Programação Orientada a Objetos

Definição e tratamento de exceções

Prof. Leandro Rodrigues Pinto
<leandrorodp@gmail.com>

Introdução a exceções em POO

“Uma exceção é uma indicação de um problema que ocorre durante a execução de um programa. O nome “exceção” significa que o problema não ocorre frequentemente”

(DEITEL; DEITEL, 2010, p. 336)

Introdução a exceções em POO

Em Java, as exceções são divididas em duas categorias (FURGERI, 2013):

Unchecked Exception: significa “exceção não verificada”. Neste tipo de exceção o Java não verifica o código-fonte para determinar se a exceção está sendo capturada. Por isso, o tratamento aqui é opcional. Fazem parte destas exceções de tratamento opcional, por exemplo, a verificação de acesso a um índice inexistente num vetor (trataremos de vetores na Unidade 4), a tentativa de se usar um método de um objeto ainda não instanciado e a conversão de um String em inteiro.

Checked Exception: significa “exceção verificada”. Neste tipo de exceção o compilador Java obriga o programador a tratá-la. O Java verifica o código-fonte com a finalidade de determinar se a exceção está sendo capturada.

Introdução a exceções em POO

Em Java, as exceções são divididas em duas categorias (FURGERI, 2013):

Unchecked Exception: exceção não verificada

- **Exemplo:** NullPointerException, NumberFormatException, ArrayIndexOutOfBoundsException, ...
- Derivada da classe **Error** ou **RuntimeException**

Checked Exception: exceção verificada

- Exceções previsíveis
- Devem ser tratadas pelo programa

Introdução a exceções em POO

Arnold, Gosling e Holmes (2007) observam que as **exceções verificadas** forçam o programador a pensar sobre o que fazer com erros nos locais em que podem ocorrer no código. O não tratamento de uma exceção verificada é notificado durante a compilação e não durante a execução.

O programador pode não querer, em algumas situações, fazer o controle sobre uma exceção. A linguagem Java permite a ele que um erro seja descartado. No entanto, é preciso que essa previsão seja informada na declaração do método.

Exceções

Falhas durante a execução de uma aplicação:

- Métodos:
  - problemas de estado interno;
  - Falha com objetos e dados manipulados;
  - Violação do contrato básico.
- Encontra algo inesperado:
  - Problema no hardware;
  - Array fora da faixa;
  - Valores de variáveis → divisão por zero
  - Parâmetro de métodos e falha de memória, etc...

Exceções

O que deve ser feito:

– Notificar o usuário

– Conseguir salvar todo o trabalho

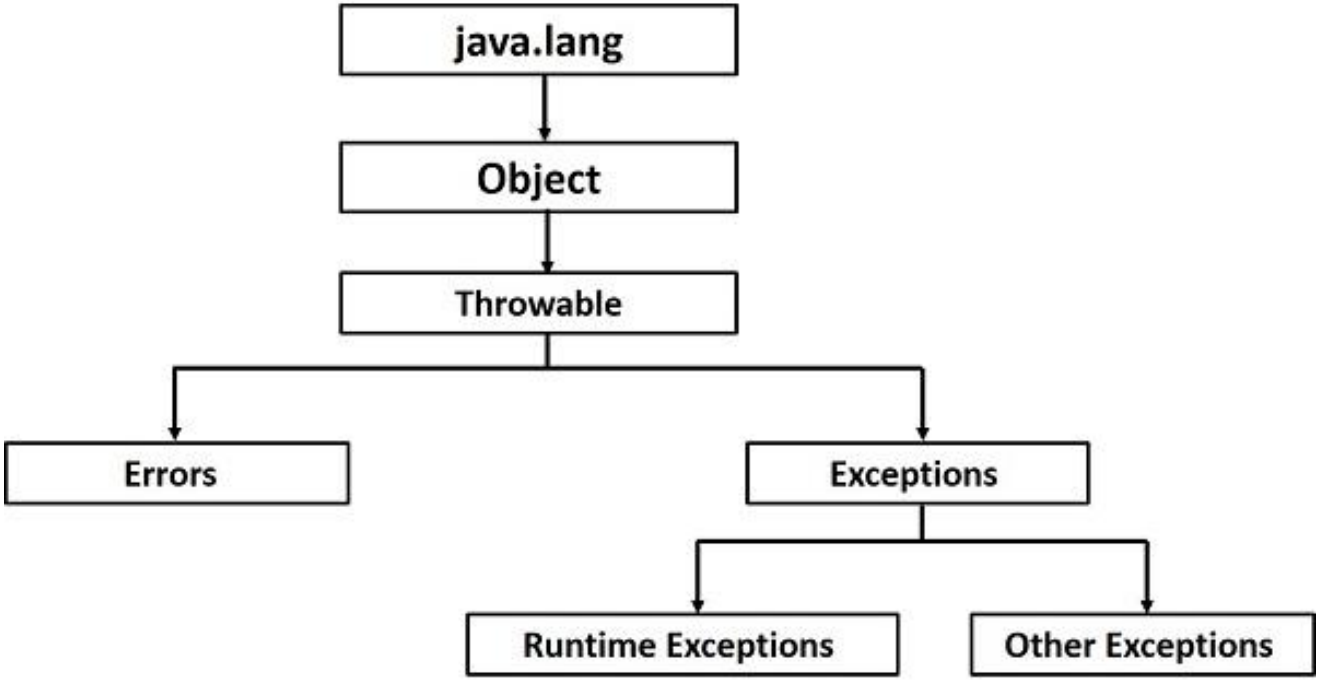
– Permitir a saída do sistema de forma normal

