Exceções e Controle de Erros

Capítulo IX

Sistema de Compras

1. O departamento de Compras da sua empresa solicitou o desenvolvimento de um sistema interno de aquisições visando eliminar a troca exagerada de e-mails.

   a. Cada compra é composta por um **Pedido**. Um pedido tem um *código único* e pode conter um ou vários **Item**, sendo que a quantidade máxima de itens é 3 (três).
   b. O sistema idealizado é similar ao de **Carrinho**, muito comum em portais de comércio eletrônico. Cada pedido deve implementar as seguintes funcionalidades de um carrinho: adicionar item, listar itens e calcular o valor total da compra.
   c. Um **Item** é composto de um *código de identificação*, *descrição*, *quantidade* e *valor unitário*.

Sistema de Compras

O método **getValorTotal** calcula e retorna o valor total do item

```java
public class Item {
    private final int ID;
    private String descricao;
    private int quantidade;
    private BigDecimal valorUnitario;
    private BigDecimal valorTotal;

    public Item(int ID){
        this.ID = ID;
        valorUnitario = new BigDecimal("1");
        valorTotal = new BigDecimal("1");

    }
    public int getId() {
        return ID;
    }
    public String getDescricao() {
        return descricao;
    }
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
    public int getQuantidade() {
        return quantidade;
    }
    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }
    public BigDecimal getValorUnitario() {
        return valorUnitario;
    }
    public void setValorUnitario(BigDecimal valorUnitario) {
        this.valorUnitario = valorUnitario;
    }
    public BigDecimal getValorTotal() {
        valorTotal = valorUnitario.multiply(new BigDecimal(quantidade));
        return valorTotal;
    }
}
```

Sistema de Compras

```java
public interface Carrinho {

    public abstract void adicionarItem(Item it);
    public abstract void listarItens();
    public abstract BigDecimal calcularValorTotal();
```

```java
public class Pedido implements Carrinho{
    private int id;
    private Item[] itens;
    private final int MAXIMO_ITENS;
    private int qtdeItens;
    public Pedido(int ID){
        MAXIMO_ITENS = 3;
        qtdeItens = 0;
        itens = new Item[MAXIMO_ITENS];
    }
    @Override
    public void adicionarItem(Item it) {
            itens[getQtdeItens()] = it;
            qtdeItens++;
    }
    @Override
    public void listarItens() {
        for(Item it : itens){
            if(it != null){
                System.out.println("ID: " + it.getId());
                System.out.println("Descrição: " + it.getDescricao());
                System.out.println("Quantidade: " + it.getQuantidade());
                System.out.println("Valor unitário: " + it.getValorUnitario());
                System.out.println("Valor total do item: " + it.getValorTotal());
            }
        }
        System.out.println("Valor total do pedido: " + calcularValorTotal());
    }
```

```java
    @Override
    public BigDecimal calcularValorTotal() {
        BigDecimal total = new BigDecimal("0");
        for(Item it : itens){
            if(it != null)
                total = total.add(it.getValorTotal());
        }
        return total;

    }

    public int getId() {
        return id;
    }

    public int getMAXIMO_ITENS() {
        return MAXIMO_ITENS;
    }

    public int getQtdeItens() {
        return qtdeItens;
    }

}
```

