

Campus: Polo Jacarepaguá, Rio de Janeiro/RJ

Curso: Desenvolvimento Full Stack

Disciplina: RPG0014 – Iniciando pelo Java

Semestre Letivo: 2023.1 FLEX

Integrante: Fabio Fernando dos Santos

Matrícula: 202302690807

Link do repositório GitHub: <https://github.com/FabioFernandoSantos/Mundo3-Nivel1.git>

Título da Prática:

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

Objetivo da Prática:

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários

1º Procedimento | Criação das Entidades e Sistema de Persistência

Todos os códigos Solicitados:

1 – Pessoa.java

```
package model.entidades;

import java.io.Serializable;

public class Pessoa implements Serializable {

    private int id;

    private String nome;

    public Pessoa() {

    }

    public Pessoa(int id, String nome) {

        this.id = id;

        this.nome = nome;

    }

    public void exibir() {

        System.out.println("ID: " + id + ", Nome: " + nome);

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getNome() {

        return nome;

    }

    public void setNome(String nome) {

        this.nome = nome;

    }

}
```

2 – PessoaFisica.java

```
package model.entidades;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {

    private String cpf;

    private int idade;

    public PessoaFisica() {

        super(); // Chama o construtor da classe pai (Pessoa)

    }

    public PessoaFisica(int id, String nome, String cpf, int idade) {

        super(id, nome); // Chama o construtor da classe pai com id e nome

        this.cpf = cpf;

        this.idade = idade;

    }

    @Override

    public void exibir() {

        // Primeiro chama o método exibir da classe pai

        super.exibir();

        // Em seguida, imprime as informações adicionais

        System.out.println("CPF: " + cpf + ", Idade: " + idade);

    }

    public String getCpf() {

        return cpf;

    }

    public void setCpf(String cpf) {

        this.cpf = cpf;

    }

    public int getIdade() {

        return idade;

    }

    public void setIdade(int idade) {

        this.idade = idade;

    }

}
```

3 – PessoaJuridica.java

```
package model.entidades;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {

    private String cnpj;

    public PessoaJuridica() {

        super(); // Invoca o construtor da classe Pai (Pessoa)

    }

    public PessoaJuridica(int id, String nome, String cnpj) {

        super(id, nome); // Invoca o construtor da classe pai com id e nome

        this.cnpj = cnpj;

    }

    @Override

    public void exhibir() {

        super.exibir();

        System.out.println("CNPJ: " + cnpj);

    }

    public String getCnpj() {

        return cnpj;

    }

    public void setCnpj(String cnpj) {

        this.cnpj = cnpj;

    }

}
```

4 – PessoaFisicaRepo.java

```
package model.gerenciadores;

import java.util.List;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;

import model.entidades.PessoaFisica;

public class PessoaFisicaRepo {

    private List<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {

        this.pessoasFisicas = new ArrayList<>();
    }

    public void inserir(PessoaFisica pessoaFisica) {

        pessoasFisicas.add(pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica) {

        for (int i = 0; i < pessoasFisicas.size(); i++) {

            if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {

                pessoasFisicas.set(i, pessoaFisica);

                return; }}}

    public void excluir(int id) {

        pessoasFisicas.removeIf(pessoaFisica -> pessoaFisica.getId() == id);
    }

    public PessoaFisica obter(int id) {

        for (PessoaFisica pf : pessoasFisicas) {

            if (pf.getId() == id) {

                return pf;}}

        return null; }

    public List<PessoaFisica> obterTodos() {

        return new ArrayList<>(pessoasFisicas); }

    public void persistir(String nomeArquivo) throws IOException {

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {

            oos.writeObject(pessoasFisicas); }

    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {

            pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();}}

    }
```

5 – PessoaJuridicaRepo.java

```
package model.gerenciadores;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.List;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import model.entidades.PessoaJuridica;

public class PessoaJuridicaRepo {

    private List<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {

        this.pessoasJuridicas = new ArrayList<>();
    }

    public void inserir(PessoaJuridica pessoaJuridica) {

        pessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {

        for (int i = 0; i < pessoasJuridicas.size(); i++) {

            if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {

                pessoasJuridicas.set(i, pessoaJuridica);

                return;
            }
        }
    }

    public void excluir(int id) {

        pessoasJuridicas.removeIf(pessoaJuridica -> pessoaJuridica.getId() == id);
    }

    public PessoaJuridica obter(int id) {

        for (PessoaJuridica pj : pessoasJuridicas) {

            if (pj.getId() == id) {

                return pj;
            }
        }

        return null;
    }

    public List<PessoaJuridica> obterTodos() {

        return new ArrayList<>(pessoasJuridicas);
    }

    public void persistir(String nomeArquivo) throws IOException {

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {

            oos.writeObject(pessoasJuridicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {

            pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
        }
    }
}
```

