



# Java Excellence

## Application Delivery Fundamentals: Java



### Module 5: Tratamento de Exceção

# Module Objectives

- No final deste módulo, você será capaz de:
  - Explique o conceito de exceções e asserções.
  - Explique o uso de exceções e asserções.
  - Gerencie exceções usando try-catch-finally.
  - Crie exceções personalizadas e condições de exceção.
  - Use declarações de asserção para melhorar a qualidade do código.



# Exceptions

**“Exceptions”**

# Exceptions

## Exceções

- Mecanismo para tratar os erros durante a execução do programa.
- Exceções são classes que representam erros durante a execução de um programa java.
- Os objetos que representam as exceções podem ser “lançados”, capturados e manipulados.

# Exceptions

- A exceção é:
  - Um evento que acontece durante a execução do programa e impede que o programa continue normalmente.
  - Uma condição de erro que altera o fluxo normal de controle em um programa.
  - Um sinal que indica alguma condição inesperada ocorreu no programa.

# Exceptions

## Condições de Erro

- Tentar abrir um arquivo que não existe
- Tentar usar uma conexão de rede interrompida
- Tentar acessar posição de arrays fora da faixa
- Tentar carregar uma classe que não está no classpath

# Exceptions

- A exceção:



# Exceptions

- A exceção:

## Exemplo

```
public class Semaforo {  
    public static void main (String args[ ]) {  
        int i = 0;  
        String semaforo [ ] = {"Verde", "Amarelo", "Vermelho"};  
        while (i < 4) {  
            System.out.println (sinais[i]);  
            i++;  
        }  
    }  
}
```

Ao executar o resultado é

java Semaforo

Verde

Amarelo

Vermelho

java.lang.ArrayIndexOutOfBoundsException: 3  
at Semaforo.main(Semaforo.java:12)



# Exceptions

- Tratamento de exceção:



# Exceptions

- Tratamento de exceção:

```
Conta c = new Conta();  
try {  
    c.saque(100);  
} catch (EstouroSaqueException e) {  
    System.out.println("Erro: " + e.getMessage() );  
}  
c.mostraSaldo();
```

# Using try-catch-finally Blocks

```
try {  
    /*  
     * some codes to test here  
     */  
} catch (SQLException sx) {  
    /*  
     * handle Exception1 here  
     */  
} catch (IOException ix) {  
    /*  
     * handle Exception2 here  
     */  
} catch (Exception ex) {  
    /*  
     * handle Exception3 here  
     */  
} finally {  
    /*  
     * always execute codes here  
     */  
}
```

**Try block** inclui o contexto em que uma possível exceção pode ser lançada

Cada **Catch() block** é um manipulador de exceções e pode aparecer várias vezes

O opcional **Finally block** é sempre executado antes de sair da instrução **Try**.



Refer to the TryCatchFinallySample.java sample code.

# Exceptions

## Instrução *throw* e cláusula *throws*

```
public void deposito(double valor) throws
DepositoInvalidoException {
    if (valor > 0) {
        saldo = saldo + valor;
    } else {
        throw new DepositoInvalidoException("Este
valor é invalido para depósito: " + valor);
    }
}
```

# Exceptions

## Tratar ou lançar?

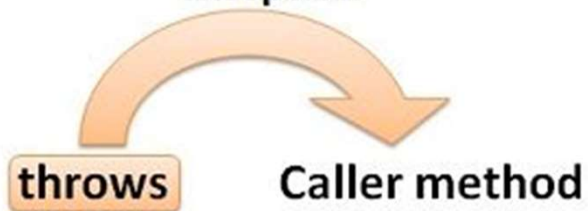
- O tratamento de exceções pode ser de 2 modos:
- Tratar dentro do próprio método usando try, catch e finally
- Lançar para quem chamou o método onde a exceção foi gerada usando a clausula throws