Sistema de Compras

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

    int op;
    Scanner teclado = new Scanner(System.in);
    Pedido po = null;
    Item it;
    Random geradorID = new Random();
    do {
        System.out.println("---------------------------------------");
        System.out.println("SISTEMA DE COMPRAS DA EMPRESA ABCD S.A.");
        System.out.println("Digite a opção desejada");
        System.out.println("1- Para cadastrar um novo Item no pedido");
        System.out.println("2- Para visualizar todos os itens do pedido");
        System.out.println("9- Para sair");
        op = teclado.nextInt();
        System.out.println("---------------------------------------");
        switch(op){
            case 1:
                if(po == null)
                    po = new Pedido(geradorID.nextInt());
                System.out.println("Entre com a identificação do item");
                it = new Item(teclado.nextInt());
                teclado.nextLine();
                System.out.println("Entre com a descrição do item");
                it.setDescricao(teclado.nextLine());
                System.out.println("Entre com a quantidade deste item");
                it.setQuantidade(teclado.nextInt());
                System.out.println("Entre com o valor unitário deste item");
                teclado.nextLine();
                it.setValorUnitario(new BigDecimal(teclado.nextLine()));
                po.adicionarItem(it);
                break;
```

Sistema de Compras

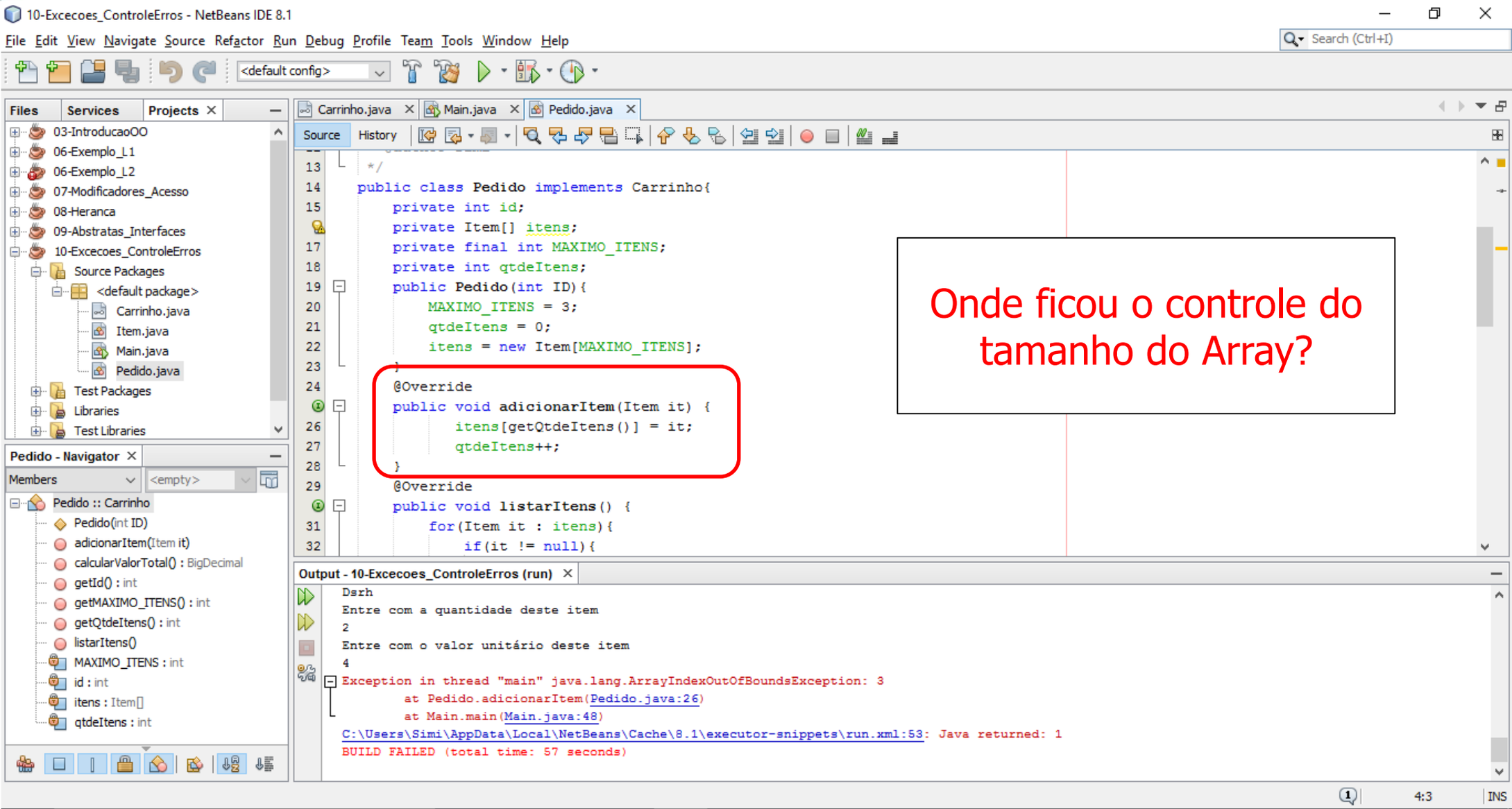
```
            case 2:
                if(po !=null)
                    po.listarItens();
                else
                    System.out.println("Pedido não criado!");
                break;
            case 9:
                System.out.println("Obrigado por usar nosso sistema!");
                break;
            default:
                System.out.println("Opção inválida!");
        }
    } while (op!=9);