6 – Main.java

```
package principal;

import java.io.IOException;

import model.entidades.PessoaFisica;
import model.entidades.PessoaJuridica;
import model.gerenciadores.PessoaFisicaRepo;
import model.gerenciadores.PessoaJuridicaRepo;

public class Main {

    public static void main(String[] args) {

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        PessoaFisica pf1 = new PessoaFisica(1, "Julio Marques", "123.456.789-00", 36);

        PessoaFisica pf2 = new PessoaFisica(2, "Ingrid Albuquerque", "987.654.321-00", 35);

        repo1.inserir(pf1);

        repo1.inserir(pf2);

        String nomeArquivo = "pessoasFisicas.dat";

        try {

            repo1.persistir(nomeArquivo);

        } catch (IOException e) {

            System.err.println("Erro: " + e); }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

        try {

            repo2.recuperar(nomeArquivo);

            System.out.println("## Dados de Pessoas Fisicas Armazenados");

        } catch (IOException | ClassNotFoundException e) {

            System.err.println("Erro: " + e); }

        System.out.println("### Pessoas Cadastradas:");

        for (PessoaFisica pf : repo2.obterTodos()) {

            pf.exibir();}

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        PessoaJuridica pj1 = new PessoaJuridica(1, "XPTO Sales", "12.345.678/0001-99");

        PessoaJuridica pj2 = new PessoaJuridica(2, "XPTO Solutions", "98.765.432/0001-11");

        repo3.inserir(pj1);

        repo3.inserir(pj2);
```

```

String nomeArquivoPJ = "pessoasJuridicas.dat";

try {

    repo3.persistir(nomeArquivoPJ);

    System.out.println("\n## Dados de Pessoas Juridicas Armazenados");

} catch (IOException e) {

    System.err.println("Erro Encontrado: " + e); }

PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

String nomeArquivoPJ2 = "pessoasJuridicas.dat";

try {

    repo4.recuperar(nomeArquivoPJ2);

} catch (IOException | ClassNotFoundException e) {

    System.err.println("Erro Encontrado: " + e);

}

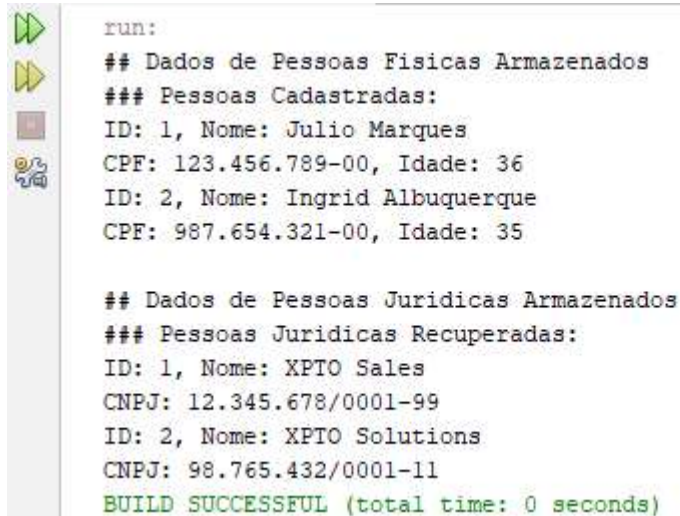
System.out.println("### Pessoas Juridicas Recuperadas:");

for (PessoaJuridica pj : repo4.obterTodos()) {

    pj.exibir(); } }

```

Resultados da execução dos códigos:



```

run:
## Dados de Pessoas Fisicas Armazenados
### Pessoas Cadastradas:
ID: 1, Nome: Julio Marques
CPF: 123.456.789-00, Idade: 36
ID: 2, Nome: Ingrid Albuquerque
CPF: 987.654.321-00, Idade: 35

## Dados de Pessoas Juridicas Armazenados
### Pessoas Juridicas Recuperadas:
ID: 1, Nome: XPTO Sales
CNPJ: 12.345.678/0001-99
ID: 2, Nome: XPTO Solutions
CNPJ: 98.765.432/0001-11
BUILD SUCCESSFUL (total time: 0 seconds)

