



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

TRATAMENTO DE EXCEÇÕES

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

Exceção

2

- Situação anormal que altera ou interrompe o fluxo normal da execução.
- Essa situação pode ser causada por:
 - Erros de hardware;
 - Erros de programação;
 - Erro de divisão por zero;
 - Erro na leitura/abertura de arquivos;
 - Falha na memória;
 - Valores de variáveis;

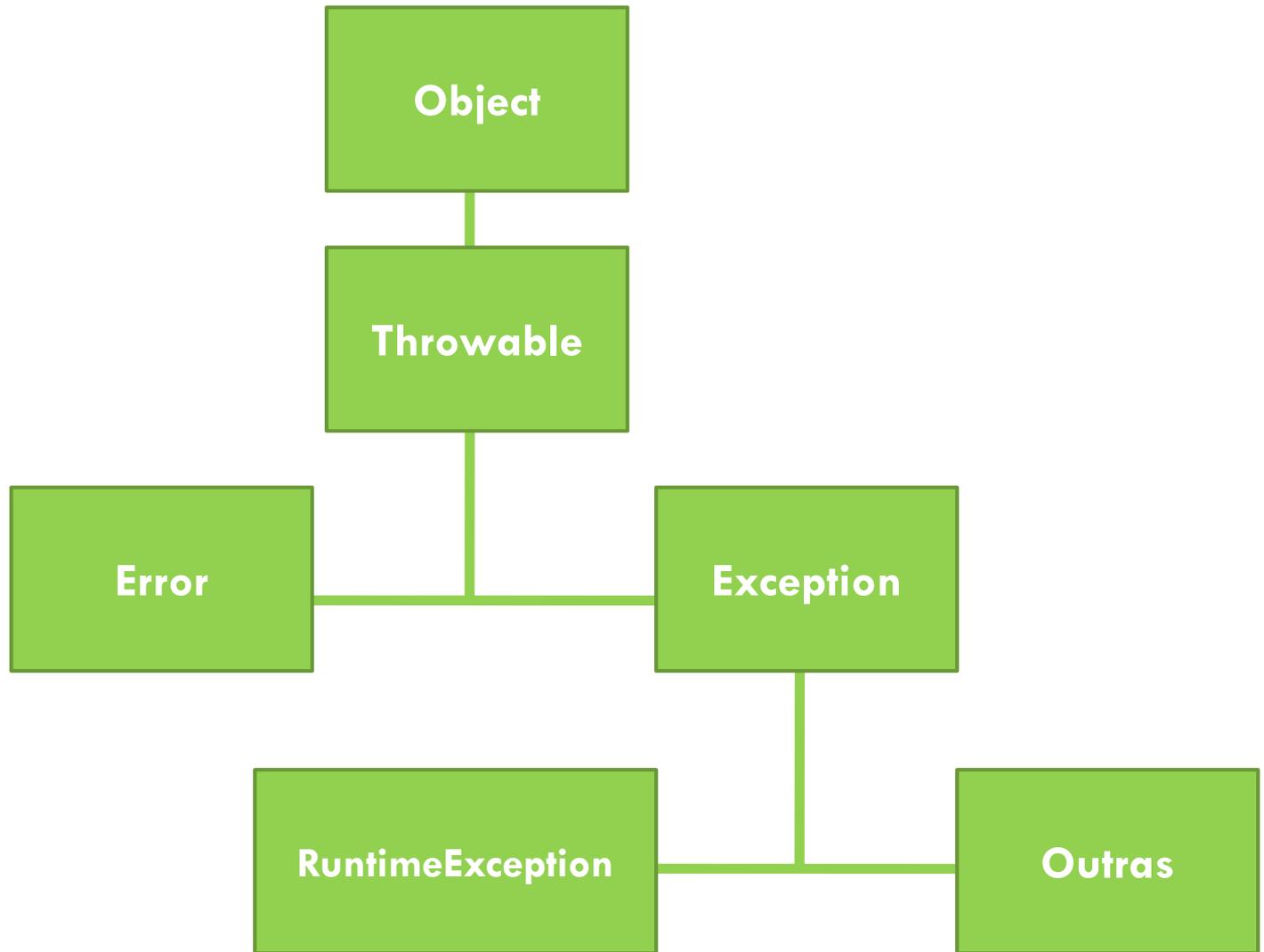
Exceção

3

- E se não tratarmos ?? O que acontece ??
 - O sistema pode parar
- Para que isso não ocorra é realizado um tratamento nessas exceções
- Se o tratamento for bem realizado, o programa se torna mais confiável.

