**Novo projeto:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Modelo de entidades:

* Fornecedor TEM Muitos Produtos
* Produto PERTENCE a 1 Fornecedor

Interface gráfica do usuário, Diagrama, Excel

Descrição gerada automaticamente

**JavaBeans**

Padrão utilizado em Java para criação de classes que tem como objetivo modelagem de dados, tais como as entidades. São características de uma classe JavaBean:

* Atributos privados
* Métodos de encapsulamento para cada atributo, denominados de setters e getters
* Construtor sem entrada de argumentos
* Construtor com entrada de argumentos
* Sobrescrita dos métodos da classe Object

**package** entities;

**public** **class** Produto {

**private** Integer idProduto;

**private** String nome;

**private** Double preco;

**private** Integer quantidade;

**private** Fornecedor fornecedor;

**public** Integer getIdProduto() {

**return** idProduto;

}

**public** **void** setIdProduto(Integer idProduto) {

**this**.idProduto = idProduto;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** Double getPreco() {

**return** preco;

}

**public** **void** setPreco(Double preco) {

**this**.preco = preco;

}

**public** Integer getQuantidade() {

**return** quantidade;

}

**public** **void** setQuantidade(Integer quantidade) {

**this**.quantidade = quantidade;

}

**public** Fornecedor getFornecedor() {

**return** fornecedor;

}

**public** **void** setFornecedor(Fornecedor fornecedor) {

**this**.fornecedor = fornecedor;

}

}

**Collections**

Biblioteca Java composta da maioria dos componentes utilizados para manipulação de coleções de dados em Java. Podemos trabalhar em Java com Arrays (Vetores), porem, o uso das collections torna o trabalho de manipulação de coleções de dados muito mais fácil.

A biblioteca Collections do Java é composta de 3 principais componentes:

* **List**

Tipo de coleção utilizado para manipulação de listas simples de objetos. É o tipo mais utilizado em Java para coleções de dados.

Pode ser subdivido em: Filas e Pilhas.

* **Set**

Tipo de coleção baseado em uma regra: Não aceita objetos duplicados.

* **Map**

Tipo de coleção que armazena objetos utilizando o padrão de endereçamento CHAVE / VALOR.

Ex:

**CHAVE VALOR**

P01 Pessoa Id: 1, Nome: Sergio Mendes

P02 Pessoa Id: 2, Nome: Ana Paula



**package** entities;

**import** java.util.List;

**public** **class** Fornecedor {

**private** Integer idFornecedor;

**private** String nome;

**private** String cnpj;

**private** List<Produto> produtos;

**public** Integer getIdFornecedor() {

**return** idFornecedor;

}

**public** **void** setIdFornecedor(Integer idFornecedor) {

**this**.idFornecedor = idFornecedor;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** String getCnpj() {

**return** cnpj;

}

**public** **void** setCnpj(String cnpj) {

**this**.cnpj = cnpj;

}

**public** List<Produto> getProdutos() {

**return** produtos;

}

**public** **void** setProdutos(List<Produto> produtos) {

**this**.produtos = produtos;

}

}

Criando a classe principal:

/principal/Program.java

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**package** principal;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

}

}

**Objeto (**variável de instância**)**

Consiste em uma variável criada a partir do espaço de memória de uma classe. Esta variável é inicializada recebendo a instancia (referência) de uma classe.

Exemplo:

**Fornecedor fornecedor = new Fornecedor();**

[Classe] [Objeto] [Inicializando / Instanciando]

**Método construtor**

Consiste de um método padrão que toda classe já possui e que o compilado já assume que é implícito (não precisa estar escrito no código fonte da classe).

Este método é executado sempre depois da palavra reservada **new** e é utilizado para instanciarmos (inicializarmos) os objetos de uma classe.