}
```

Se o pedido não foi criado, então não podemos listar os itens !

Sistema de Compras

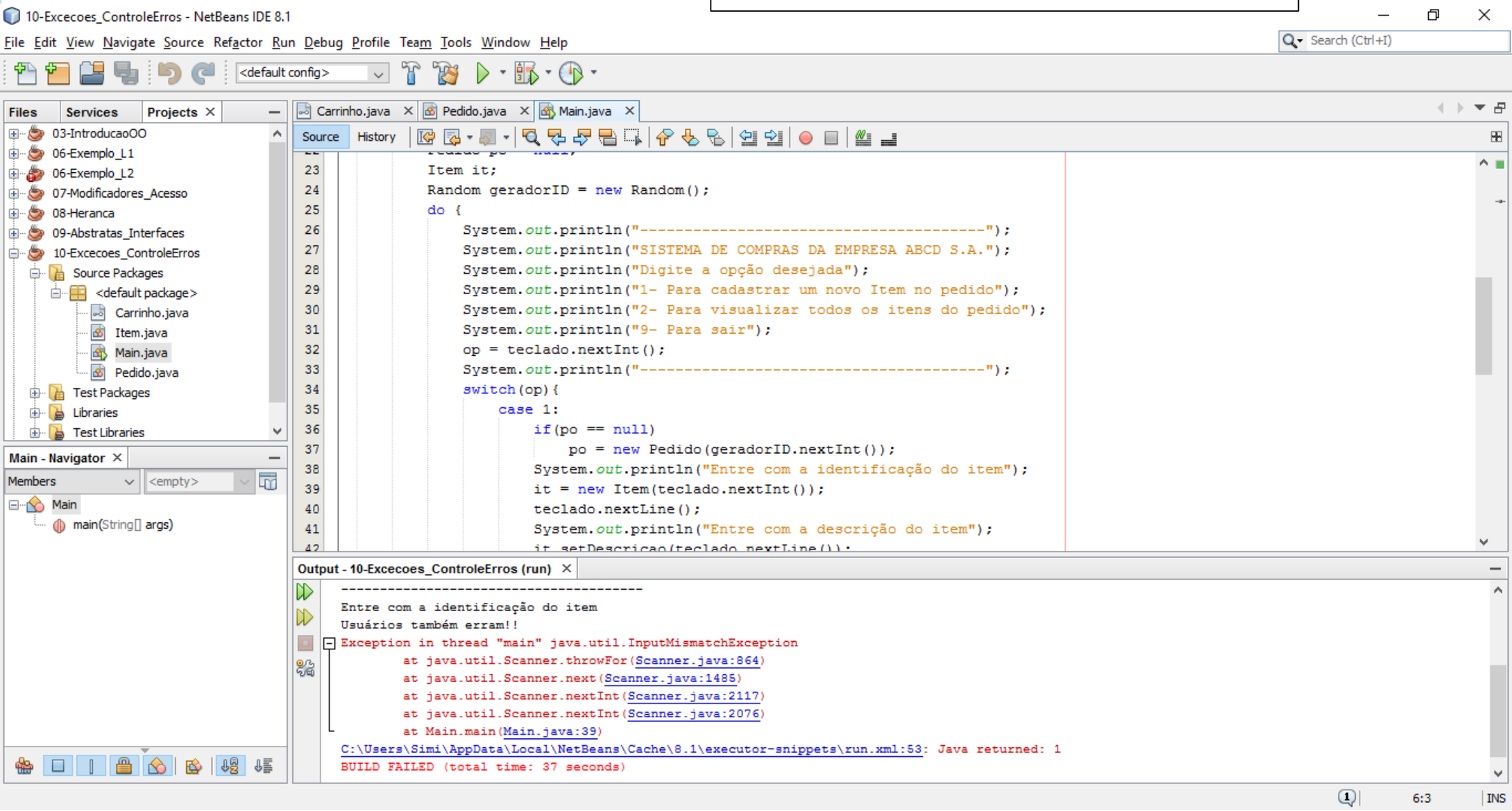
Se esse fosse um Sistema **real**, quais seriam os problemas?

Sistema de Compras



Sistema de Compras

A entrada dos dados pode não ser aquilo que esperamos!



Sistema de Compras

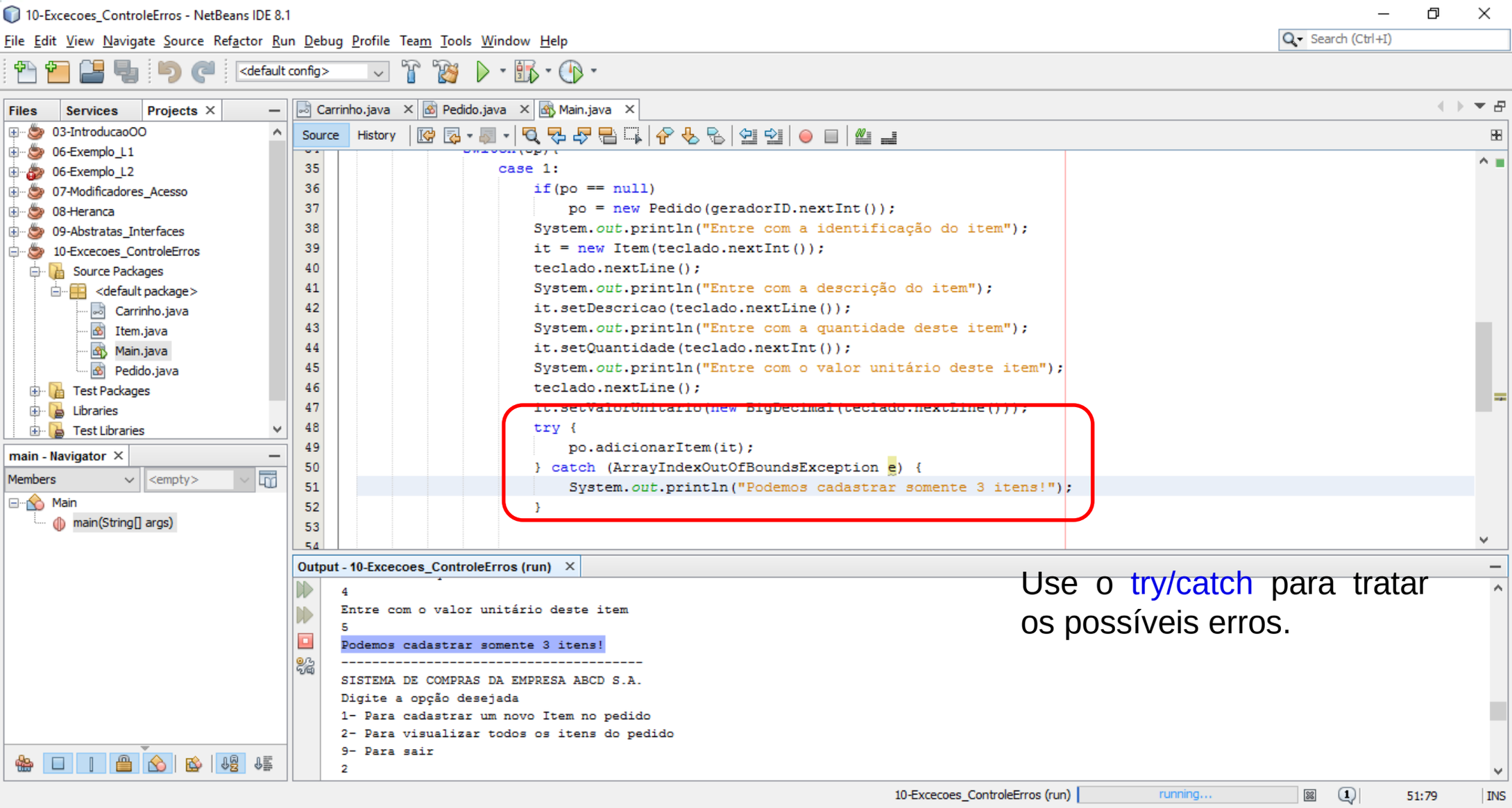
```
public class Pedido implements Carrinho{
    private int id;
    private Item[] itens;
    private final int MAXIMO_ITENS;
    private int qtdeItens;
    public Pedido(int ID){
        MAXIMO_ITENS = 3;
        qtdeItens = 0;
        itens = new Item[MAXIMO_ITENS];
    }
    @Override
    public void adicionarItem(Item it) throws ArrayIndexOutOfBoundsException{
            itens[getQtdeItens()] = it;
            qtdeItens++;
    }
```

Capturar exceções para fazer tratamento de erros é uma boa prática!

Quando o seu método não souber o que fazer com a exceção ele deve passar para quem têm esse controle... Use o throws

