



**Campus de Santa Cruz da Serra
Desenvolvimento Full Stack
Iniciando o Caminho Pelo Java**



2023.1

3º Semestre

Carlos Daniel Pereira dos Santos

Iniciando o caminho pelo Java - *Criação das Entidades e Sistema de Persistência*

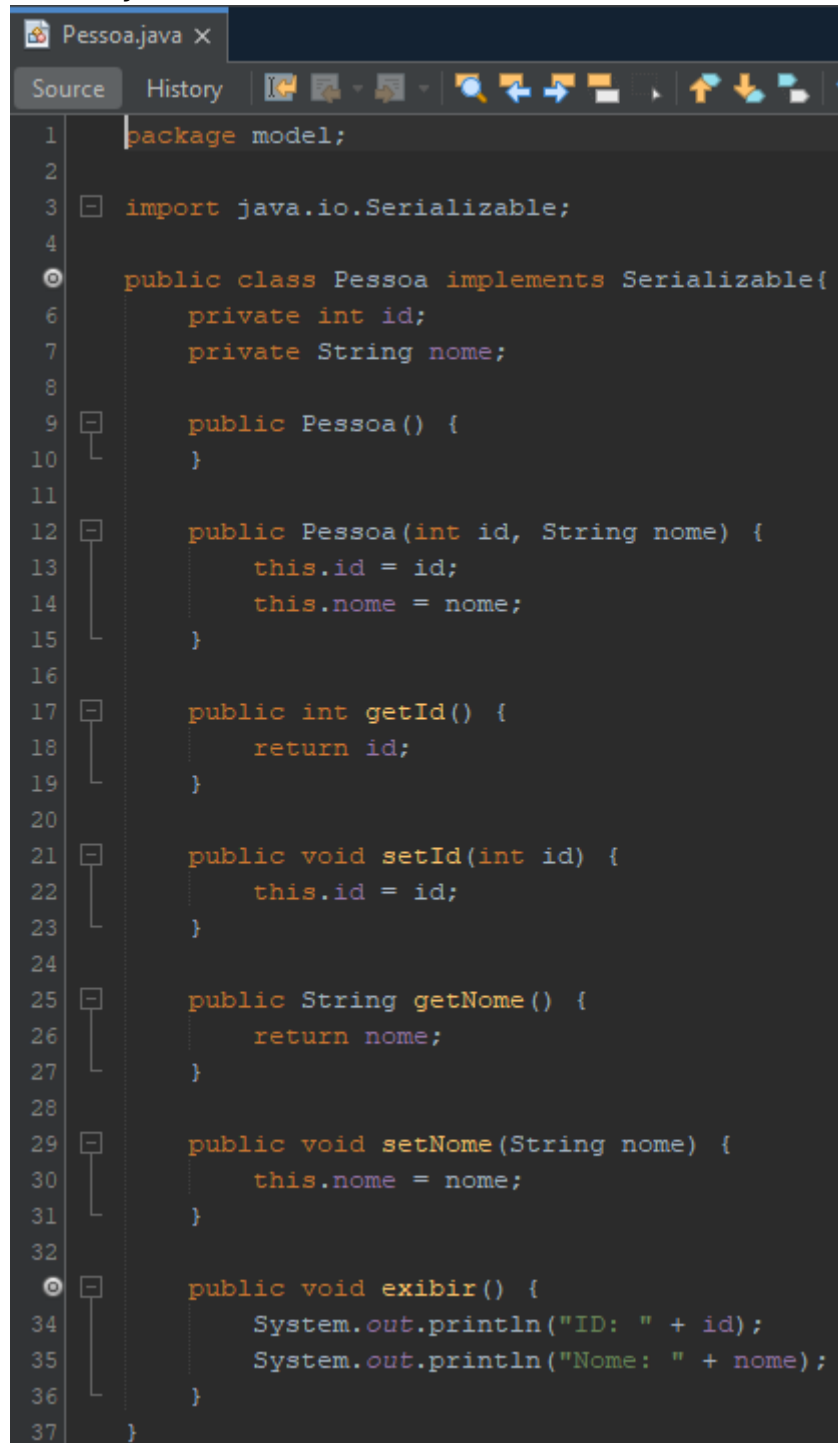
Objetivo:

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

Link GitHub: <https://github.com/dan-perr/Iniciando-o-caminho-pelo-Java.git>

Todos os códigos solicitados:

Pessoa.java:



```
1 package model;
2
3 import java.io.Serializable;
4
5 public class Pessoa implements Serializable{
6     private int id;
7     private String nome;
8
9     public Pessoa() {
10    }
11
12     public Pessoa(int id, String nome) {
13         this.id = id;
14         this.nome = nome;
15     }
16
17     public int getId() {
18         return id;
19     }
20
21     public void setId(int id) {
22         this.id = id;
23     }
24
25     public String getNome() {
26         return nome;
27     }
28
29     public void setNome(String nome) {
30         this.nome = nome;
31     }
32
33     public void exibir() {
34         System.out.println("ID: " + id);
35         System.out.println("Nome: " + nome);
36     }
37 }
```

Por extenso (Pessoa.java):

```
package model;
```

```
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 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;  
    }  
}
```

```
    public void exibir() {  
        System.out.println("ID: " + id);  
        System.out.println("Nome: " + nome);  
    }  
}
```

PessoaFisica.java



```
1 package model;
2
3 import java.io.Serializable;
4
5 public class PessoaFisica extends Pessoa {
6     private String cpf;
7     private int idade;
8
9     public PessoaFisica() {
10    }
11
12     public PessoaFisica(int id, String nome, String cpf, int idade) {
13         super(id, nome);
14         this.cpf = cpf;
15         this.idade = idade;
16     }
17
18     @Override
19     public void exibir() {
20         super.exibir();
21         System.out.println("CPF: " + cpf);
22         System.out.println("Idade: " + idade);
23     }
24
25     public String getCpf() {
26         return cpf;
27     }
28
29     public void setCpf(String cpf) {
30         this.cpf = cpf;
31     }
32
33     public int getIdade() {
34         return idade;
35     }
36
37     public void setIdade(int idade) {
38         this.idade = idade;
39     }
40 }
```

Por extenso (PessoaFisica.java):

```
package model;
```

```
import java.io.Serializable;
```

```
public class PessoaFisica extends Pessoa {  
    private String cpf;  
    private int idade;
```

```
    public PessoaFisica() {  
    }
```

```
    public PessoaFisica(int id, String nome, String cpf, int idade) {  
        super(id, nome);  
        this.cpf = cpf;  
        this.idade = idade;  
    }
```

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

```
}  
}
```

PessoaJuridica.java

```
1  package model;  
2  
3  import java.io.Serializable;  
4  
5  public class PessoaJuridica extends Pessoa {  
6      private String cnpj;  
7  
8      public PessoaJuridica() {  
9      }  
10  
11     public PessoaJuridica(int id, String nome, String cnpj) {  
12         super(id, nome);  
13         this.cnpj = cnpj;  
14     }  
15  
16     @Override  
17     public void exibir() {  
18         super.exibir();  
19         System.out.println("CNPJ: " + cnpj);  
20     }  
21  
22     public String getCnpj() {  
23         return cnpj;  
24     }  
25  
26     public void setCnpj(String cnpj) {  
27         this.cnpj = cnpj;  
28     }  
29 }
```

Por Extenso (PessoaJuridica.java):

```
package model;
```

```
import java.io.Serializable;
```

```
public class PessoaJuridica extends Pessoa {  
    private String cnpj;
```

```
    public PessoaJuridica() {  
    }
```

```
    public PessoaJuridica(int id, String nome, String cnpj) {  
        super(id, nome);  
        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;  
    }  
}
```

PessoaFisicaRepo.java

```
1  package model;
2
3  import java.util.List;
4  import java.util.ArrayList;
5  import java.io.FileInputStream;
6  import java.io.FileOutputStream;
7  import java.io.IOException;
8  import java.io.ObjectOutputStream;
9  import java.io.ObjectInputStream;
10
11 public class PessoaFisicaRepo{
12     private List<PessoaFisica> pessoasFisicas;
13
14     public PessoaFisicaRepo() {
15         pessoasFisicas = new ArrayList<>();
16     }
17
18     public void inserir(PessoaFisica pessoaFisica) {
19         pessoasFisicas.add(e: pessoaFisica);
20     }
21
22     public void alterar(int indice, PessoaFisica pessoaFisica) {
23         if (indice >= 0 && indice < pessoasFisicas.size()) {
24             pessoasFisicas.set(index: indice, element:pessoaFisica);
25         } else {
26             throw new IndexOutOfBoundsException(s: "Índice inválido");
27         }
28     }
29
30     public void excluir(int indice) {
31         if (indice >= 0 && indice < pessoasFisicas.size()) {
32             pessoasFisicas.remove(index: indice);
33         } else {
34             throw new IndexOutOfBoundsException(s: "Índice inválido");
35         }
36     }
37
38     public PessoaFisica obter(int indice) {
39         if (indice >= 0 && indice < pessoasFisicas.size()) {
40             return pessoasFisicas.get(index: indice);
41         } else {
42             throw new IndexOutOfBoundsException(s: "Índice inválido");
43         }
44     }
45
46     public List<PessoaFisica> obterTodos() {
47         return new ArrayList<>(: pessoasFisicas);
48     }
```



```

49
50 public void persistir(String nomeArquivo) throws IOException {
51     try (FileOutputStream fos = new FileOutputStream(name: nomeArquivo);
52          ObjectOutputStream oos = new ObjectOutputStream(out: fos)) {
53         oos.writeObject(obj: pessoasFisicas);
54     }
55 }
56
57 public void recuperar(String nomeArquivo) {
58     try (FileInputStream fis = new FileInputStream(name: nomeArquivo);
59          ObjectInputStream ois = new ObjectInputStream(in: fis)) {
60         pessoasFisicas = (List<PessoaFisica>) ois.readObject();
61     } catch (IOException | ClassNotFoundException e) {
62         e.printStackTrace();
63     }
64 }
65

