Estácio - Unidade São pedro
RJ 140 Km 2, 512 loja 1, São Pedro da Aldeia - RJ, 28941-182

Desenvolvimento Full Stack
Classe: Missão Prática | Nível 3 | Mundo 3
3º Semestre
Marvin de Almeida Costa

Título da Prática: 2º Procedimento | Alimentando a Base

Objetivos da prática:

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL
- Server na persistência de dados.

Todos os códigos solicitados neste roteiro de aula:

CadastroBD.java

```
package cadastrobd;

/**
 *
 * @author Marvin
 */
import java.util.*;
import cadastro.model.util.ConectorBD;
import cadastrobd.model.PessoaFisica;
```

```
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastro.model.util.SequenceManager;

public class CadastroBD {
    // Instanciando ConectorBD e SequenceManager (simulando a inicialização)
    private static Scanner scanner = new Scanner(System.in);
    private static ConectorBD conectorBD = new ConectorBD();
    private static SequenceManager sequenceManager = new SequenceManager();

    // Instanciando DAOs
    private static PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD,
sequenceManager);
    private static PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD, sequenceManager);

    public static void main(String[] args) {
        int opcao;

        do {
            System.out.println("==================================");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
            System.out.println("3 - Excluir Pessoa");
            System.out.println("4 - Buscar pelo Id");
            System.out.println("5 - Exibir Todos");
            System.out.println("0 - Finalizar Programa");
            System.out.println("==================================");
            opcao = scanner.nextInt();

            switch (opcao) {
                case 1:
                    incluirPessoa();
                    break;
                case 2:
                    alterarPessoa();
                    break;
                case 3:
                    excluirPessoa();
                    break;
                case 4:
                    buscarPeloId();
                    break;
```

```
            case 5:
                exibirTodos();
                break;
            case 0:
                System.out.println("Saindo...");
                break;
            default:
                System.out.println("Opção inválida!");
        }
    } while (opcao!= 0);
}

private static void incluirPessoa() {
    String tipo;
    int idPessoa;
    String nome;
    String cpf;
    String cnpj;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");

    tipo = scanner.next().toUpperCase();

    if (!tipo.equals("J") &&!tipo.equals("F")) {
        System.out.println("Opção inválida!");
        return;
    }

    System.out.println("Insira os dados...");
    System.out.println("Nome:");
    nome = scanner.next();
    scanner.nextLine();

    System.out.println("Logradouro:");
    logradouro = scanner.nextLine();

    System.out.println("Cidade:");
    cidade = scanner.nextLine();
```

```
        boolean isValidInputEstado = false;
        do {
            System.out.println("Estado:");
            estado = scanner.next();
            scanner.nextLine();
            if (estado.length() > 2) {
                System.out.println("Entrada inválida! Digite uma abreviação de estado com no
máximo 2 caracteres.");
            } else {
                isValidInputEstado = true;
            }
        } while (!isValidInputEstado);

        System.out.println("Telefone:");
        telefone = scanner.next();

        System.out.println("Email:");
        email = scanner.next();

        boolean isValidInput = false;

        if (tipo.equals("J")) {
            do {
                System.out.println("CNPJ:");
                cnpj = scanner.next();

                if (cnpj.length() > 14) {
                    System.out.println("Entrada inválida! Digite um máximo de 14 caracteres.");
                } else {
                    isValidInput = true;
                }
            } while (!isValidInput);

            PessoaJuridica pessoaJuridica = new PessoaJuridica(0, nome, logradouro, cidade,
estado, telefone, email, cnpj);
            pessoaJuridicaDAO.incluir(pessoaJuridica);
        }

        if (tipo.equals("F")) {
            do {
                System.out.println("CPF:");
                cpf = scanner.next();

                if (cpf.length() > 11) {
```

```
                System.out.println("Entrada inválida! Digite um máximo de 11 caracteres.");
            } else {
                isValidInput = true;
            }
        } while (!isValidInput);


        PessoaFisica pessoaFisica = new PessoaFisica(0, nome, logradouro, cidade, estado,
telefone, email, cpf);
        pessoaFisicaDAO.incluir(pessoaFisica);
    }
  }

  private static void alterarPessoa() {
    String tipo;
    int id;
    String nome;
    String cpf;
    String cnpj;
    PessoaJuridica pessoaJuridica = null;
    PessoaFisica pessoaFisica = null;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    tipo = scanner.next().toUpperCase();

    if (!tipo.equals("J") &&!tipo.equals("F")) {
        System.out.println("Opção inválida!");
        return;
    }

    System.out.println("Digite o id da Pessoa:");
    id = scanner.nextInt();

    if (tipo.equals("J")) {
        pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
        if (pessoaJuridica == null) {
            System.out.println("Pessoa não encontrada");
            return;
        }
```

```
}

if (tipo.equals("F")) {
    pessoaFisica = pessoaFisicaDAO.getPessoa(id);
    if (pessoaFisica == null) {
        System.out.println("Pessoa não encontrada");
        return;
    }
}

System.out.println("Insira os dados...");

if (tipo.equals("J")) {
    System.out.printf("Nome (%s):", pessoaJuridica.getNome());
    System.out.println("");
}

if (tipo.equals("F")) {
    System.out.printf("Nome (%s):", pessoaFisica.getNome());
    System.out.println("");
}

nome = scanner.next();
scanner.nextLine();

if (tipo.equals("J")) {
    System.out.printf("Logradouro (%s):", pessoaJuridica.getLogradouro());
    System.out.println("");
}

if (tipo.equals("F")) {
    System.out.printf("Logradouro (%s):", pessoaFisica.getLogradouro());
    System.out.println("");
}

logradouro = scanner.nextLine();

if (tipo.equals("J")) {
    System.out.printf("Cidade (%s):", pessoaJuridica.getCidade());
    System.out.println("");
}

if (tipo.equals("F")) {
    System.out.printf("Cidade (%s):", pessoaFisica.getCidade());
```

