

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

```
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

```
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());
    }
}
```

```
public interface Carrinho {

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

    @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

10-Excecoes_ControlErros - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Files Services Projects

- 03-IntroducaoOO
- 06-Exemplo_L1
- 06-Exemplo_L2
- 07-Modificadores_Acesso
- 08-Heranca
- 09-Abstratas_Interfaces
- 10-Excecoes_ControlErros
 - Source Packages
 - <default package>
 - Carrinho.java
 - Item.java
 - Main.java
 - Pedido.java
 - Test Packages
 - Libraries
 - Test Libraries

Pedido - Navigator

Members

- Pedido :: Carrinho
 - Pedido(int ID)
 - adicionarItem(Item it)
 - calcularValorTotal() : BigDecimal
 - getId() : int
 - getMAXIMO_ITENS() : int
 - getQtdeItens() : int
 - listarItens()
 - MAXIMO_ITENS : int
 - id : int
 - itens : Item[]
 - qtdeItens : int

Source

```
13  /*
14  public class Pedido implements Carrinho{
15      private int id;
16      private Item[] itens;
17      private final int MAXIMO_ITENS;
18      private int qtdeItens;
19      public Pedido(int ID){
20          MAXIMO_ITENS = 3;
21          qtdeItens = 0;
22          itens = new Item[MAXIMO_ITENS];
23      }
24      @Override
25      public void adicionarItem(Item it) {
26          itens[getQtdeItens()] = it;
27          qtdeItens++;
28      }
29      @Override
30      public void listarItens() {
31          for(Item it : itens){
32              if(it != null){
```

Onde ficou o controle do tamanho do Array?

Output - 10-Excecoes_ControlErros (run)

```
Dsrh
Entre com a quantidade deste item
2
Entre com o valor unitário deste item
4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
    at Pedido.adicionarItem(Pedido.java:26)
    at Main.main(Main.java:48)
C:\Users\Simi\AppData\Local\NetBeans\Cache\8.1\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 57 seconds)
```


Sistema de Compras

A entrada dos dados pode não ser aquilo que esperamos!

The screenshot shows the NetBeans IDE 8.1 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains icons for file operations, running, and debugging. The left sidebar shows the 'Files' view with a project tree for '10-Excecoes_ControlErros' containing 'Carrinho.java', 'Item.java', 'Main.java', and 'Pedido.java'. The 'Main - Navigator' view shows the 'Main' class with a 'main(String[] args)' method. The central editor displays the code for 'Main.java', which includes a loop for user input and a switch statement for menu options. The bottom pane shows the 'Output - 10-Excecoes_ControlErros (run)' window, which displays the program's execution and a runtime exception.

```
23 Item it;
24 Random geradorID = new Random();
25 do {
26     System.out.println("-----");
27     System.out.println("SISTEMA DE COMPRAS DA EMPRESA ABCD S.A.");
28     System.out.println("Digite a opção desejada");
29     System.out.println("1- Para cadastrar um novo Item no pedido");
30     System.out.println("2- Para visualizar todos os itens do pedido");
31     System.out.println("9- Para sair");
32     op = teclado.nextInt();
33     System.out.println("-----");
34     switch(op){
35     case 1:
36         if(po == null)
37             po = new Pedido(geradorID.nextInt());
38         System.out.println("Entre com a identificação do item");
39         it = new Item(teclado.nextInt());
40         teclado.nextLine();
41         System.out.println("Entre com a descrição do item");
42         it.setDescricao(teclado.nextLine());
```

Output - 10-Excecoes_ControlErros (run)

```
-----
Entre com a identificação do item
Usuários também erram!!
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at Main.main(Main.java:39)
C:\Users\Simi\AppData\Local\NetBeans\Cache\8.1\executor-snippets\run.xml:53: Java returned: 1
BUILD FAILED (total time: 37 seconds)
```

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

10-Excecoes_ControlErros - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Files Services Projects

03-IntroducaoOO
06-Exemplo_L1
06-Exemplo_L2
07-Modificadores_Acesso
08-Heranca
09-Abstratas Interfaces
10-Excecoes_ControlErros
Source Packages
<default package>
Carrinho.java
Item.java
Main.java
Pedido.java
Test Packages
Libraries
Test Libraries

main - Navigator
Members
Main
main(String[] args)

Carrinho.java x Pedido.java x Main.java x

Source History

```

35 case 1:
36     if(po == null)
37         po = new Pedido(geradorID.nextInt());
38     System.out.println("Entre com a identificação do item");
39     it = new Item(teclado.nextInt());
40     teclado.nextLine();
41     System.out.println("Entre com a descrição do item");
42     it.setDescricao(teclado.nextLine());
43     System.out.println("Entre com a quantidade deste item");
44     it.setQuantidade(teclado.nextInt());
45     System.out.println("Entre com o valor unitário deste item");
46     teclado.nextLine();
47     it.setValorUnitario(new BigDecimal(teclado.nextLine()));
48     try {
49         po.adicionarItem(it);
50     } catch (ArrayIndexOutOfBoundsException e) {
51         System.out.println("Podemos cadastrar somente 3 itens!");
52     }
53
54

```

Output - 10-Excecoes_ControlErros (run)

```

4
5
6 Entre com o valor unitário deste item
7
8 Podemos cadastrar somente 3 itens!
9
10 -----
11 SISTEMA DE COMPRAS DA EMPRESA ABCD S.A.
12 Digite a opção desejada
13 1- Para cadastrar um novo Item no pedido
14 2- Para visualizar todos os itens do pedido
15 9- Para sair
16
17 2

```

Use o **try/catch** para tratar os possíveis erros.