```

Por Extenso (PessoaFisicaRepo.java):

package model;

import java.util.List;

import java.util.ArrayList;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

public class PessoaFisicaRepo{

private List<PessoaFisica> pessoasFisicas;

public PessoaFisicaRepo() {

pessoasFisicas = new ArrayList<>();

}

public void inserir(PessoaFisica pessoaFisica) {

pessoasFisicas.add(pessoaFisica);

}

public void alterar(int indice, PessoaFisica pessoaFisica) {

if (indice >= 0 && indice < pessoasFisicas.size()) {

pessoasFisicas.set(indice, pessoaFisica);

} else {

throw new IndexOutOfBoundsException("Índice inválido");

}

}

public void excluir(int indice) {

if (indice >= 0 && indice < pessoasFisicas.size()) {

pessoasFisicas.remove(indice);

} else {

```

        throw new IndexOutOfBoundsException("Índice inválido");
    }
}

public PessoaFisica obter(int indice) {
    if (indice >= 0 && indice < pessoasFisicas.size()) {
        return pessoasFisicas.get(indice);
    } else {
        throw new IndexOutOfBoundsException("Índice inválido");
    }
}

public List<PessoaFisica> obterTodos() {
    return new ArrayList<>(pessoasFisicas);
}

public void persistir(String nomeArquivo) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos)) {
        oos.writeObject(pessoasFisicas);
    }
}

public void recuperar(String nomeArquivo) {
    try (FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis)) {
        pessoasFisicas = (List<PessoaFisica>) ois.readObject();
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
}

```

PessoaJuridicaRepo.java

```
1 package model;
2
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class PessoaJuridicaRepo {
12     private List<PessoaJuridica> pessoasJuridicas;
13
14     public PessoaJuridicaRepo() {
15         pessoasJuridicas = new ArrayList<>();
16     }
17
18     public void inserir(PessoaJuridica pessoaJuridica) {
19         pessoasJuridicas.add(pessoaJuridica);
20     }
21
22     public void alterar(PessoaJuridica pessoaJuridica) {
23         for (int i = 0; i < pessoasJuridicas.size(); i++) {
24             if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {
25                 pessoasJuridicas.set(i, pessoaJuridica);
26                 return;
27             }
28         }
29         throw new IllegalArgumentException("Pessoa com o ID especificado não encontrada para alteração");
30     }
31
32     public void excluir(int id) {
33         for (int i = 0; i < pessoasJuridicas.size(); i++) {
34             if (pessoasJuridicas.get(i).getId() == id) {
35                 pessoasJuridicas.remove(i);
36                 return;
37             }
38         }
39         throw new IllegalArgumentException("Pessoa com o ID especificado não encontrada para exclusão");
40     }
41
42     public PessoaJuridica obter(int id) {
43         for (PessoaJuridica pessoa : pessoasJuridicas) {
44             if (pessoa.getId() == id) {
45                 return pessoa;
46             }
47         }
48         throw new IllegalArgumentException("Pessoa com o ID especificado não encontrada");
49     }
50
51     public List<PessoaJuridica> obterTodos() {
52         return new ArrayList<>(pessoasJuridicas);
53     }
54
55     public void persistir(String nomeArquivo) throws IOException {
56         try (FileOutputStream fos = new FileOutputStream(nomeArquivo);
57             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
58             oos.writeObject(pessoasJuridicas);
59         }
60     }
61
62     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
63         try (FileInputStream fis = new FileInputStream(nomeArquivo);
64             ObjectInputStream ois = new ObjectInputStream(fis)) {
65             pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
66         }
67     }
68 }
```