# Exceptions

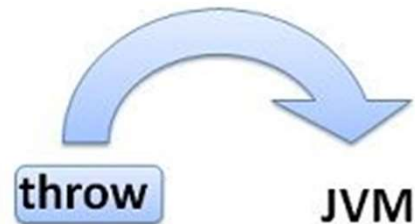
throws He will throw



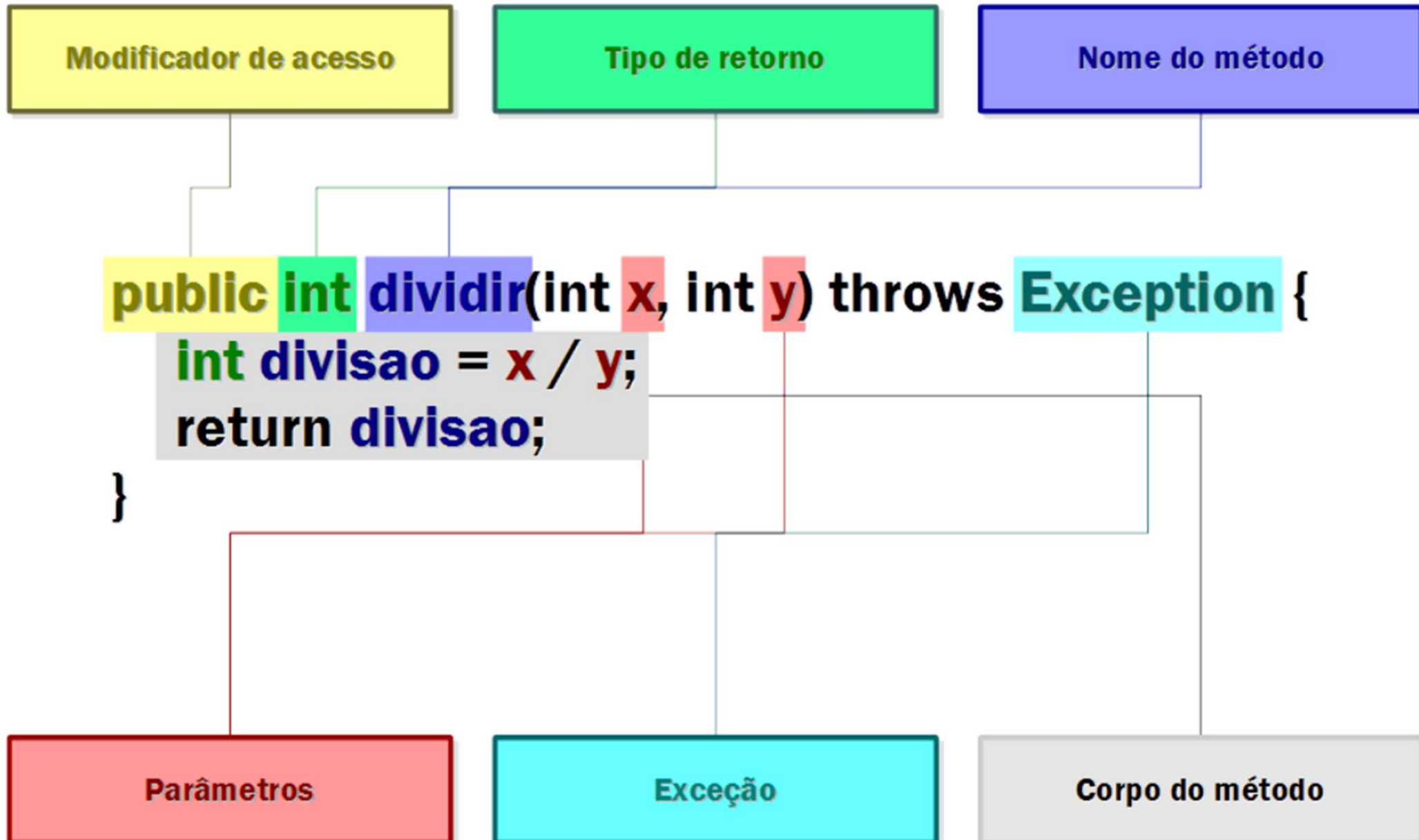
Responsibility of handling  
exception



throw He is throwing  
right now

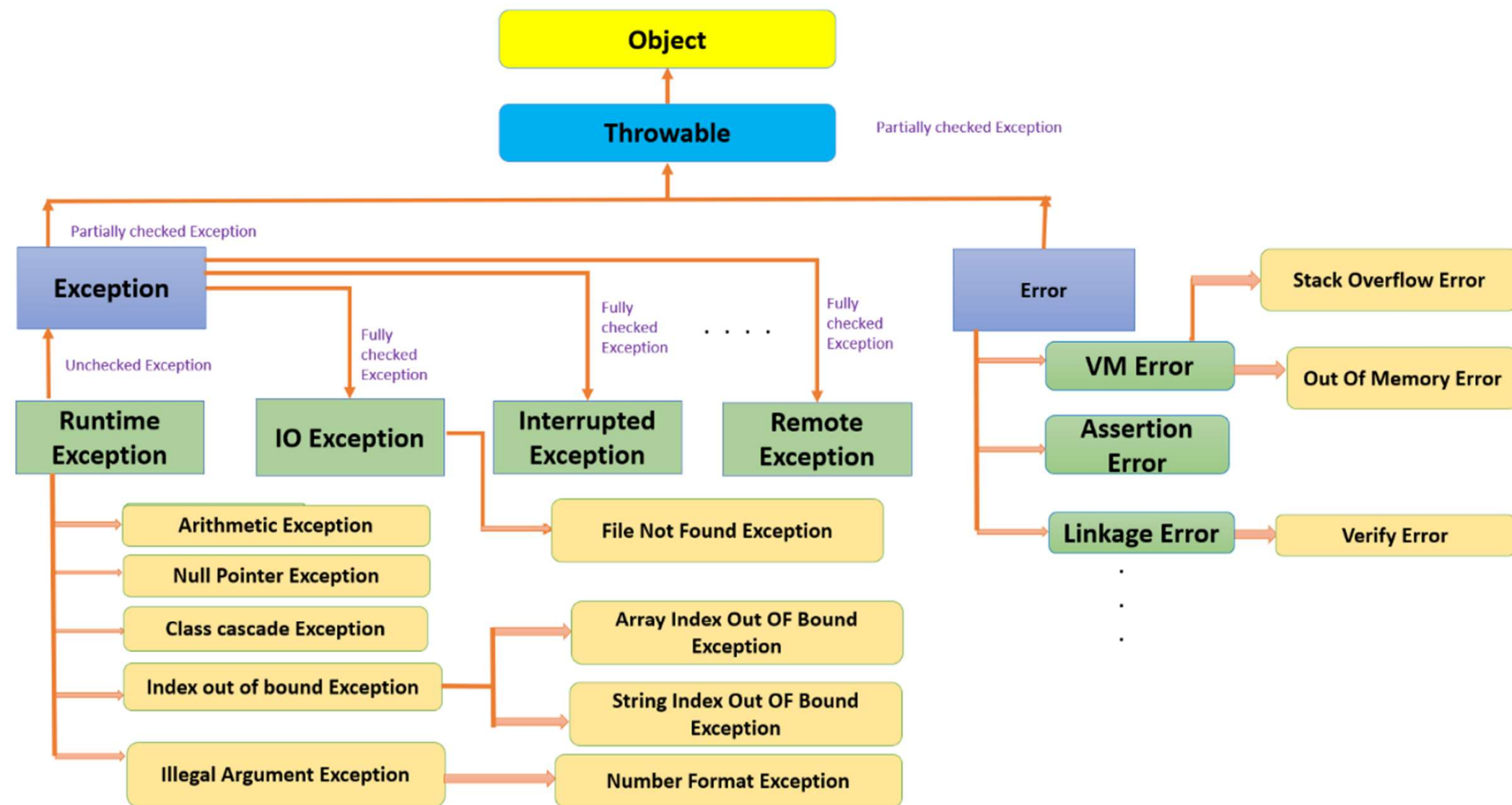


# Exceptions






# Exception Class Hierarchy





# Customizing Exceptions

- Exceções na API padrão podem não ser suficientes para cobrir os cenários necessários ao aplicativo.
- Uma exceção personalizada pode ser declarada subclassificando a classe Exception.
- A exceção personalizada deve conter dados e comportamento apropriados para ajudar na identificação e correção do problema.

 Refer to the CustomException.java and CustomExceptionSample.java sample code.

# Activity 2 – Custom Exception

- Nesta atividade, você:
  - Abra os arquivos ‘CustomExceptionActivity.java’ e ‘CustomExceptionActivityTest.java’ no pacote `sef.module8.activity`.
  - Leia as instruções e crie o código para concluir este programa.



# Exceptions

## Criando Exceções

```
public class DepositoInvalidoException extends Exception {  
  
    public DepositoInvalidoException (String motivo) {  
        // chama construtor da classe pai  
        super(motivo);  
    }  
}
```

# Exceptions

- Existem muitos métodos que são úteis ao implementar o tratamento de exceções em Java:
- 1. **getMessage()**- Esta função retorna uma mensagem resumida sobre a exceção que ocorreu. Ele também é inicializado no construtor de uma classe Throwable. A sintaxe deste método é `public String getMessage();`