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

        cidade = scanner.nextLine();

        if (tipo.equals("J")) {
            System.out.printf("Estado (%s):", pessoaJuridica.getEstado());
            System.out.println("");
        }

        if (tipo.equals("F")) {
            System.out.printf("Estado (%s):", pessoaFisica.getEstado());
            System.out.println("");
        }

        boolean isValidInputEstado = false;
        do {
            estado = scanner.next();
            scanner.nextLine();
            if (estado.length() > 2) {
                System.out.println("Entrada inválida! Digite uma abreviação de estado com no
máximo 2 caracteres.");
            } else {
                isValidInputEstado = true;
            }
        } while (!isValidInputEstado);

        if (tipo.equals("J")) {
            System.out.printf("Telefone (%s):", pessoaJuridica.getTelefone());
        }

        if (tipo.equals("F")) {
            System.out.printf("Telefone (%s):", pessoaFisica.getTelefone());
        }

        System.out.println("");
        telefone = scanner.next();

        if (tipo.equals("J")) {
            System.out.printf("Email (%s):", pessoaJuridica.getEmail());
        }

        if (tipo.equals("F")) {
            System.out.printf("Email (%s):", pessoaFisica.getEmail());
```

```
        }

        System.out.println("");
        email = scanner.next();

        boolean isValidInput = false;

        if (tipo.equals("J")) {
            do {
                System.out.printf("CNPJ (%s):", pessoaJuridica.getCnpj());
                System.out.println("");
                cnpj = scanner.next();

                if (cnpj.length() > 14) {
                    System.out.println("Entrada inválida! Digite um máximo de 14 caracteres.");
                } else {
                    isValidInput = true;
                }
            } while (!isValidInput);

            PessoaJuridica pessoaJuridicaData = new PessoaJuridica(id, nome, logradouro, cidade,
estado, telefone, email, cnpj);
            pessoaJuridicaDAO.alterar(pessoaJuridicaData);
        }

        if (tipo.equals("F")) {
            do {
                System.out.printf("CPF (%s):", pessoaFisica.getCpf());
                System.out.println("");
                cpf = scanner.next();

                if (cpf.length() > 11) {
                    System.out.println("Entrada inválida! Digite um máximo de 11 caracteres.");
                } else {
                    isValidInput = true;
                }
            } while (!isValidInput);

            PessoaFisica pessoaFisicaData = new PessoaFisica(id, nome, logradouro, cidade,
estado, telefone, email, cpf);
            pessoaFisicaDAO.alterar(pessoaFisicaData);
        }
    }
```

```
private static void buscarPeloId() {
    String tipo;
    int id;

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    tipo = scanner.next().toUpperCase();

    if (!tipo.equals("J") &&!tipo.equals("F")) {
        System.out.println("Opção inválida!");
        return;
    }

    System.out.println("Digite o id da Pessoa:");
    id = scanner.nextInt();

    if (tipo.equals("J")) {
        PessoaJuridica pessoa = pessoaJuridicaDAO.getPessoa(id);
        if (pessoa == null) {
            System.out.println("Pessoa não encontrada");
        } else {
            System.out.println("Pessoa Juridica:");
            pessoa.exibir();
        }
    }

    if (tipo.equals("F")) {
        PessoaFisica pessoa = pessoaFisicaDAO.getPessoa(id);
        if (pessoa == null) {
            System.out.println("Pessoa não encontrada");
        } else {
            System.out.println("Pessoa Fisica:");
            pessoa.exibir();
        }
    }
}

private static void excluirPessoa() {
    String tipo;
    int id;

    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    tipo = scanner.next().toUpperCase();

    if (!tipo.equals("J") &&!tipo.equals("F")) {
```

```
            System.out.println("Opção inválida!");
            return;
        }

        System.out.println("Digite o id da Pessoa:");
        id = scanner.nextInt();

        if (tipo.equals("J")) {
            PessoaJuridica pessoa = pessoaJuridicaDAO.getPessoa(id);
            if (pessoa == null) {
                System.out.println("Pessoa não encontrada");
            } else {
                pessoaJuridicaDAO.excluir(id);
            }
        }

        if (tipo.equals("F")) {
            PessoaFisica pessoa = pessoaFisicaDAO.getPessoa(id);
            if (pessoa == null) {
                System.out.println("Pessoa não encontrada");
            } else {
                pessoaFisicaDAO.excluir(id);
            }
        }
    }

    private static void exibirTodos() {
        System.out.println("Dados de Pessoas Fisicas:");
        List<PessoaFisica> todasPessoasFisicas = pessoaFisicaDAO.getPessoas();
        if (!todasPessoasFisicas.isEmpty()) {
            for (PessoaFisica pf : todasPessoasFisicas) {
                pf.exibir();
                System.out.println("");
            }
        } else {
            System.out.println("Nenhuma Pessoa Fisica");
        }

        System.out.println("----------------------------");

