertas especiais. Além disso, implementará funcionalidades de busca binária e ordenação utilizando Quick Sort, Bubble Sort ou Insertion Sort.

Descrição do Sistema:

1. **Cadastro de Clientes**:
    - Cada cliente será cadastrado com as seguintes informações:
        - Nome
        - CPF
        - Tipo de produto comprado (por exemplo: painéis solares, lâmpadas LED, carros elétricos)
        - Total de Compras (em valor monetário)
        - Apto para Oferta (Inicialmente, true)
    - O cadastro será feito em uma árvore AVL, onde o critério de organização será o CPF do cliente.
2. **Oferecimento de Produtos Sustentáveis**:
    - Periodicamente, a empresa decide lançar promoções para incentivar os clientes a comprarem mais produtos sustentáveis. O gerente define um valor mínimo de compras para qualificar o cliente para uma oferta.
    - O programa percorre a árvore AVL de clientes e cria uma nova árvore AVL, chamada Árvore de Ofertas, com os clientes que atendem ao critério de compras mínimas.
    - A árvore será organizada de forma decrescente pelo total de compras, priorizando clientes com maior valor de compras para as ofertas.
3. **Simulação de Contato com os Clientes**:
    - A Árvore de Ofertas será utilizada para simular o contato com os clientes:

        - Se o cliente aceitar a oferta (por exemplo, descontos em novos produtos), o atributo "Apto para Oferta" do cliente é alterado para false na árvore de cadastro.
        - Se o cliente não aceitar a oferta, seu registro permanece inalterado.

4. **Consulta ao Cadastro de Clientes**:
    - O sistema permite ao gerente realizar as seguintes consultas:
        - **Consulta por CPF**: Exibe todos os dados do cliente, incluindo tipo de produto comprado e o total de compras realizadas.
        - **Somatório do Total de Compras**: Calcula o valor total gasto por todos os clientes cadastrados.
        - **Quantidade de Clientes com Compras Acima de um Valor Mínimo**: Conta quantos clientes têm compras totais superiores a um valor especificado.
5. **Ordenação e Busca**:
    - O sistema deve permitir a ordenação dos clientes cadastrados utilizando o algoritmo **Quick Sort**, **Bubble Sort** ou **Insertion Sort**.
    - Além disso, o sistema deve permitir realizar uma **busca binária** para localizar um cliente específico pelo CPF.
6. **Encerramento do Programa**:
    - Antes de encerrar o sistema, o programa deve exibir todos os clientes que não aceitaram a oferta ou que não atendem aos critérios para receber ofertas.

Funcionalidades do Sistema em Java:

O sistema deve ter um menu principal com as seguintes opções:

1. **Inscrição de Cliente**: O gerente preenche as informações do cliente e o insere na árvore AVL de cadastro.
2. **Oferta de Produtos Sustentáveis**: O gerente informa o valor mínimo de compras para qualificar o cliente para a oferta. A árvore AVL é percorrida, e uma nova árvore AVL é criada com os clientes aptos.