```
try {
    po.adicionarItem(it);
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Podemos cadastrar somente 3 itens!");
}
```

Sistema de Compras



Use o try/catch para tratar os possíveis erros.

Tipos de Exceções

Checked: Exceções que obrigatoriamente devem ser tratadas.

Ex: FileNotFoundException, IOException;

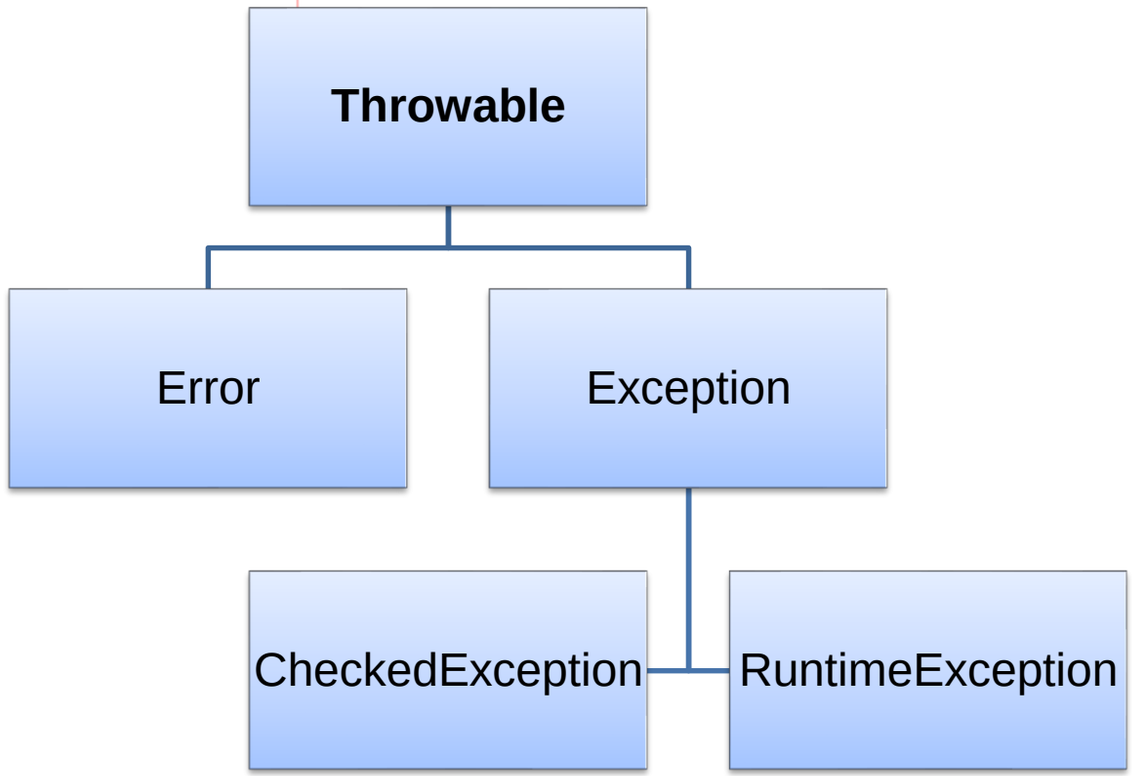
Unchecked: Exceções que podem ou não ser tratadas.

Ex: NullPointerException, ArithmeticException.

Sistema de Compras