        System.out.println("Dados de Pessoas Juridicas:");

        List<PessoaJuridica> todasPessoasJuridicas = pessoaJuridicaDAO.getPessoas();
        if (!todasPessoasJuridicas.isEmpty()) {
```

```
            for (PessoaJuridica pf : todasPessoasJuridicas) {
                pf.exibir();
                System.out.println("");
            }
        } else {
            System.out.println("Nenhuma Pessoa Juridica");
        }
    }
}
```

Pessoa.java

```
package cadastrobd.model;

/**
 *
 * @author marvin
 */
public class Pessoa {
    protected int idPessoa;
    protected String nome;
    protected String logradouro;
    protected String cidade;
    protected String estado;
    protected String telefone;
    protected String email;

    // Construtor padrão
    public Pessoa() {}

    // Construtor completo
    public Pessoa(int idPessoa, String nome, String logradouro, String cidade, String estado,
String telefone, String email) {
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public void setId(int id) {
        this.idPessoa = id;
```

```
}

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

// Método para exibir dados
public void exibir() {
    System.out.println("ID: " + idPessoa);
    System.out.println("Nome: " + nome);
    System.out.println("Logradouro: " + logradouro);
    System.out.println("Cidade: " + cidade);
    System.out.println("Estado: " + estado);
    System.out.println("Telefone: " + telefone);
    System.out.println("Email: " + email);
}

public int getId() {
    return idPessoa;
}

public String getNome() {
    return nome;
}

public String getLogradouro() {
    return logradouro;
}

public String getCidade() {
    return cidade;
}

public String getEstado() {
    return estado;
}

public String getTelefone() {
    return telefone;
}

public String getEmail() {
    return email;
}
```

```
}
```

PessoaFisica.java

```
package cadastrobd.model;

/**
 *
 * @author marvin
 */
public class PessoaFisica extends Pessoa {
    private String cpf;

    // Construtor padrão
    public PessoaFisica() {
        super();
    }

    // Construtor completo
    public PessoaFisica(int idPessoa, String nome, String logradouro, String cidade, String
estado, String telefone, String email, String cpf) {
        super(idPessoa, nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
}
```

PessoaFisicaDAO.java

```
package cadastrobd.model;

/**
 *
 * @author marvin
 */
import java.util.*;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Connection;

public class PessoaFisicaDAO {
   private ConectorBD conector;
   private SequenceManager sequenceManager;

   public PessoaFisicaDAO(ConectorBD conector, SequenceManager sequenceManager) {
      this.conector = conector;
      this.sequenceManager = sequenceManager;
   }

   // Método para obter uma pessoa física pelo ID
   public PessoaFisica getPessoa(int id) {
      String sql = "SELECT * FROM Pessoa JOIN PessoaFisica ON Pessoa.idPessoa =
PessoaFisica.idPessoa WHERE Pessoa.idPessoa = ?";
      try (
            Connection conn = ConectorBD.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
         ) {
         pstmt.setInt(1, id);
         ResultSet rs = pstmt.executeQuery();
         if (rs.next()) {
            String nome = rs.getString("nome");
            String logradouro = rs.getString("logradouro");
            String cidade = rs.getString("cidade");
            String estado = rs.getString("estado");
            String telefone = rs.getString("telefone");
            String email = rs.getString("email");
```

```
            String cpf = rs.getString("cpf");
            return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
        }
    } catch (SQLException e) { // Catch SQLException specifically first
        e.printStackTrace();
    } catch (Exception e) { // Then catch the more general Exception
        e.printStackTrace();
    }
    return null;
}

// Método para obter todas as pessoas físicas
public List<PessoaFisica> getPessoas() {
    List<PessoaFisica> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa JOIN PessoaFisica ON Pessoa.idPessoa =
PessoaFisica.idPessoa";
    try (
            Connection conn = ConectorBD.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery()
        ) {
        while (rs.next()) {
            int id = rs.getInt("idPessoa");
            String nome = rs.getString("nome");
            String logradouro = rs.getString("logradouro");
            String cidade = rs.getString("cidade");
            String estado = rs.getString("estado");
            String telefone = rs.getString("telefone");
            String email = rs.getString("email");
            String cpf = rs.getString("cpf");
            pessoas.add(new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email,
cpf));
        }
    } catch (SQLException e) { // Catch SQLException specifically first
        e.printStackTrace();
    } catch (Exception e) { // Then catch the more general Exception
        e.printStackTrace();
    }
    return pessoas;
}

// Método para incluir uma nova pessoa física
public int incluir(PessoaFisica pessoaFisica) {
```

```
        String sqlInsertPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf) VALUES (?,
?)";
        String sqlMaxIdPessoa = "SELECT MAX(idPessoa) AS MaxId FROM Pessoa";

        try {
            // Insert Pessoa
            try (Connection conn = ConectorBD.getConnection();
                 PreparedStatement pstmt = conn.prepareStatement(sqlInsertPessoa)) {

                pstmt.setString(1, pessoaFisica.getNome());
                pstmt.setString(2, pessoaFisica.getLogradouro());
                pstmt.setString(3, pessoaFisica.getCidade());
                pstmt.setString(4, pessoaFisica.getEstado());
                pstmt.setString(5, pessoaFisica.getTelefone());
                pstmt.setString(6, pessoaFisica.getEmail());
                pstmt.executeUpdate();