Exemplo:

**Fornecedor fornecedor = new Fornecedor();**

[Classe] [Objeto] [Inicializando / Instanciando]

Escrevendo o método construtor:

**package** entities;

**import** java.util.List;

**public** **class** Fornecedor {

**private** Integer idFornecedor;

**private** String nome;

**private** String cnpj;

**private** List<Produto> produtos;

**public Fornecedor() {**

**// TODO Auto-generated constructor stub**

**}**

**public** Integer getIdFornecedor() {

**return** idFornecedor;

}

**public** **void** setIdFornecedor(Integer idFornecedor) {

**this**.idFornecedor = idFornecedor;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** String getCnpj() {

**return** cnpj;

}

**public** **void** setCnpj(String cnpj) {

**this**.cnpj = cnpj;

}

**public** List<Produto> getProdutos() {

**return** produtos;

}

**public** **void** setProdutos(List<Produto> produtos) {

**this**.produtos = produtos;

}

}

**Sobrecarga de métodos (**Overloading**)**

Prática utilizada em POO onde criamos métodos em uma classe que possuam o mesmo nome, porém, com entrada de argumentos diferentes.

Exemplo:

**class** Test {

**public void print() {**

System.***out***.println("Hello!");

}

**public void print(String name) {**

System.***out***.println("Hello, " + name + "!");

}

}

**class** Main {

**public** **static** **void** main(String[] args) {

**Test test = new Test();**

**test.print("Ana");**

}

}

**Sobrecarga de métodos construtores**

Podemos, em uma classe Java, criar quantos construtores quisermos. Todos eles terão o mesmo nome (nome da classe), porém terão entrada de argumentos diferentes.

Exemplo:

**package** entities;

**import** java.util.List;

**public** **class** Fornecedor {

**private** Integer idFornecedor;

**private** String nome;

**private** String cnpj;

**private** List<Produto> produtos;

**public Fornecedor() {**

**// TODO Auto-generated constructor stub**

**}**

**public Fornecedor(Integer idFornecedor, String nome,**

**String cnpj) {**

**this.idFornecedor = idFornecedor;**

**this.nome = nome;**

**this.cnpj = cnpj;**

**}**

**public Fornecedor(Integer idFornecedor, String nome,**

**String cnpj, List<Produto> produtos) {**

**super();**

**this.idFornecedor = idFornecedor;**

**this.nome = nome;**

**this.cnpj = cnpj;**

**this.produtos = produtos;**

**}**

**public** Integer getIdFornecedor() {

**return** idFornecedor;

}

**public** **void** setIdFornecedor(Integer idFornecedor) {

**this**.idFornecedor = idFornecedor;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** String getCnpj() {

**return** cnpj;

}

**public** **void** setCnpj(String cnpj) {

**this**.cnpj = cnpj;

}

**public** List<Produto> getProdutos() {

**return** produtos;

}

**public** **void** setProdutos(List<Produto> produtos) {

**this**.produtos = produtos;

}

}

**Criando os construtores da classe Produto:**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**package** entities;

**public** **class** Produto {

**private** Integer idProduto;

**private** String nome;

**private** Double preco;

**private** Integer quantidade;

**private** Fornecedor fornecedor;

**public Produto() {**

**// TODO Auto-generated constructor stub**

**}**

**public Produto(Integer idProduto, String nome,**

**Double preco, Integer quantidade) {**

**super();**

**this.idProduto = idProduto;**

**this.nome = nome;**

**this.preco = preco;**

**this.quantidade = quantidade;**

**}**

**public Produto(Integer idProduto, String nome,**

**Double preco, Integer quantidade,**

**Fornecedor fornecedor) {**

**super();**

**this.idProduto = idProduto;**

**this.nome = nome;**

**this.preco = preco;**

**this.quantidade = quantidade;**

**this.fornecedor = fornecedor;**

**}**

**public** Integer getIdProduto() {

**return** idProduto;

}

**public** **void** setIdProduto(Integer idProduto) {

**this**.idProduto = idProduto;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** Double getPreco() {

**return** preco;

}

**public** **void** setPreco(Double preco) {

**this**.preco = preco;

}

**public** Integer getQuantidade() {

**return** quantidade;

}

**public** **void** setQuantidade(Integer quantidade) {

**this**.quantidade = quantidade;

}

**public** Fornecedor getFornecedor() {

**return** fornecedor;

}

**public** **void** setFornecedor(Fornecedor fornecedor) {

**this**.fornecedor = fornecedor;

}

}

**Voltando na classe Program.java**

**package** principal;

**import** java.util.ArrayList;

**import** java.util.List;

**import** entities.Fornecedor;

**import** entities.Produto;

**public** **class** Program {

**public** **static** **void** main(String[] args) {

// criando um objeto Fornecedor

Fornecedor fornecedor = **new** Fornecedor

(1, "Loja de Informática", "38.179.298/0001-11");