```
void connect() throws ConnectionFailedException {
    try {
        doConnection(); //Faz a conexão, mas lança várias exceções específicas
    } catch ( SocketException e) {
        throw new ConnectionFailedException(e);
    }
    catch ( ServerException e) {
        throw new ConnectionFailedException(e);
    } catch ( IOException e) {
        throw new ConnectionFailedException(e);
    }
```

Você também pode criar sua própria exceção, mas use com moderação, estendendo da classe Exception ou RuntimeException



Finally

```
try
{
    // statements that might cause exceptions
    // possibly including function calls
}
catch ( exception-1 ex-1 )
{
    // statements to handle this exception
}
catch ( exception-2 ex-2 )
{
    // statements to handle this exception
  .
  .
}
finally
{
    // statements to execute every time this try block
    executes
}
```

JOptionPane

```
showConfirmDialog(Component parentComponent, Object message)
showConfirmDialog(Component parentComponent, Object message, String title, int optionType)
showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)
showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)
```

```
showInputDialog(Object message)
showInputDialog(Component parentComponent, Object message)
showInputDialog(Object message, Object initialSelectionValue)
showInputDialog(Component parentComponent, Object message, Object initialSelectionValue)
showInputDialog(Component parentComponent, Object message, String title, int messageType)
```

```
showMessageDialog(Component parentComponent, Object message)
showMessageDialog(Component parentComponent, Object message, String title, int messageType)
showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)
```

JOptionPane

ParentComponent: qual componente da GUI é seu pai. Serve para posicionar o JOptionPane. Caso não existir pai utilizasse null para centraliza-lo na tela;
ObjectMessage: objeto com valor que se deseja mostrar. Pode ser String, int, boolean ou qualquer outro objeto;
Title: título da mensagem. Fica no cabeçalho da janela;
OptionType: configuração dos botões (quais botões iram aparecer);
MessageType: Modica o ícone da mensagem para alguns predefinidos;
Icon: recebe um objeto do tipo Icon que o usuário escolher e substitui o ícone predefinido;
InitialValueSection: valor default para o campo.

Exercícios

Faça crítica das entradas de dados e divisão por zero através de Exceção

1. Implemente uma calculadora que realize as seguintes operações entre 2 números: soma, subtrair, dividir e multiplicar.

1. A empresa na qual você trabalha decidiu variar os tipos de investimentos que ela realiza. Para isso, ela criou uma área que irá investir no Mercado de capitais (Bolsa de Valores) e você foi designado para implementar a solução que irá receber as cotações da BOVESPA.

   As funcionalidades implementadas devem ser de cadastro das ações, visualização e atualização das informações das cotações.

   Nesta primeira fase os valores das ações serão digitados manualmente pelos operadores, as informações que são cadastradas e utilizadas para negociação de ações são:

   i. código = 5 caracteres (ex.:AMBV4 [Bebidas Ambev], BBDC4 [Banco Bradesco])
   ii. data = dd/mm/aaaa (ex.: 16/08/2016)
   iii. horário = hh:mm (ex.: 16:40)
   iv. cotação = dd,cc (06,78)
   v. Observação: em um dia podem ser negociadas até 5 ações

Obrigado!