                PreparedStatement pstmtMaxId = conn.prepareStatement(sqlMaxIdPessoa);
                ResultSet rsMaxId = pstmtMaxId.executeQuery();
                if (rsMaxId.next()) {
                    int lastInsertedId = rsMaxId.getInt("MaxId");
                    // Now use this ID to insert PessoaFisica
                    try (PreparedStatement pstmt2 = conn.prepareStatement(sqlInsertPessoaFisica)) {
                        pstmt2.setInt(1, lastInsertedId);
                        pstmt2.setString(2, pessoaFisica.getCpf());
                        pstmt2.executeUpdate();

                        System.out.println("Pessoa Física incluída com sucesso.");
                        return lastInsertedId;
                    }
                } else {
                    System.out.println("No se encontró el último ID insertado.");
                }

            }
        } catch (SQLException e) {
            e.printStackTrace();
            // Handle exception, such as logging the error or informing the user
        }
        return 0;
    }
```

```
    // Método para alterar os dados de uma pessoa física
    public void alterar(PessoaFisica pessoaFisica) {
        String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?,
telefone=?, email=? WHERE idPessoa=?";
        String sql2 = "UPDATE PessoaFisica SET cpf=? WHERE idPessoa=?";

        try {
            // Insert Pessoa
            try (Connection conn = ConectorBD.getConnection();
                PreparedStatement pstmt = conn.prepareStatement(sql);
                PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {

                pstmt.setString(1, pessoaFisica.getNome());
                pstmt.setString(2, pessoaFisica.getLogradouro());
                pstmt.setString(3, pessoaFisica.getCidade());
                pstmt.setString(4, pessoaFisica.getEstado());
                pstmt.setString(5, pessoaFisica.getTelefone());
                pstmt.setString(6, pessoaFisica.getEmail());
                pstmt.setInt(7, pessoaFisica.getId());
                pstmt.executeUpdate();

                pstmt2.setString(1, pessoaFisica.getCpf());
                pstmt2.setInt(2, pessoaFisica.getId());
                pstmt2.executeUpdate();

                System.out.println("Dados da pessoa fisica alterados com sucesso.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            // Handle exception, such as logging the error or informing the user
        }
    }

    // Método para excluir uma pessoa física
    public void excluir(int id) {
        String sql = "DELETE FROM PessoaFisica WHERE idPessoa=?";
        String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";

        try (
            Connection conn = ConectorBD.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
            pstmt.setInt(1, id);
```

```
            pstmt.executeUpdate();

            pstmt2.setInt(1, id);
            pstmt2.executeUpdate();

            System.out.println("Pessoa Física excluída com sucesso.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

PessoaJuridica.java

```
package cadastrobd.model;

/**
 *
 * @author marvin
 */
public class PessoaJuridica extends Pessoa {
    private String cnpj;

    // Construtor padrão
    public PessoaJuridica() {
        super();
    }

    // Construtor completo
    public PessoaJuridica(int id, String nome, String logradouro, String cidade, String estado,
String telefone, String email, String cnpj) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cnpj = cnpj;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

    public String getCnpj() {
        return cnpj;
```

```
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
}
```