// criando objetos da classe produto

Produto produto1 = **new** Produto

(1, "Notebook Dell", 5000.0, 10);

Produto produto2 = **new** Produto

(2, "Mouse Optico", 90.0, 20);

Produto produto3 = **new** Produto

(3, "Impressora Laser", 500.0, 15);

Produto produto4 = **new** Produto

(4, "Teclado sem Fio", 150.0, 10);

// criando uma lista de produtos

List<Produto> produtos = **new** ArrayList<Produto>();

// adicionando os produtos dentro da lista

produtos.add(produto1);

produtos.add(produto2);

produtos.add(produto3);

produtos.add(produto4);

//adicionando a lista de produtos no fornecedor

fornecedor.setProdutos(produtos);

}

}

**Interfaces**

São componentes de programação orientada a objetos através do qual podemos criar métodos abstratos que deverão ser implementados pelas classes que herdarem a interface.

Nas interfaces, iremos declarar **métodos abstratos**. Ou seja, métodos que não terão corpo, apenas assinatura. Quando uma classe IMPLEMENTAR uma interface, a classe deverá fornecer corpo para todos os métodos abstratos da interface.

Exemplo:

**//abstração!**

**interface IDispositivo {**

**void ligar(); //método abstrato**

**}**

Após a criação da interface, iremos desenvolver classes para implementar a interface. Ou seja, classes que serão implementações de Dispositivo.

Regra: Quando uma classe IMPLEMENTA uma interface, a classe deve programar (fornecer corpo) para todos os métodos da interface.

**class Lampada implements IDispositivo {**

**@Override**

**public void ligar() {**

System.***out***.println("Lampada acesa!");

}

}

Exemplo:

//abstração!

**interface** IDispositivo {

**void** ligar(); //método abstrato

}

**class** Lampada **implements** IDispositivo {

@Override

**public** **void** ligar() {

System.***out***.println("Lampada acesa!");

}

}

**class** Alarme **implements** IDispositivo {

@Override

**public** **void** ligar() {

System.***out***.println("Alarme disparado!");

}

}

Exemplo:

**interface** IDatabase {

**void** inserir();

**void** alterar();

**void** excluir();

**void** consultar();

}

**class** OracleDatabase **implements** IDatabase {

@Override

**public** **void** inserir() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** alterar() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** excluir() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** consultar() {

// **TODO** Auto-generated method stub

}

}

**class** MySqlDatabase **implements** IDatabase {

@Override

**public** **void** inserir() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** alterar() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** excluir() {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** consultar() {

// **TODO** Auto-generated method stub

}

}

**Sobrescrita de métodos (**Override**)**

Ocorre quando uma classe implementa (sobrescreve) um método de sua classe pai ou de sua interface. No Java, sempre que ocorrer uma implementação de método, a annotation **@Override** será incluída acima de declaração do método que foi implementado.

Exemplo:

**interface** IDispositivo {

**void** ligar(); //método abstrato}

**class** Lampada **implements** IDispositivo {

**@Override**

**public void ligar() {**

**System.*out*.println("Lampada acesa!");**

**}**

}

Criando uma interface para abstrair métodos relacionados a repositório de fornecedores.

Interface gráfica do usuário, Texto, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**package** interfaces;

**import** entities.Fornecedor;

**public** **interface** IFornecedorRepository {

**void** exportar(Fornecedor fornecedor) **throws** Exception;

}

Em seguida, vamos criar classes que implementem esta interface:

\*\* Primeiro, vamos criar uma classe que implemente a interface para gravar os dados do fornecedor em um arquivo texto.