```


Análise e Conclusão:

- 1) Quais as vantagens e desvantagens do uso de herança?
Com heranças podemos reutilizar os códigos bem como os métodos e suas variáveis, como realizado por exemplo com as Classes Pessoa, PessoaFisica e PessoaJuridica. Outra vantagem está na manutenção do código, uma simples mudança no código de uma classe pai já ocorre automaticamente mudanças em suas classes filhas. Outras pessoas que necessitem ajustar o código, uma facilidade no entendimento.
- 2) Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?
Ela sinaliza para JVM que objetos de uma classe podem ser transformados em uma sequência de bytes **serializados** e reconstruídos de volta em objetos **desserializados**. Portanto torna possível e fácil salvar o estado completo de um objeto em um arquivo binário e recuperá-lo posteriormente, mantendo todas as suas informações intactas.
- 3) Como o paradigma funcional é utilizado pela API Stream no Java?
API Stream do Java incorpora o paradigma funcional permitindo operações em coleções de dados de forma declarativa, imutável e sem efeitos colaterais. Utilizamos expressões lambda ou Arrow Function para operações como filtragem, mapeamento e redução.
- 4) Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?
O padrão é a serialização/deserialização, onde a interface Serializable é utilizada para marcar classes onde os objetos podem ser convertidos em uma sequência de bytes e posteriormente reconstruídos. Este método permite a gravação e leitura desses arquivos, facilitando a sua armazenagem em arquivos ou transmissão por redes.

2º Procedimento | Criação do Cadastro em Modo Texto

Todos os códigos Solicitados:

```
package principal;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import static java.lang.Integer.parseInt;
import java.util.List;
import java.util.Scanner;

import model.entidades.PessoaFisica;
import model.entidades.PessoaJuridica;
import model.gerenciadores.PessoaFisicaRepo;
import model.gerenciadores.PessoaJuridicaRepo;

public class MainMenu {

    private static Scanner scanner = new Scanner(System.in);
    private static PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    private static PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

    public static void main(String[] args) {
        int opcao;
        do {
            exibirMenu();
            opcao = scanner.nextInt();
            scanner.nextLine(); // Limpar buffer do scanner
            processarOpcao(opcao);
        } while (opcao != 0);
    }

    private static void exibirMenu() {
        System.out.println("Escolha uma opcao:");
        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("6 - Persistir Dados");
        System.out.println("7 - Recuperar Dados");
        System.out.println("0 - Finalizar Programa");
        System.out.println("=====");
        System.out.print("Digite uma opcao: ");
    }

    private static void processarOpcao(int opcao) {
        switch (opcao) {
            case 1 ->
                incluir();
            case 2 ->
                alterar();
        }
    }
}
```

```

        case 3 ->
            excluir();
        case 4 ->
            exibirPorId();
        case 5 ->
            exibirTodos();
        case 6 ->
            salvarDados();
        case 7 ->
            recuperarDados();
        case 0 ->
            System.out.println("Finalizando...");
        default ->
            System.out.println("\n ## Opção invalida! ##");
    }
}

```

```

private static void incluir() {
    System.out.println("Incluir Pessoa (1 - Fisica, 2 - Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisica pf = new PessoaFisica();

        System.out.println("Digite um ID: ");
        pf.setId(parseInt(scanner.nextLine()));

        System.out.println("Digite o nome: ");
        pf.setNome(scanner.nextLine());

        System.out.println("Digite o CPF: ");
        pf.setCpf(scanner.nextLine());

        System.out.println("Digite a idade: ");
        int idade = scanner.nextInt();
        scanner.nextLine();

        pf.setIdade(idade);

        repoFisica.inserir(pf);
        System.out.println("\n## Pessoa Fisica adicionada com sucesso! ##");
    } else if (tipo == 2) {
        PessoaJuridica pj = new PessoaJuridica();

        System.out.println("Digite um ID: ");
        pj.setId(parseInt(scanner.nextLine()));

        System.out.println("Digite o nome:");
        pj.setNome(scanner.nextLine());

        System.out.println("Digite o CNPJ:");
        pj.setCnpj(scanner.nextLine());

        repoJuridica.inserir(pj);
        System.out.println("\n## Pessoa Juridica adicionada com sucesso! ##");
    } else {
        System.out.println("\n## Tipo invalido. ##");
    }
}

```

```
}  
}
```

```
private static void alterar() {  
    System.out.println("Alterar Pessoa (1 - Fisica, 2 - Juridica): ");  
    int tipo = scanner.nextInt();  
    scanner.nextLine();  
  
    System.out.println("Digite o ID da pessoa:");  
    int id = scanner.nextInt();  
    scanner.nextLine();  
  
    if (tipo == 1) {  
        PessoaFisica pf = repoFisica.obter(id);  
        if (pf != null) {  
            System.out.println("Dados atuais: ");  
            pf.exibir();  
  
            System.out.println("Digite o novo nome (deixe em branco para nao alterar:");  
            String nome = scanner.nextLine();  
            if (!nome.isEmpty()) {  
                pf.setNome(nome);  
            }  
  
            System.out.println("Digite o novo CPF (deixe em branco para nao alterar:");  
            String cpf = scanner.nextLine();  
            if (!cpf.isEmpty()) {  
                pf.setCpf(cpf);  
            }  
  
            System.out.println("Digite a nova idade (insira 0 para nao alterar:");  
            int idade = scanner.nextInt();  
            scanner.nextLine();  
            if (idade != 0) {  
                pf.setIdade(idade);  
            }  
  
            repoFisica.alterar(pf);  
            System.out.println("\n## Pessoa Fisica atualizada com sucesso!");  
        } else {  
            System.out.println("\n## Pessoa Fisica não encontrada.");  
        }  
    } else if (tipo == 2) {  
        PessoaJuridica pj = repoJuridica.obter(id);  
        if (pj != null) {  
            System.out.println("Dados atuais: ");  
            pj.exibir();  
  
            System.out.println("Digite o novo nome (deixe em branco para nao alterar:");  
            String nome = scanner.nextLine();  
            if (!nome.isEmpty()) {  
                pj.setNome(nome);  
            }  
  
            System.out.println("Digite o novo CNPJ (deixe em branco para nao alterar:");  
            String cnpj = scanner.nextLine();  
            if (!cnpj.isEmpty()) {  
                pj.setCnpj(cnpj);  
            }  
        }  
    }  
}
```

```

    }

    repoJuridica.alterar(pj);
    System.out.println("\n## Pessoa Juridica atualizada com sucesso! ##");
} else {
    System.out.println("\n## Pessoa Juridica não encontrada. ##");
}
} else {
    System.out.println("\n## Tipo invalido. ##");
}
}

```

```

private static void excluir() {
    System.out.println("Excluir Pessoa (1 - Fisica, 2 - Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    System.out.println("Digite o ID da pessoa a ser excluida:");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) {
            repoFisica.excluir(id);
            System.out.println("\n## Pessoa Fisica removida com sucesso! ##");
        } else {
            System.out.println("\n## Pessoa Fisica não encontrada. ##");
        }
    } else if (tipo == 2) {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            repoJuridica.excluir(id);
            System.out.println("\n## Pessoa Juridica removida com sucesso! ##");
        } else {
            System.out.println("\n## Pessoa Juridica nao encontrada. ##");
        }
    } else {
        System.out.println("\n## Tipo invalido. ##");
    }
}
}

```

```

private static void exibirPorId() {
    System.out.println("Exibir dados de Pessoa (1 - Fisica, 2 - Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine(); // Limpar buffer do scanner

    System.out.println("Digite o ID da pessoa:");
    int id = scanner.nextInt();
    scanner.nextLine(); // Limpar buffer do scanner

    if (tipo == 1) {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) {
            System.out.println("Dados da Pessoa Fisica:");
            pf.exibir();
        } else {
            System.out.println("\n## Pessoa Fisica nao encontrada. ##");
        }
    }
}