- 2. **getCause()**- Esta função retorna a causa da exceção que surge no programa na forma de um Throwable Object. A sintaxe é `public Throwable getCause();`
- 3. **toString()**- Este método retorna a string que contém o nome da string anexada à mensagem de exceção. A sintaxe é `public String toString();`

# Exceptions

- 4. **printStackTrace()**-Esta função imprime o rastreamento de pilha do sistema para o fluxo de saída de erro. A sintaxe desta função é `public void printStackTrace();`
- 5. **getStackTrace()**- Esta função é responsável por retornar um array que contém os elementos de rastreamento de pilha. O último elemento é a parte inferior da pilha e o primeiro elemento é o topo do rastreamento de pilha.
- 6. **fillnStackTrace()**- Retorna um objeto do tipo `throwable`. Este objeto é o novo rastreamento de pilha da exceção. No entanto, o rastreamento de pilha mais antigo não é excluído. A sintaxe é `public throwable fillnStackTrace();`



# Atividade 8 - Exceptions

## Exercício

1. Crie as seguintes exceções no pacote **erros**:
  - a. **EstouroSaqueException** - Acontece quando é feita uma tentativa de tirar mais dinheiro do que a conta possui.
  - b. **DepositoInvalidoException** - Acontece quando uma quantia inválida de dinheiro ( $\text{quantia} < \text{zero}$ ) é depositada.
2. Reescreva as operações de saque e depósito para lançar estas exceções.



# Questions and Comments

- What questions or comments do you have?



# Checkpoint Question

**KAHOOT!**