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**package** repositories;

**import** java.io.PrintWriter;

**import** entities.Fornecedor;

**import** entities.Produto;

**import** interfaces.IFornecedorRepository;

**public** **class** FornecedorRepositoryTXT

**implements** IFornecedorRepository {

@Override

**public** **void** exportar(Fornecedor fornecedor) **throws** Exception {

PrintWriter printWriter = **new** PrintWriter

("c:\\temp\\fornecedor.txt");

printWriter.write("\nId do fornecedor...: "

+ fornecedor.getIdFornecedor());

printWriter.write("\nNome...............: "

+ fornecedor.getNome());

printWriter.write("\nCnpj...............: "

+ fornecedor.getCnpj());

//foreach (para cada..)

**for**(Produto produto : fornecedor.getProdutos()) {

printWriter.write("\n\tId do produto...: "

+ produto.getIdProduto());

printWriter.write("\n\tNome............: "

+ produto.getNome());

printWriter.write("\n\tPreço...........: "

+ produto.getPreco());

printWriter.write("\n\tQuantidade......: "

+ produto.getQuantidade());

}

printWriter.flush(); //salvar o arquivo

printWriter.close(); //fechar o arquivo

}

}

Por último, vamos criar uma classe para exportar os dados do fornecedor para um arquivo de extensão .CSV

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**package** repositories;

**import** java.io.PrintWriter;

**import** entities.Fornecedor;

**import** entities.Produto;

**import** interfaces.IFornecedorRepository;

**public** **class** FornecedorRepositoryCSV **implements** IFornecedorRepository {

@Override

**public** **void** exportar(Fornecedor fornecedor) **throws** Exception {

PrintWriter printWriter = **new** PrintWriter

("c:\\temp\\fornecedor.csv");

printWriter.write(fornecedor.getIdFornecedor()

+ ";" + fornecedor.getNome()

+ ";" + fornecedor.getCnpj());

**for**(Produto produto : fornecedor.getProdutos()) {

printWriter.write("\n" + produto.getIdProduto()

+ ";" + produto.getNome()

+ ";" + produto.getPreco()

+ ";" + produto.getQuantidade());

}

printWriter.flush();

printWriter.close();

}

}

**Polimorfismo (**muitas formas**)**

Permite que um objeto assuma formas diferentes conforme a sua inicialização. Dessa maneira os seus métodos, ao serem executados, irão gerar comportamentos diferentes de acordo com a maneira como o objeto foi inicializado.

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente

package principal;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JOptionPane;

import entities.Fornecedor;

import entities.Produto;

import interfaces.IFornecedorRepository;

import repositories.FornecedorRepositoryCSV;

import repositories.FornecedorRepositoryTXT;

public class Program {

public static void main(String[] args) {

// criando um objeto Fornecedor

Fornecedor fornecedor = new Fornecedor

(1, "Loja de Informática", "38.179.298/0001-11");

// criando objetos da classe produto

Produto produto1 = new Produto(1, "Notebook Dell", 5000.0, 10);

Produto produto2 = new Produto(2, "Mouse Optico", 90.0, 20);

Produto produto3 = new Produto(3, "Impressora Laser", 500.0, 15);

Produto produto4 = new Produto(4, "Teclado sem Fio", 150.0, 10);

// criando uma lista de produtos

List<Produto> produtos = new ArrayList<Produto>();

// adicionando os produtos dentro da lista

produtos.add(produto1);

produtos.add(produto2);

produtos.add(produto3);

produtos.add(produto4);

//adicionando a lista de produtos no fornecedor

fornecedor.setProdutos(produtos);

//criando um polimorfismo da interface

IFornecedorRepository fornecedorRepository = null; //vazio

String opcao = JOptionPane.showInputDialog("Informe CSV ou TXT:");

switch(opcao.toUpperCase()) {

case "TXT":

//POLIMORFISMO

fornecedorRepository = new FornecedorRepositoryTXT();

break;

case "CSV":

//POLIMORFISMO

fornecedorRepository = new FornecedorRepositoryCSV();

break;

}

try {

fornecedorRepository.exportar(fornecedor);

System.out.println("Dados gravados com sucesso!");

}

catch(Exception e) {

System.out.println("Erro: " + e.getMessage());

}

}

}

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Executando:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**Arquivos gerados:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente