JAVA - AULA 13

Preenchendo um arquivo aleatório (CsvPopulator.java)

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;
import javax.swing.JOptionPane;

public class CsvPopulator {
    private static final String filePath = "registros.csv";
    private static final String[] nomes = {
      "Alice", "Bruno", "Carla", "Daniel", "Elisa", "Fernando",
      "Gabriela", "Henrique", "Isabela", "João", "Karen", "Leonardo",
      "Marina", "Nicolas", "Olivia", "Pedro", "Quintino", "Renata",
      "Samuel", "Tatiana", "Ulisses", "Vera", "Wesley", "Xavier", "Yara",
      "Zeca"
    };
    private static final String[] sobreNomes = {
      "Barboza", "Almeida", "Cardozo", "Duarte", "Silva", "Moura",
      "Gomes", "Alves", "Andrade", "Barros", "Batista", "Borges",
      "Campos", "Carvalho", "Castro", "Costa", "Dias", "Freitas",
      "Fernandes", "Gonçalves", "Lima", "Lopes", "Machado", "Marques",
      "Medeiros", "Mendes"
    };
    private static final String[] cidades = {
      "São Paulo", "Rio de Janeiro", "Belo Horizonte", "Curitiba",
      "Porto Alegre", "Salvador", "Fortaleza", "Brasília", "Manaus",
      "Recife"
    };

    public static void popularArquivoCsv() {
        Random random = new Random();
        try (FileWriter writer = new FileWriter(filePath, true)) {
            for (int i = 0; i < 50; i++) {
                String nome = nomes[random.nextInt(nomes.length)]+
                  ' '+sobreNomes[random.nextInt(sobreNomes.length)];
                int idade = random.nextInt(63) + 18; // Idade de 18 a 80
                String cidade = cidades[random.nextInt(cidades.length)];
                writer.append(nome).append(",")
                      .append(String.valueOf(idade)).append(",")
                      .append(cidade).append("\n");
            }
            JOptionPane.showMessageDialog(null,
                  "50 registros aleatórios adicionados com sucesso!");
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                  "Erro ao gravar no arquivo.");
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        popularArquivoCsv(); // Chama a função para popular o arquivo
    }
}
```

CRUD com menu (CrudMenuCsv.java)

```
import javax.swing.JOptionPane;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class CrudMenuCsv {
    private static final String filePath = "registros.csv";

    public static void main(String[] args) {
        while (true) {
            String[] options = {
                  "Adicionar", "Listar", "Atualizar", "Remover", "Sair"};
            int choice = JOptionPane.showOptionDialog(null,
                  "Escolha uma opção", "Menu CRUD",
                   JoptionPane.DEFAULT_OPTION,
                  JOptionPane.INFORMATION_MESSAGE,
                  null, options, options[0]);

            switch (choice) {
                case 0 -> adicionarRegistro();
                case 1 -> listarRegistros();
                case 2 -> atualizarRegistro();
                case 3 -> removerRegistro();
                case 4 -> {
                    JOptionPane.showMessageDialog(null,
                        "Encerrando o programa.");
                    System.exit(0);
                }
                default -> JOptionPane.showMessageDialog(null,
                  "Opção inválida.");
            }
        }
    }

    private static void adicionarRegistro() {
        String nome = JOptionPane.showInputDialog("Digite o nome:");
        String idade = JOptionPane.showInputDialog("Digite a idade:");
        String cidade = JOptionPane.showInputDialog("Digite a cidade:");

        try (FileWriter writer = new FileWriter(filePath, true)) {

writer.append(nome).append(",").append(idade).append(",").append(cidade).
append("\n");
            JOptionPane.showMessageDialog(null,
                  "Registro adicionado com sucesso!");
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                  "Erro ao gravar no arquivo.");
            e.printStackTrace();
        }
    }
```

```
    private static void listarRegistros() {
        try (BufferedReader reader = new BufferedReader(
            new FileReader(filePath))) {
            StringBuilder registros = new StringBuilder("Registros:\n");
            String linha;
            while ((linha = reader.readLine()) != null) {
                registros.append(linha).append("\n");
            }
            JOptionPane.showMessageDialog(null, registros.toString());
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                  "Erro ao ler o arquivo.");
            e.printStackTrace();
        }
    }

    private static void atualizarRegistro() {
        List<String> registros = lerRegistros();
        if (registros.isEmpty()) {
            JOptionPane.showMessageDialog(null,
                  "Nenhum registro encontrado.");
            return;
        }

        String nomeBusca = JoptionPane.showInputDialog(
            "Digite o nome do registro a ser atualizado:");
        boolean encontrado = false;
        for (int i = 0; i < registros.size(); i++) {
            String[] dados = registros.get(i).split(",");
            if (dados[0].equalsIgnoreCase(nomeBusca)) {
                String novoNome = JoptionPane.showInputDialog(
                  "Digite o novo nome:", dados[0]);
                String novaIdade = JoptionPane.showInputDialog(
                  "Digite a nova idade:", dados[1]);
                String novaCidade = JoptionPane.showInputDialog(
                  "Digite a nova cidade:", dados[2]);
                registros.set(i, novoNome + "," + novaIdade + "," +
novaCidade);
                encontrado = true;
                break;
            }
        }

        if (encontrado) {
            escreverRegistros(registros);
            JOptionPane.showMessageDialog(null,
                  "Registro atualizado com sucesso!");
        } else {
            JOptionPane.showMessageDialog(null,
                  "Registro não encontrado.");
        }
    }
```

```
    private static void removerRegistro() {
        List<String> registros = lerRegistros();
        if (registros.isEmpty()) {
            JOptionPane.showMessageDialog(null,
                  "Nenhum registro encontrado.");
            return;
        }

        String nomeBusca = JoptionPane.showInputDialog(
            "Digite o nome do registro a ser removido:");
        boolean encontrado = registros.removeIf(registro ->
registro.split(",")[0].equalsIgnoreCase(nomeBusca));

        if (encontrado) {
            escreverRegistros(registros);
            JOptionPane.showMessageDialog(null,
                  "Registro removido com sucesso!");
        } else {
            JOptionPane.showMessageDialog(null,
                  "Registro não encontrado.");
        }
    }

    private static List<String> lerRegistros() {
        List<String> registros = new ArrayList<>();
        try (BufferedReader reader = new BufferedReader(new
FileReader(filePath))) {
            String linha;
            while ((linha = reader.readLine()) != null) {
                registros.add(linha);
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                  "Erro ao ler o arquivo.");
            e.printStackTrace();
        }
        return registros;
    }

    private static void escreverRegistros(List<String> registros) {
        try (FileWriter writer = new FileWriter(filePath)) {
            for (String registro : registros) {
                writer.append(registro).append("\n");
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                  "Erro ao escrever no arquivo.");
            e.printStackTrace();
        }
    }
}
```