PessoaJuridicaDAO.java

```
package cadastrobd.model;

/**
 *
 * @author marvin
 */
import java.util.*;
import java.util.List;
import cadastro.model.util.ConectorBD;
import cadastro.model.util.SequenceManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Connection;

public class PessoaJuridicaDAO {
    private ConectorBD conector;
    private SequenceManager sequenceManager;

    public PessoaJuridicaDAO(ConectorBD conector, SequenceManager sequenceManager) {
        this.conector = conector;
        this.sequenceManager = sequenceManager;
    }

    // Método para obter uma pessoa física pelo ID
    public PessoaJuridica getPessoa(int id) {
        String sql = "SELECT * FROM Pessoa JOIN PessoaJuridica ON Pessoa.idPessoa =
PessoaJuridica.idPessoa WHERE Pessoa.idPessoa = ?";
        try (
                Connection conn = ConectorBD.getConnection();
                PreparedStatement pstmt = conn.prepareStatement(sql);
            ) {
            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
```

```
            String nome = rs.getString("nome");
            String logradouro = rs.getString("logradouro");
            String cidade = rs.getString("cidade");
            String estado = rs.getString("estado");
            String telefone = rs.getString("telefone");
            String email = rs.getString("email");
            String cnpj = rs.getString("cnpj");
            return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email,
cnpj);
        }
    } catch (SQLException e) { // Catch SQLException specifically first
        e.printStackTrace();
    } catch (Exception e) { // Then catch the more general Exception
        e.printStackTrace();
    }
    return null;
  }

  // Método para obter todas as pessoas físicas
  public List<PessoaJuridica> getPessoas() {
    List<PessoaJuridica> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa JOIN PessoaJuridica ON Pessoa.idPessoa =
PessoaJuridica.idPessoa";
    try (
            Connection conn = ConectorBD.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            ResultSet rs = pstmt.executeQuery()
        ) {
        while (rs.next()) {
            int id = rs.getInt("idPessoa");
            String nome = rs.getString("nome");
            String logradouro = rs.getString("logradouro");
            String cidade = rs.getString("cidade");
            String estado = rs.getString("estado");
            String telefone = rs.getString("telefone");
            String email = rs.getString("email");
            String cnpj = rs.getString("cnpj");
            pessoas.add(new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone,
email, cnpj));
        }
    } catch (SQLException e) { // Catch SQLException specifically first
        e.printStackTrace();
    } catch (Exception e) { // Then catch the more general Exception
        e.printStackTrace();
```

```
        }
        return pessoas;
    }

    // Método para incluir uma nova pessoa física
    public int incluir(PessoaJuridica pessoaJuridica) {
        String sqlInsertPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, cnpj) VALUES
(?, ?)";
        String sqlMaxIdPessoa = "SELECT MAX(idPessoa) AS MaxId FROM Pessoa";

        try {
            // Insert Pessoa
            try (Connection conn = ConectorBD.getConnection();
                 PreparedStatement pstmt = conn.prepareStatement(sqlInsertPessoa)) {

                pstmt.setString(1, pessoaJuridica.getNome());
                pstmt.setString(2, pessoaJuridica.getLogradouro());
                pstmt.setString(3, pessoaJuridica.getCidade());
                pstmt.setString(4, pessoaJuridica.getEstado());
                pstmt.setString(5, pessoaJuridica.getTelefone());
                pstmt.setString(6, pessoaJuridica.getEmail());
                pstmt.executeUpdate();


                PreparedStatement pstmtMaxId = conn.prepareStatement(sqlMaxIdPessoa);
                ResultSet rsMaxId = pstmtMaxId.executeQuery();
                if (rsMaxId.next()) {
                    int lastInsertedId = rsMaxId.getInt("MaxId");
                    // Now use this ID to insert PessoaJuridica
                    try (PreparedStatement pstmt2 = conn.prepareStatement(sqlInsertPessoaJuridica))
{
                        pstmt2.setInt(1, lastInsertedId);
                        pstmt2.setString(2, pessoaJuridica.getCnpj());
                        pstmt2.executeUpdate();

                        System.out.println("Pessoa Jurídica incluída com sucesso.");
                        return lastInsertedId;
                    }
                } else {
                    System.out.println("No se encontró el último ID insertado.");
                }
```

```
            }
        } catch (SQLException e) {
            e.printStackTrace();
            // Handle exception, such as logging the error or informing the user
        }
        return 0;
    }

    // Método para alterar os dados de uma pessoa física
    public void alterar(PessoaJuridica pessoaJuridica) {
        String sql = "UPDATE Pessoa SET nome=?, logradouro=?, cidade=?, estado=?,
telefone=?, email=? WHERE idPessoa=?";
        String sql2 = "UPDATE PessoaJuridica SET cnpj=? WHERE idPessoa=?";

        try {
            // Insert Pessoa
            try (Connection conn = ConectorBD.getConnection();
                PreparedStatement pstmt = conn.prepareStatement(sql);
                PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {

                pstmt.setString(1, pessoaJuridica.getNome());
                pstmt.setString(2, pessoaJuridica.getLogradouro());
                pstmt.setString(3, pessoaJuridica.getCidade());
                pstmt.setString(4, pessoaJuridica.getEstado());
                pstmt.setString(5, pessoaJuridica.getTelefone());
                pstmt.setString(6, pessoaJuridica.getEmail());
                pstmt.setInt(7, pessoaJuridica.getId());
                pstmt.executeUpdate();

                pstmt2.setString(1, pessoaJuridica.getCnpj());
                pstmt2.setInt(2, pessoaJuridica.getId());
                pstmt2.executeUpdate();

                System.out.println("Dados da pessoa jurídica alterados com sucesso.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            // Handle exception, such as logging the error or informing the user
        }
    }

    // Método para excluir uma pessoa física
    public void excluir(int id) {
        String sql = "DELETE FROM PessoaJuridica WHERE idPessoa=?";
```

```
        String sql2 = "DELETE FROM Pessoa WHERE idPessoa=?";

        try (
            Connection conn = ConectorBD.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);
            PreparedStatement pstmt2 = conn.prepareStatement(sql2)) {
            pstmt.setInt(1, id);
            pstmt.executeUpdate();

            pstmt2.setInt(1, id);
            pstmt2.executeUpdate();

            System.out.println("Pessoa Jurídica excluída com sucesso.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

ConectorBD.java

```
package cadastro.model.util;

/**
 *
 * @author marvin
 */
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConectorBD {

    // Método para obter uma conexão com o banco de dados
    public static Connection getConnection() throws SQLException {
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=Loja3;trustServerCertificate=true";

        String usuario = "sa";
        String senha = "1234";
```

```
        return DriverManager.getConnection(url, usuario, senha);
    }

    // Método para obter um PreparedStatement
    public static PreparedStatement getPrepared(String sql) throws Exception {
        try (Connection connection = getConnection()) {
            return connection.prepareStatement(sql);
        }
    }

    // Método para obter um ResultSet
    public static ResultSet getSelect(String sql) throws Exception {
        try (Connection connection = getConnection(); Statement statement =
connection.createStatement()) {
            return statement.executeQuery(sql);
        }
    }

