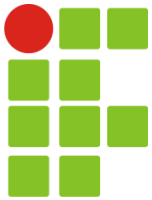


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

PROGRAMAÇÃO COM ACESSO A BANCO DE DADOS
TRATAMENTO DE EXCEÇÕES

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

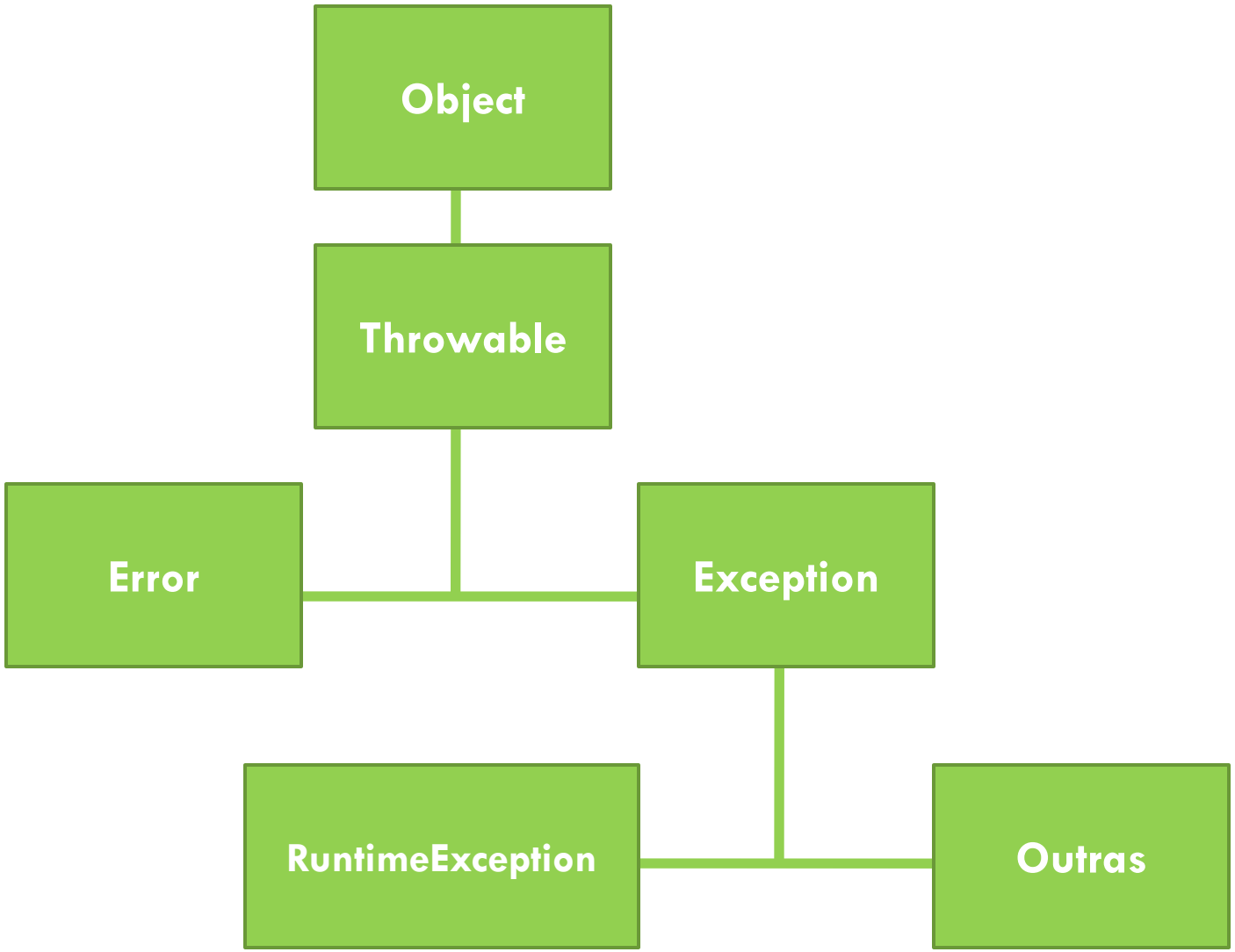
Exceção

- 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

- 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



Tipos de exceções

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

Exceções Verificadas

- 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

- **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

- 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


```
intx=0;

try{

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

} catch(ArithmeticExceptione){

    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


```
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

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

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

Lançando uma exceção

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

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