Por Extenso (PessoaJuridicaRepo.java):

```
package model;
```

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

```
public class PessoaJuridicaRepo {  
    private List<PessoaJuridica> pessoasJuridicas;
```

```
    public PessoaJuridicaRepo() {  
        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;  
            }  
        }  
        throw new IllegalArgumentException("Pessoa com o ID especificado não  
encontrada para alteração");  
    }
```

```
    public void excluir(int id) {  
        for (int i = 0; i < pessoasJuridicas.size(); i++) {  
            if (pessoasJuridicas.get(i).getId() == id) {  
                pessoasJuridicas.remove(i);  
                return;  
            }  
        }  
        throw new IllegalArgumentException("Pessoa com o ID especificado não  
encontrada para exclusão");  
    }
```

```
public PessoaJuridica obter(int id) {
    for (PessoaJuridica pessoa : pessoasJuridicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    throw new IllegalArgumentException("Pessoa com o ID especificado não encontrada");
}

public List<PessoaJuridica> obterTodos() {
    return new ArrayList<>(pessoasJuridicas);
}

public void persistir(String nomeArquivo) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos)) {
        oos.writeObject(pessoasJuridicas);
    }
}

public void recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
    try (FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis)) {
        pessoasJuridicas = (List<PessoaJuridica>) ois.readObject();
    }
}
}
```

ProjetoPOO.java (Main):

```
Source History
1 package projetopoo;
2
3 import java.util.List;
4 import java.io.IOException;
5 import model.PessoaFisicaRepo;
6 import model.PessoaFisica;
7 import model.PessoaJuridicaRepo;
8 import model.PessoaJuridica;
9
10 public class ProjetoPOO {
11
12     public static void main(String[] args) {
13         PessoaFisicaRepo repol = new PessoaFisicaRepo();
14         PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();
15
16         repol.inserir(new PessoaFisica(id: 1, nome: "Ana", cpf: "11111111111", idade: 25));
17         repol.inserir(new PessoaFisica(id: 2, nome: "Carlos", cpf: "22222222222", idade: 52));
18
19         try {
20             repol.persistir(nomeArquivo: "pessoasFisicas.dat");
21             System.out.println("Dados de Pessoa Fisica Armazenados.");
22         } catch (IOException e) {
23             System.out.println("Erro ao persistir os dados das pessoas fisicas: " + e.getMessage());
24         }
25
26         try {
27             repo2.recuperar(nomeArquivo: "pessoasFisicas.dat");
28             System.out.println("Dados de Pessoa Fisica Recuperados.");
29         } catch (IOException | ClassNotFoundException e) {
30             System.out.println("Erro ao recuperar os dados das pessoas fisicas: " + e.getMessage());
31         }
32
33         List<PessoaFisica> pessoasFisicasRecuperadas = repol.obterTodos();
34         for (PessoaFisica pessoa: pessoasFisicasRecuperadas) {
35             System.out.println("Id: " + pessoa.getId());
36             System.out.println("Nome: " + pessoa.getNome());
37             System.out.println("CPF: " + pessoa.getCpf());
38             System.out.println("Idade: " + pessoa.getIdade());
39         }
40
41         PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
42         PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
43
44         repo3.inserir(new PessoaJuridica(id: 3, nome: "XPTO Sales", cnpj: "3333333333333333"));
45         repo3.inserir(new PessoaJuridica(id: 4, nome: "XPTO Solutions", cnpj: "4444444444444444"));
46
47         try {
48             repo3.persistir(nomeArquivo: "pessoasJuridicas.dat");
49             System.out.println("Dados de Pessoa Juridica Armazenados.");
50         } catch (IOException e) {
51             System.out.println("Erro ao recuperar os dados das pessoas juridicas: " + e.getMessage());
52         }
53
54         try {
55             repo4.recuperar(nomeArquivo: "pessoasJuridicas.dat");
56             System.out.println("Dados de Pessoa Juridica Recuperados.");
57         } catch (IOException | ClassNotFoundException e) {
58             System.out.println("Erro ao recuperar os dados das pessoas juridicas: " + e.getMessage());
59         }
60
61         List<PessoaJuridica> pessoasJuridicasRecuperadas = repo4.obterTodos();
62         for (PessoaJuridica pessoa : pessoasJuridicasRecuperadas) {
63             System.out.println("ID: " + pessoa.getId());
64             System.out.println("Nome: " + pessoa.getNome());
65             System.out.println("CNPJ: " + pessoa.getCnpj());
66         }
67     }
68 }
69
```

```
package projetopoo;
```

```
import java.util.List;
import java.io.IOException;
import model.PessoaFisicaRepo;
import model.PessoaFisica;
import model.PessoaJuridicaRepo;
import model.PessoaJuridica;

public class ProjetoPOO {

    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaJuridicaRepo repo2 = new PessoaJuridicaRepo();

        repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
        repo1.inserir(new PessoaFisica(2, "Carlos", "2222222222", 52));

        try {
            repo1.persistir("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        } catch (IOException e) {
            System.out.println("Erro ao persistir os dados das pessoas fisicas: " +
e.getMessage());
        }

        try {
            repo2.recuperar("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados das pessoas fisicas: " +
e.getMessage());
        }

        List<PessoaFisica> pessoasFisicasRecuperadas = repo1.obterTodos();
        for (PessoaFisica pessoa: pessoasFisicasRecuperadas) {
            System.out.println("Id: " + pessoa.getId());
            System.out.println("Nome: " + pessoa.getNome());
            System.out.println("CPF: " + pessoa.getCpf());
            System.out.println("Idade: " + pessoa.getIdade());
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "3333333333333333"));
```

```
repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "4444444444444444"));

try {
    repo3.persistir("pessoasJuridicas.dat");
    System.out.println("Dados de Pessoa Juridica Armazenados.");
} catch (IOException e) {
    System.out.println("Erro ao recuperar os dados das pessoas juridicas: " +
e.getMessage());
}

try {
    repo4.recuperar("pessoasJuridicas.dat");
    System.out.println("Dados de Pessoa Juridica Recuperados.");
} catch (IOException | ClassNotFoundException e) {
    System.out.println("Erro ao recuperar os dados das pessoas juridicas: " +
e.getMessage());
}

List<PessoaJuridica> pessoasJuridicasRecuperadas = repo4.obterTodos();
for (PessoaJuridica pessoa : pessoasJuridicasRecuperadas) {
    System.out.println("ID: " + pessoa.getId());
    System.out.println("Nome: " + pessoa.getNome());
    System.out.println("CNPJ: " + pessoa.getCnpj());
}
}

}
```