Tipos de exceções

4



Tipos de exceções

5

- Podemos classificar as exceções em 3 tipos:
 - Verificadas;
 - Não verificadas;
 - Erros.

Exceções Verificadas

6

- São verificadas no momento da compilação.
- São as exceções do tipo **Exception** e todos os seus subtipos, exceto **RuntimeException** e seus subtipos.
- Todas essas exceções devem ser tratadas:
 - Utilizando o bloco **try – catch**
 - Ou na declaração do método informando que ele pode lançar uma exceção, utilizando a palavra **throws**.

Exceções Não Verificadas

7

- ❑ **NÃO** verificadas no momento da compilação.
- ❑ Geralmente são erro de programador (ex: divisão por zero).
- ❑ São do tipo `RuntimeException` e seus subtipos.
- ❑ Não precisam de tratamento.

Erros

8

- São irrecuperáveis e apresentam condições sérias.
 - OutOfMemoryError
 - StackOverflowError
 - VirtualMachineError



Tratando as exceções

- Vimos anteriormente que as exceções verificadas precisam de tratamento.
- O bloco `try – catch` é utilizado para tratar exceções.
- Os passos do tratamento de exceções são:
 1. Tenta executar o código (**try**)
 2. Se o código lançar algum erro, este é capturado (**catch**)
 3. Executa um trecho de código para finalizar (**finally**)

Tratando as exceções

10

```
int x=0;

try{

    int y=100/x;
    System.out.println("Resultado: "+y);

} catch (ArithmeticException) {

    System.out.println("Operação inválida");
    System.out.println("\nDetalhes do erro: " + e);

} finally {

    System.out.println("Finalizando!!");

}
```



**TENTA
EXECUTAR**



**CAPTURA A
EXCEÇÃO**



**EXECUTA
SEMPRE**

Tratando as exceções

- ❑ Tenta executar o código do bloco **try**.
- ❑ Se ocorrer uma exceção, para a execução no momento da exceção e vai executar o bloco **catch**.
- ❑ Pode haver mais de um bloco **catch**.
- ❑ O bloco **finally** é opcional.

Tratando as exceções

12

```
Scanner sc = new Scanner(System.in);  
try{  
    int x = sc.nextInt();  
    int y = 100 / x;  
    System.out.println ("Resultado: " + y);  
}  
catch (InputMismatchException e){  
    System.out.println ("Formato inválido!");  
    System.out.println("\n Detalhes do erro:" + e.getMessage());  
}  
catch (ArithmeticException e){  
    System.out.println ("Operação inválida!");  
    System.out.println("\n Detalhes do erro:" + e.getMessage());  
}
```

Tratando as exceções

13

- A palavra reservada **throws** informa que o método pode lançar uma exceção do tipo declarado.

```
public FileInputStream(String name) throws  
    FileNotFoundException {  
    //...  
}
```

Lançando uma exceção

14

- Para lançar uma exceção propositalmente usamos a palavra reservada **throw** e uma instância da **Exception**.

```
if (arquivo == null)
    throw new FileNotFoundException();
```

Exercício 01

15

1. Crie um programa em Java para ler um número, digitado pelo, e depois calcular o quadrado desse número.
 1. Você deve realizar o tratamento da exceção () caso o usuário digite um número inválido (ex: 3w).
 2. Apresente a mensagem “Opa... Acho que você se enganou! Digite novamente...”, a mensagem do erro e leia o número novamente, até o usuário digitar um número correto.

Exercício 02

16

1. Crie uma classe `Conta` que possui um saldo como atributo e os métodos `sacar` (retirar um valor do saldo) e `depositar` (adicionar um valor ao saldo).
2. Crie uma nova classe `SaldoException` que irá estender de `Exception`.
3. Sobrescreva o método `getMessage()`, que retorna uma `String` com a mensagem de erro da exception.
4. Na classe `Conta` (método `sacar`) se o valor a ser sacado for maior que o saldo, seu programa deverá lançar a `SaldoException`.
5. Crie uma classe `Teste` e realize testes da sua classe `Conta`.