```

```

    }
} else if (tipo == 2) {
    PessoaJuridica pj = repoJuridica.obter(id);
    if (pj != null) {
        System.out.println("Dados da Pessoa Juridica:");
        pj.exibir();
    } else {
        System.out.println("\n## Pessoa Jurídica nao encontrada. ##");
    }
} else {
    System.out.println("\n## Tipo invalido. ##");
}
}

private static void exibirTodos() {
    System.out.println("Exibir todos (1 - Fisica, 2 - Juridica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        System.out.println("Lista de Todas as Pessoas Fisicas:");
        for (PessoaFisica pf : repoFisica.obterTodos()) {
            pf.exibir();
            System.out.println("-----");
        }
    } else if (tipo == 2) {
        System.out.println("Lista de Todas as Pessoas Juridicas:");
        for (PessoaJuridica pj : repoJuridica.obterTodos()) {
            pj.exibir();
            System.out.println("-----");
        }
    } else {
        System.out.println("\n## Tipo invalido. ##");
    }
}

private static void salvarDados() {
    System.out.println("Digite o prefixo para salvar os arquivos:");
    String prefixo = scanner.nextLine();

    // Salvando dados de Pessoa Física
    try (ObjectOutputStream oosFisica = new ObjectOutputStream(new FileOutputStream(prefixo + ".fisica.bin"))) {
        oosFisica.writeObject(repoFisica.obterTodos());
        System.out.println("\n## Dados de Pessoas Fisicas salvos com sucesso.");
    } catch (IOException e) {
        System.err.println("\n## Erro ao salvar dados de Pessoas Fisicas: " + e.getMessage());
    }

    // Salvando dados de Pessoa Jurídica
    try (ObjectOutputStream oosJuridica = new ObjectOutputStream(new FileOutputStream(prefixo + ".juridica.bin"))) {
        oosJuridica.writeObject(repoJuridica.obterTodos());
        System.out.println("\n## Dados de Pessoas Juridicas salvos com sucesso. ##");
    } catch (IOException e) {
        System.err.println("\n ##Erro ao salvar dados de Pessoas Juridicas: " + e.getMessage());
    }
}

private static void recuperarDados() {

```

```

System.out.println("Digite o prefixo dos arquivos para recuperacao:");
String prefixo = scanner.nextLine();

// Recuperando dados de Pessoa Física
try (ObjectInputStream oisFisica = new ObjectInputStream(new FileInputStream(prefixo + ".fisica.bin"))) {
    List<PessoaFisica> listaFisica = (List<PessoaFisica>) oisFisica.readObject();
    repoFisica.setLista(listaFisica);
    System.out.println("\n##Dados de Pessoas Fisicas recuperados com sucesso. ##");
} catch (IOException | ClassNotFoundException e) {
    System.err.println("\n## Erro ao recuperar dados de Pessoas Fisicas: " + e.getMessage());
}

// Recuperando dados de Pessoa Jurídica
try (ObjectInputStream oisJuridica = new ObjectInputStream(new FileInputStream(prefixo + ".juridica.bin"))) {
    List<PessoaJuridica> listaJuridica = (List<PessoaJuridica>) oisJuridica.readObject();
    repoJuridica.setLista(listaJuridica);
    System.out.println("\n ## Dados de Pessoas Juridicas recuperados com sucesso. ##");
} catch (IOException | ClassNotFoundException e) {
    System.err.println("\n## Erro ao recuperar dados de Pessoas Juridicas: " + e.getMessage());
}
}
}

```

Resultados da Execução:

```

run:
Escolha uma opcao:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite uma opcao: 1
Incluir Pessoa (1 - Fisica, 2 - Juridica):
1
Digite um ID:
1
Digite o nome:
Julio Marques
Digite o CPF:
001231231
Digite a idade:
36

## Pessoa Fisica adicionada com sucesso! ##

```

Análise e Conclusão:

1. O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?
O método main é o ponto de entrada de qualquer programa Java. Quando um programa Java é iniciado, a JVM busca pela classe especificada e invoca o método main dessa classe. Se o main não fosse estático, a JVM precisaria instanciar a classe primeiro, o que não é prático nem desejável para um ponto de entrada do programa.
2. Para que serve a classe Scanner?
É para ler dados de diferentes fontes, incluindo entrada de usuário no console, strings, arquivos, entre outros. É usada para ler e interpretar tipos primitivos de dados de forma fácil e eficiente.
3. Como o uso de classes de repositório impactou na organização do código?
É arquiteturas MVC é comum utilizarmos camadas para isolarmos as responsabilidades, portanto é uma boa prática trabalharmos com classes de repositórios onde conseguimos por exemplo, separar alguma lógica das classes mais abstratas. Facilitando assim a manutenção do código.