    // Métodos sobrecarregados para fechar Statement, ResultSet e Connection
    public static void close(Statement statement) {
        if (statement != null) {
            try {
                statement.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public static void close(ResultSet resultSet) {
        if (resultSet != null) {
            try {
                resultSet.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public static void close(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (Exception e) {
```

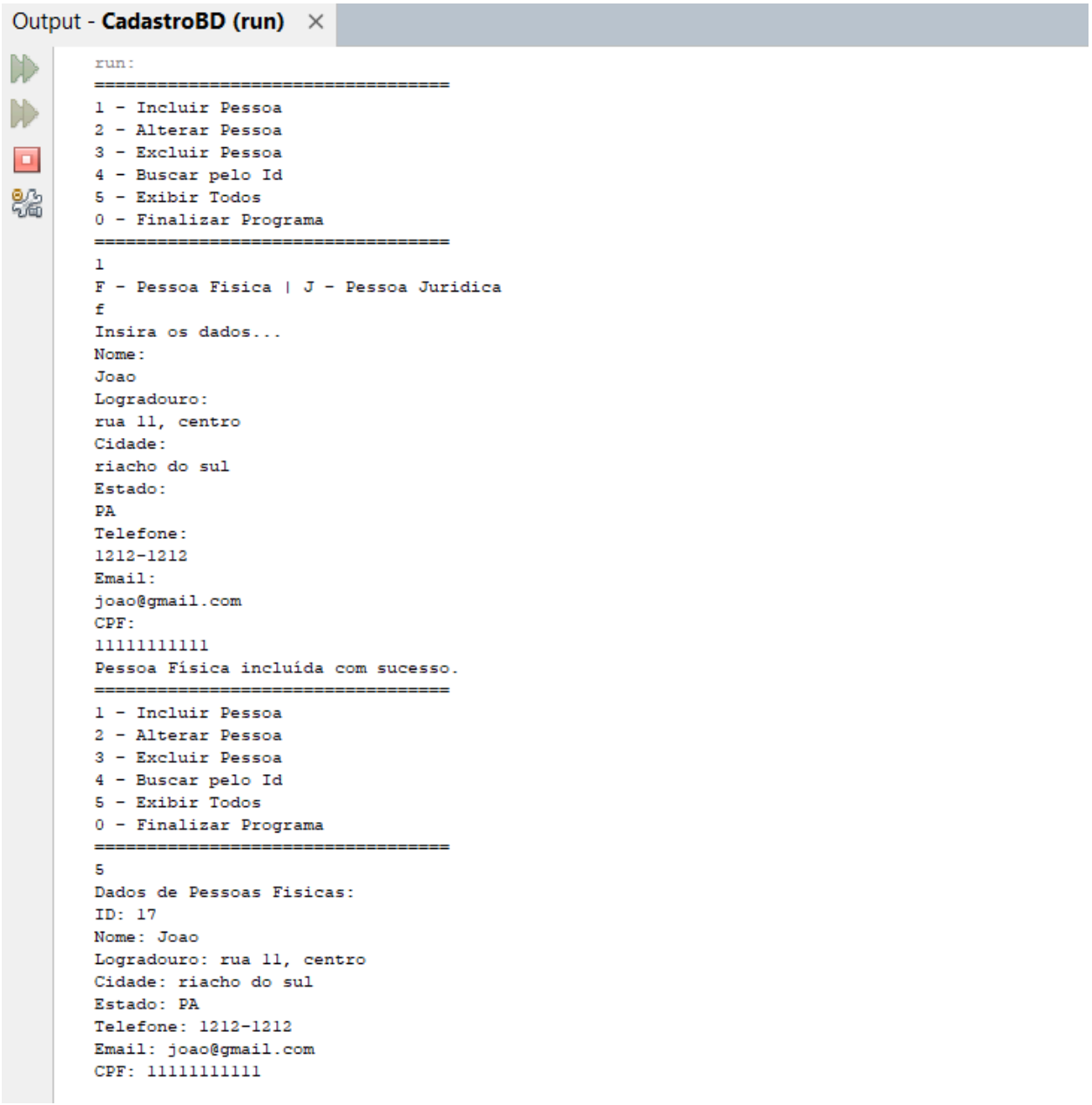
```
            e.printStackTrace();
        }
    }
  }
}
```

SequenceManager.java

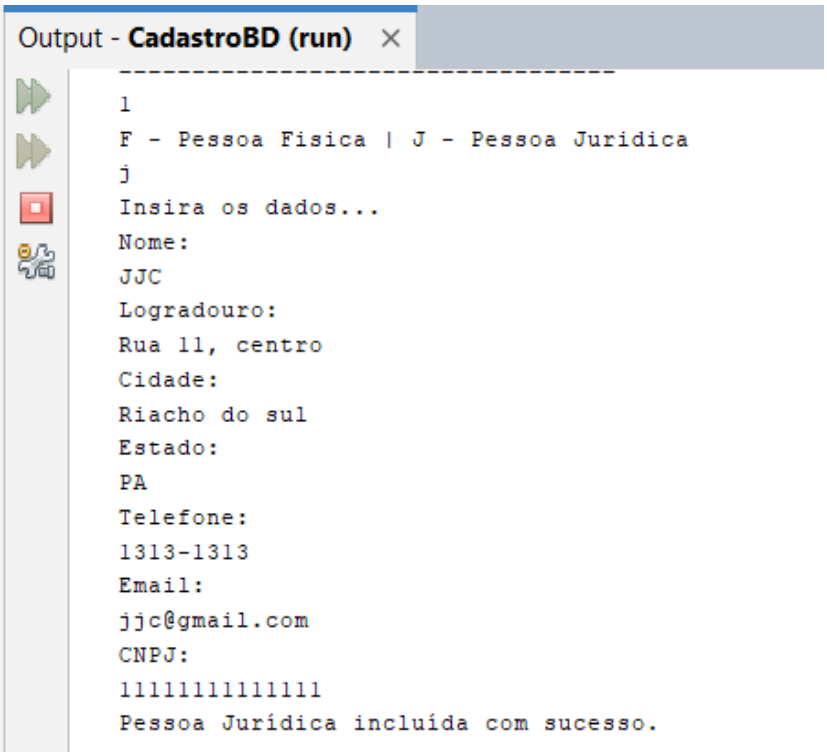
```
package cadastro.model.util;