10-Excecoes_ControlErros (run) | running... | 51:79 | INS

Tipos de Exceções

Checked: Exceções que obrigatoriamente devem ser tratadas.

Ex: FileNotFoundException, IOException;

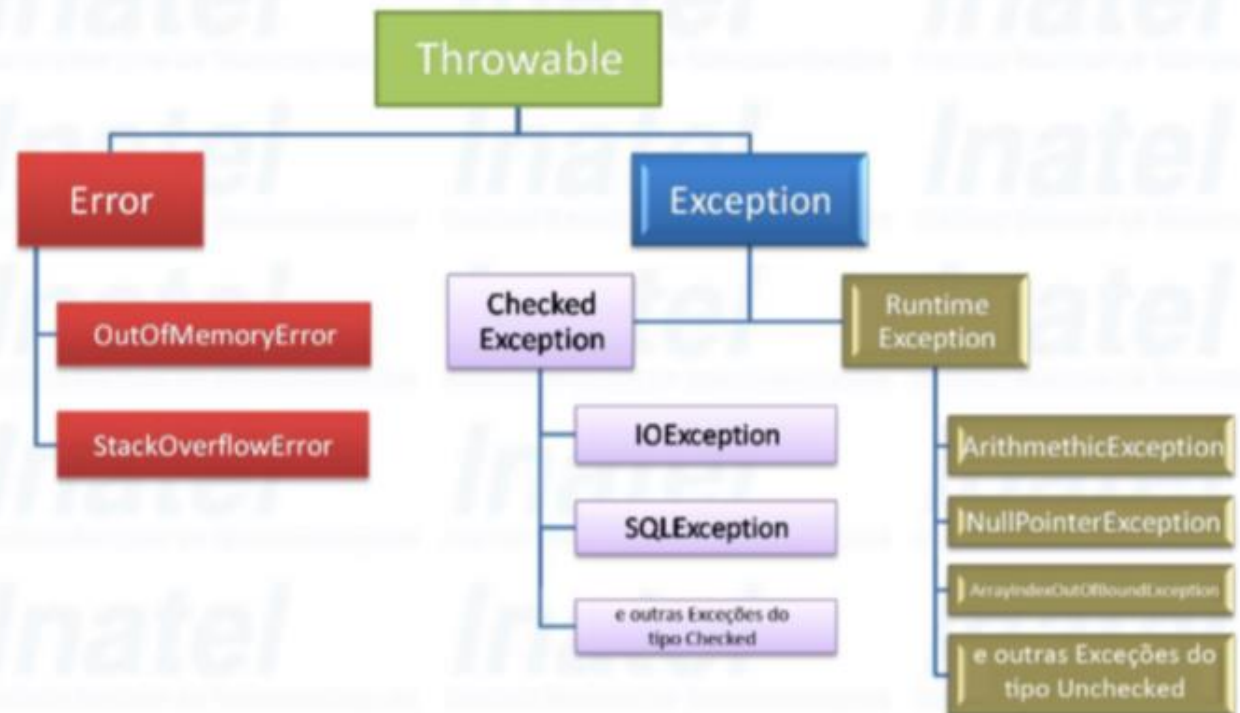
Unchecked: Exceções que podem ou não ser tratadas.

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

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

```
❶ showInputDialog(Object message)
❷ showInputDialog(Component parentComponent, Object message)
❸ showInputDialog(Object message, Object initialSelectionValue)
❹ showInputDialog(Component parentComponent, Object message, Object initialSelectionValue)
❺ showInputDialog(Component parentComponent, Object message, String title, int messageType)
```

```
❶ showMessageDialog(Component parentComponent, Object message)
❷ showMessageDialog(Component parentComponent, Object message, String title, int messageType)
❸ 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;

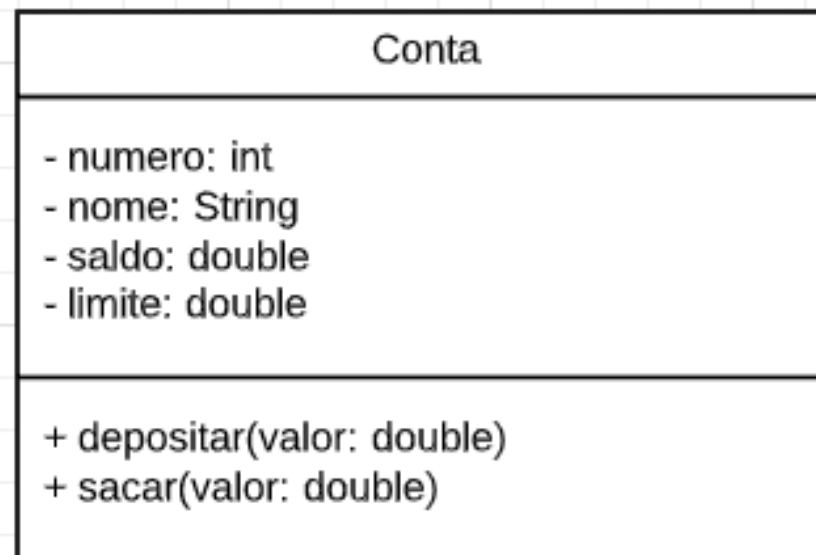
Icon: recebe um objeto do tipo Icon que o usuário escolher e substitui o ícone predefinido;

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.
2. Fazer um programa para ler os dados de uma conta bancária e depois realizar um saque nesta conta bancária, mostrando o novo saldo. Um saque não pode ocorrer ou se não houver saldo na conta, ou se o valor do saque for superior ao limite de saque da conta. Portanto crie um exceção para tratar estes erros. Implemente a conta bancária conforme projeto abaixo.



Obrigado!