3. **Consulta ao Cadastro**: O gerente pode acessar um submenu para realizar consultas sobre o cadastro de clientes:
    - Consulta por CPF.
    - Somatório do total de compras de todos os clientes.
    - Quantidade de clientes com compras acima de um valor mínimo.
4. **Ordenação de Clientes**: O gerente pode ordenar a árvore AVL utilizando um dos seguintes algoritmos:
    - Quick Sort
    - Bubble Sort
    - Insertion Sort
5. **Busca Binária**: O gerente pode buscar um cliente na árvore AVL com base no CPF, utilizando a busca binária.
6. **Encerrar o Programa**: Antes de encerrar, o sistema deve exibir todos os clientes que não aceitaram a oferta ou que não atendem aos critérios de elegibilidade.

```
import java.util.Scanner;


public class EnergiaSustentavel {
   public static void main(String[] args) {
      Scanner le = new Scanner(System.in);
      AVL cadastro = new AVL();
      AVL ofertaArvore = new AVL();
      int opcao, op;
      String nome, tipoEnergia, cpf;
      double consumoMensal;

      do {
         System.out.println("0 - Encerrar o programa");
         System.out.println("1 - Inscrição de um cliente");
         System.out.println("2 - Oferta de energia sustentável");
```

```
        System.out.println("3 - Entrar no Submenu");
        System.out.println("4 - Ordenação de clientes");
        System.out.println("5 - Buscar cliente por CPF");
        opcao = le.nextInt();
        switch (opcao) {
          case 0:
            System.out.println("\n\nClientes que não estavam aptos para a oferta:");
            cadastro.imprimirClientesNaoAceitaramOferta(cadastro.root);
            break;
          case 1:
            System.out.print("Digite nome: ");
            le.nextLine(); // Consumir a linha de quebra
            nome = le.nextLine();
            System.out.print("Digite CPF: ");
            cpf = le.next();
            System.out.print("Tipo de energia consumida (solar, eólica, elétrica): ");
            tipoEnergia = le.next();
            System.out.print("Informe consumo mensal de energia (kWh): ");
            consumoMensal = le.nextDouble();

            Cliente cliente = new Cliente(nome, cpf, tipoEnergia, consumoMensal);
            cadastro.root = cadastro.inserir(cadastro.root, cliente);
            break;
          case 2:
            System.out.print("Qual o valor mínimo de consumo de energia (kWh)
para a ofertaArvore? ");
            double minimo = le.nextDouble();
            cadastro.verificaApto(minimo, cadastro.root);
```

```
            cadastro.gerarOferta(cadastro.root, ofertaArvore);
            break;

        case 3:
            do {
                System.out.println("\t1) Consulta cliente buscando pelo CPF");
                System.out.println("\t2) Apresenta o total de consumo de energia de todos os clientes");
                System.out.println("\t3) Apresenta a quantidade de clientes com consumo acima de um valor");
                System.out.println("\t4) Voltar menu principal");
                op = le.nextInt();
                switch (op) {
                    case 1:
                        System.out.print("Informe CPF para consulta: ");
                        cpf = le.next();
                        cadastro.consultarPorCpf(cadastro.root, cpf);
                        break;
                    case 2:
                        cadastro.somarConsumoTodosClientes(cadastro.root);
                        break;
                    case 3:
                        System.out.print("Qual valor mínimo de consumo de energia para consulta? ");
                        minimo = le.nextDouble();
                        cadastro.contarClientesAcimaDeMinimo(cadastro.root, minimo);
                        break;
                    default:
                        System.out.println("Opção inválida");
```

```
        }
    } while (op != 4);
    break;

case 4: // Ordenação de clientes
    System.out.println("\nEscolha um algoritmo de ordenação:");
    System.out.println("1) Quick Sort");
    System.out.println("2) Bubble Sort");
    System.out.println("3) Insertion Sort");
    int opcaoOrdenacao = le.nextInt();
    switch (opcaoOrdenacao) {
        case 1:
            System.out.println("Ordenando com Quick Sort...");
            cadastro.quickSortClientes(cadastro.root);
            break;
        case 2:
            System.out.println("Ordenando com Bubble Sort...");
            cadastro.bubbleSortClientes(cadastro.root);
            break;
        case 3:
            System.out.println("Ordenando com Insertion Sort...");
            cadastro.insertionSortClientes(cadastro.root);
            break;
        default:
            System.out.println("Opção inválida para ordenação");
    }
    break;
```

```
            case 5: // Buscar cliente por CPF
                System.out.print("Informe CPF para busca binária: ");
                cpf = le.next();
                Cliente clienteEncontrado = cadastro.buscaBinaria(cadastro.root, cpf);
                if (clienteEncontrado != null) {
                    System.out.println("Cliente encontrado: " +
clienteEncontrado.getNome());
                } else {
                    System.out.println("Cliente não encontrado.");
                }
                break;

            default:
                System.out.println("Opção inválida");
        }
    } while (opcao != 0);
    le.close();
  }
}
```