/**
 *
 * @author marvin
 */
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;

public class SequenceManager {

   // Método para obter o próximo valor de uma sequência
   public static long getValue(String sequenceName) throws SQLException {
      try (Connection connection = ConectorBD.getConnection();
           PreparedStatement preparedStatement = connection.prepareStatement("SELECT
nextval('" + sequenceName + "')")) {
         try (ResultSet resultSet = preparedStatement.executeQuery()) {
            if (resultSet.next()) {
               long result = resultSet.getLong(1);
               return result;
            } else {
               throw new SQLException("No result found");
            }
         }
      } catch (SQLException e) {
         throw new RuntimeException(e);
      }
   }
}
```

Os resultados da execução dos códigos também devem ser apresentados;

Incluir Pessoa Física



Incluir Pessoa Jurídica




```
==================================
5
Dados de Pessoas Fisicas:
ID: 17
Nome: Joao
Logradouro: rua 11, centro
Cidade: riacho do sul
Estado: PA
Telefone: 1212-1212
Email: joao@gmail.com
CPF: 11111111111

------------------------------
Dados de Pessoas Juridicas:
ID: 18
Nome: JJC
Logradouro: Rua 11, centro
Cidade: Riacho do sul
Estado: PA
Telefone: 1313-1313
Email: jjc@gmail.com
CNPJ: 11111111111111

==================================
```

Exibir todos

```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
5
Dados de Pessoas Fisicas:
ID: 17
Nome: Joao
Logradouro: rua 11, centro
Cidade: riacho do sul
Estado: PA
Telefone: 1212-1212
Email: joao@gmail.com
CPF: 11111111111

------------------------------
Dados de Pessoas Juridicas:
ID: 18
Nome: JJC
Logradouro: Rua 11, centro
Cidade: Riacho do sul
Estado: PA
Telefone: 1313-1313
Email: jjc@gmail.com
CNPJ: 11111111111111
```

Buscar pelo Id Pessoa Física

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
4
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
17
Pessoa Fisica:
ID: 17
Nome: Joao
Logradouro: rua 11, centro
Cidade: riacho do sul
Estado: PA
Telefone: 1212-1212
Email: joao@gmail.com
CPF: 11111111111
----------------------------------
```

Buscar pelo Id Pessoa Jurídica

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
4
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id da Pessoa:
18
Pessoa Juridica:
ID: 18
Nome: JJC
Logradouro: Rua 11, centro
Cidade: Riacho do sul
Estado: PA
Telefone: 1313-1313
Email: jjc@gmail.com
CNPJ: 11111111111111
```

Buscar pelo Id Pessoa não encontrada

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
4
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
245
Pessoa não encontrada
==================================
```

Alterar Pessoa Física

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
2
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
17
Insira os dados...
Nome (Joao):
Joao edit
Logradouro (rua 11, centro):
Rua 12, centro
Cidade (riacho do sul):
Riacho do Sul E
Estado (PA):
PE
Telefone (1212-1212):
1414-1414
Email (joao@gmail.com):
joaoedit@gmail.com
CPF (11111111111):
222222222222
Entrada inválida! Digite um máximo de 11 caracteres.
CPF (11111111111):
22222222222
Dados da pessoa fisica alterados com sucesso.
```

```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
4
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
17
Pessoa Fisica:
ID: 17
Nome: Joao
Logradouro: Rua 12, centro
Cidade: Riacho do Sul E
Estado: PE
Telefone: 1414-1414
Email: joaoedit@gmail.com
CPF: 22222222222
```

Alterar Pessoa Jurídica

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
2
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id da Pessoa:
18
Insira os dados...
Nome (JJC):
JJC EDIT
Logradouro (Rua 11, centro):
Rua 13, centro
Cidade (Riacho do sul):
Riacho do SUL edit
Estado (PA):
PO
Telefone (1313-1313):
1515-1515
Email (jjc@gmail.com):
jjcedit@gmail.com
CNPJ (11111111111111):
22222222222222
Dados da pessoa jurídica alterados com sucesso.
```

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
4
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id da Pessoa:
18
Pessoa Juridica:
ID: 18
Nome: JJC
Logradouro: Rua 13, centro
Cidade: Riacho do SUL edit
Estado: PO
Telefone: 1515-1515
Email: jjcedit@gmail.com
CNPJ: 22222222222222
```

Alterar Pessoa - Pessoa não encontrada

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
3
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
245
Pessoa não encontrada
```

Excluir Pessoa Jurídica - Física

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
3
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id da Pessoa:
17
Pessoa Física excluída com sucesso.
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
3
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id da Pessoa:
18
Pessoa Jurídica excluída com sucesso.
```

Exibir todos - Nenhuma Pessoa