Resultado dos códigos:

```
Output - ProjetoPOO (run)
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
Id: 1
Nome: Ana
CPF: 111111111111
Idade: 25
Id: 2
Nome: Carlos
CPF: 222222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3
Nome: XPTO Sales
CNPJ: 33333333333333
ID: 4
Nome: XPTO Solutions
CNPJ: 4444444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
```

Por extenso:

Dados de Pessoa Fisica Armazenados.

Dados de Pessoa Fisica Recuperados.

Id: 1

Nome: Ana

CPF: 111111111111

Idade: 25

Id: 2

Nome: Carlos

CPF: 222222222222

Idade: 52

Dados de Pessoa Juridica Armazenados.

Dados de Pessoa Juridica Recuperados.

ID: 3

Nome: XPTO Sales

CNPJ: 33333333333333

ID: 4

Nome: XPTO Solutions

CNPJ: 4444444444444444

BUILD SUCCESSFUL (total time: 0 seconds)

Análise e conclusão:

a. Quais as vantagens e desvantagens do uso de herança?

Resposta:

Vantagens: Promove reutilização de código, organização, extensibilidade e polimorfismo.

Desvantagens: Pode causar acoplamento alto, problemas de design, ruptura de encapsulamento e complexidade.

Mostrar sugestões para saber mais

b. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?

Resposta:

A interface Serializable é necessária para persistência em arquivos binários porque ela permite que os objetos de uma classe sejam convertidos em um formato que pode ser gravado e lido por um fluxo de bytes, facilitando o armazenamento e a recuperação dos estados dos objetos.

c. Como o paradigma funcional é utilizado pela API stream no Java?

Resposta:

Expressões Lambda: São funções anônimas que podem ser usadas como argumento para métodos. Elas são usadas extensivamente na API Stream para realizar operações em elementos de dados.

Métodos de Ordem Superior: São métodos que aceitam funções como parâmetros ou retornam uma função. Métodos como map, filter e reduce na API Stream são exemplos de métodos de ordem superior.

d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Resposta:

No Java, o padrão **Data Access Object (DAO)** é comumente usado para persistência de dados em arquivos. Ele abstrai e encapsula todas as operações de acesso aos dados, separando a lógica de negócios da lógica de persistência de dados.