```
==================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==================================
5
Dados de Pessoas Fisicas:
Nenhuma Pessoa Fisica
----------------------------
Dados de Pessoas Juridicas:
Nenhuma Pessoa Juridica
```

Análise e Conclusão:

Quais as diferenças entre a persistência em arquivo e a persistência em banco de Dados?

A persistência de dados é um conceito fundamental na computação, referindo-se à capacidade de manter os dados disponíveis mesmo após o encerramento de uma sessão ou aplicativo. Existem duas abordagens principais para a persistência de dados: persistência em arquivo e persistência em banco de dados. Cada uma tem suas características, vantagens e desvantagens. Vamos explorar as diferenças entre elas:

Persistência em Arquivo

- Armazenamento: Os dados são salvos diretamente em arquivos no sistema de arquivos do servidor ou cliente.

- Gerenciamento de Dados: O desenvolvedor é responsável por toda a lógica de gerenciamento de dados, incluindo a criação, leitura, atualização e exclusão (CRUD) dos dados.

- Escalabilidade: A escalabilidade horizontal (adicionar mais máquinas ao sistema) pode ser limitada sem um mecanismo de gerenciamento de banco de dados.
- Concorrência: Sem um mecanismo adequado, a concorrência pode levar a problemas como condições de corrida e bloqueios.

- Segurança: A segurança dos dados depende das permissões do sistema operacional e das medidas de segurança implementadas pelo desenvolvedor.

Persistência em Banco de Dados

- Armazenamento: Os dados são armazenados em um sistema gerenciado que organiza, recupera e manipula os dados de maneira eficiente.

- Gerenciamento de Dados: O banco de dados gerencia automaticamente a lógica CRUD, garantindo integridade dos dados e isolamento de transações.

- Escalabilidade: Oferece melhorias significativas na escalabilidade, tanto vertical quanto horizontal, com suporte a replicação e sharding.

- Concorrência: Gerencia a concorrência de forma eficaz, evitando condições de corrida e bloqueios através de mecanismos como locks e transactions.

- Segurança: Fornece recursos robustos de segurança, incluindo autenticação, autorização e criptografia.

Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

O uso de operadores lambda no Java introduziu uma maneira concisa e expressiva de trabalhar com interfaces funcionais, melhorando significativamente a legibilidade e a eficiência do código em várias situações, incluindo a manipulação de coleções e a execução de tarefas assíncronas. A partir da versão Java 8, os operadores lambda permitiram que os desenvolvedores escrevessem código mais limpo e compacto ao lidar com entidades e seus valores.

Simplificação na Impressão de Valores

Antes do Java 8, para imprimir os valores de uma lista ou coleção, você precisava implementar uma interface funcional (como Iterator ou ListIterator) ou usar um loop for-each. Com o advento dos operadores lambda, essa tarefa se tornou muito mais simples e direta.

Exemplo sem Lambda:

```
List<String> nomes = Arrays.asList("Ana", "Bruno", "Carlos");

Usando Iterator

Iterator<String> iterator = nomes.iterator();
while (iterator.hasNext()) {
   System.out.println(iterator.next());
}

Usando ListIterator

ListIterator<String> listIterator = nomes.listIterator();
while (listIterator.hasNext()) {
   System.out.println(listIterator.next());
}
```

Exemplo com Lambda:

```
List<String> nomes = Arrays.asList("Ana", "Bruno", "Carlos");

Usando forEach com referência de método

nomes.forEach(System.out::println);

Usando forEach com lambda

nomes.forEach(nome -> System.out.println(nome));
```

No exemplo acima, o uso de System.out::println é uma referência de método, que é uma forma especializada de lambda onde o operador :: é usado para indicar que o método println do objeto System.out deve ser chamado com o argumento passado pelo lambda. Isso é particularmente útil quando você quer executar uma operação simples em cada elemento de uma coleção.

Benefícios Adicionais

- Legibilidade: O código fica mais fácil de ler e entender, especialmente para iniciantes ou pessoas não familiarizadas com a sintaxe específica do Java.

- Expressividade: Os lambdas permitem expressar conceitos complexos de maneira concisa, facilitando a compreensão do propósito do código.

- Flexibilidade: Eles podem ser usados em combinação com outras novidades do Java 8, como streams, para realizar operações complexas sobre coleções de maneira declarativa e funcional.

Em resumo, o uso de operadores lambda no Java simplificou significativamente a impressão de valores contidos em entidades, além de oferecer benefícios adicionais em termos de legibilidade, expressividade e flexibilidade. Essa característica é um exemplo claro de como o Java evoluiu para suportar programação funcional, tornando o código mais limpo e eficiente.

Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?

Os métodos em Java são associados a objetos da classe em que são definidos. Isso significa que, por padrão, eles operam dentro do contexto de uma instância específica dessa classe. No entanto, existem situações em que você deseja executar um método sem criar uma instância da classe. Nesses casos, os métodos devem ser declarados como static.

Razões para usar métodos estáticos:

- Inicialização de recursos compartilhados: Métodos estáticos podem ser usados para inicializar recursos que são compartilhados entre todas as instâncias de uma classe, como conexões de banco de dados ou configurações globais.

- Operações matemáticas e cálculos: Métodos que realizam operações matemáticas simples ou cálculos que não dependem do estado interno de uma instância da classe podem ser feitos estáticos.

- Métodos auxiliares: Métodos que ajudam a realizar tarefas relacionadas à manipulação de dados ou processamento, mas que não alteram o estado de uma instância específica, podem ser estáticos.

- Conveniência na criação de classes utilitárias: Classes que contêm apenas métodos estáticos são frequentemente referidas como "classes utilitárias". Elas são projetadas para fornecer funcionalidades úteis que não necessariamente estão vinculadas ao estado de uma instância específica.