Exceções

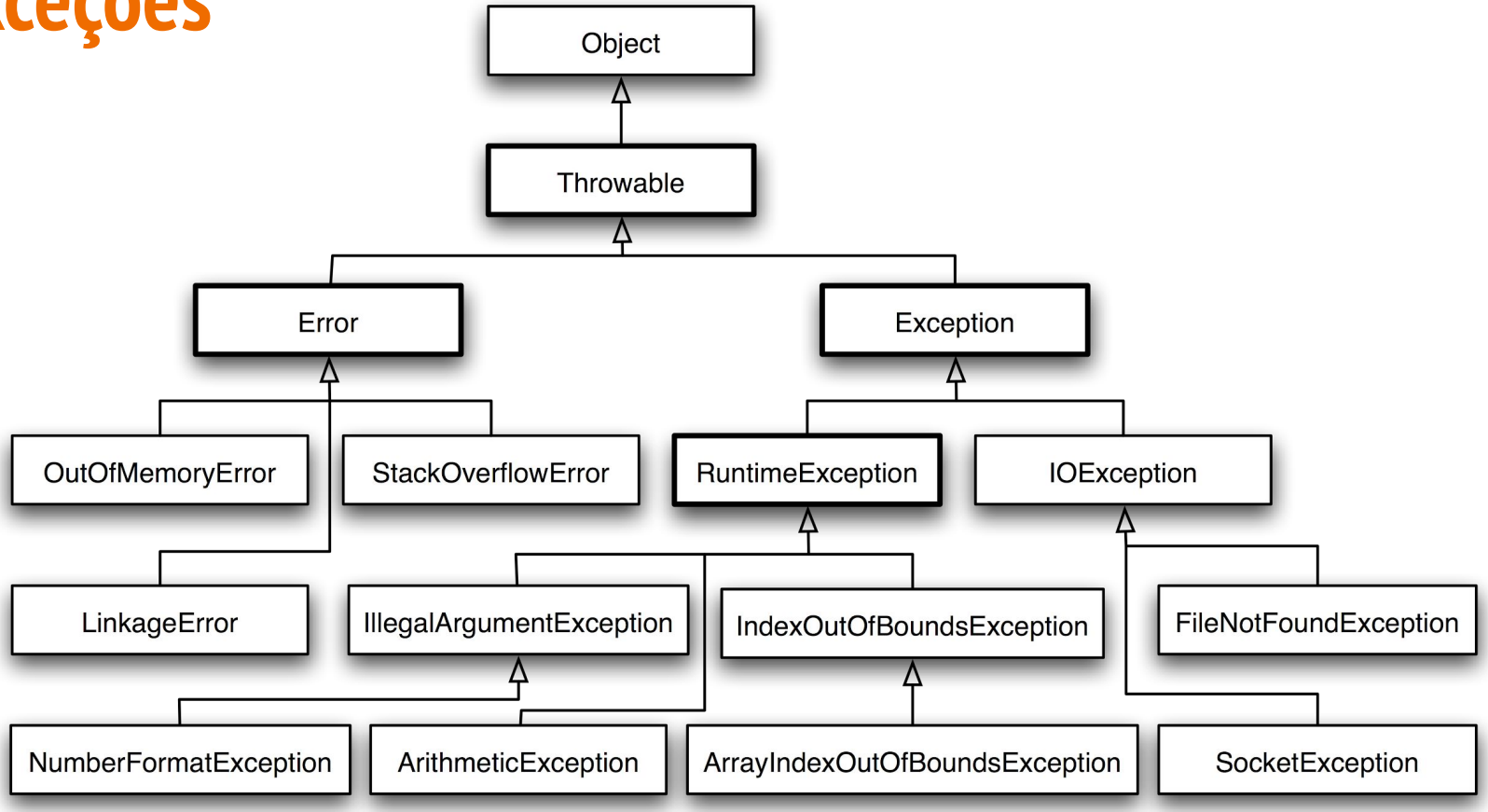
Quando um erro ocorre dentro de um método:

- Um objeto é criado com informações do erro, tipo e estado do programa.
- Ação de capturar esse objeto: throwing an exception.
- Ambiente de execução (runtime system) → tenta encontrar um tratamento.
- A busca segue a call stack – lista de chamadas de métodos.

Exceções


java.lang
Object
Throwable
Errors
Exceptions
Runtime Exceptions
Other Exceptions

Exceções


Object
Throwable
Error
Exception
OutOfMemoryError
StackOverflowError
RuntimeException
IOException
LinkageError
IllegalArgumentException
IndexOutOfBoundsException
FileNotFoundException
NumberFormatException
ArithmeticException
ArrayIndexOutOfBoundsException
SocketException

Exceções

Formas de captura:

- Através da cláusula **throws**, na assinatura do método
- Através do bloco **try-catch-finally**
- Ou da instrução **throw**.

Exceções

Formas de captura: cláusula **throws**

- Imagine uma situação em que não é desejado que uma exceção seja tratada na própria classe ou método, mas sim em outro que venha lhe chamar.
- Para solucionar tal situação utilizamos o comando throws na assinatura do método com a possível exceção que o mesmo poderá a vir lançar.
- Indica que um trecho de código que chame este método deve obrigatoriamente capturar uma possível exceção que ele lance e tratá-la ou não.

Exceções

- Formas de captura: cláusula **throws**
- Sintaxe de declaração de método com definição de exceções:

```
tipo_retorno nome_metodo() throws tipo_exceção_1, tipo_exceção_2, tipo_exceção_n {
        ...
        // Código
}
```

- Onde:
  - tipo_retorno - Tipo de retorno do método.
  - nome_metodo() - Nome do método que será utilizado.
  - tipo_exceção_1 a tipo_exceção_n - Tipo de exceções separadas por vírgula que o seu método pode vir a lançar.

Exceções

Formas de captura: bloco **try-catch-finally**

```
try {
    // executa o código
    // que pode disparar uma exceção
}
```

Tenta Executar

```
catch (Exception err) {
    // trata a exceção
    // err é uma referência para o objeto
    // da classe Exception
}
```

Captura o erro

```
finally {
    // esse bloco é sempre executado,
    // independente de ocorrer exceção
}
```

Sempre executa

Exceções

Pode haver mais de um bloco **catch**

```
try {
        <conjunto de instruções>
} catch (Nome da exceção) {
        <tratamento do erro>
} catch (Nome da exceção) {
        <tratamento do erro>
} catch (Nome da exceção) {
        <tratamento do erro>
} finally {
        <conjunto de instruções>
}
```

Exceções

Uso da instrução **throw**:

- É utilizada para disparar uma exceção.

- Ela pode forçar que uma determinada exceção ocorra

Exceções

Exemplo:

```
import java.util.Scanner;
public class InstrucaoThrow {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		float nota = 0f;
		try {
			System.out.println("Forneça uma nota entre 0 e 10.");
			nota = sc.nextFloat();
			if (nota < 0 || nota > 10) {
				throw new Exception("Fora da faixa permitida (0 a 10)!!!");
			}
		} catch (Exception e) {
			System.out.println(e.toString());
		}
	}
}
```

Exceções

Uso de getMessage() e printStacktrace()

- GetMessage(): serve para retornar a mensagem armazenada numa exceção qualquer.
- PrintStacktrace(): retorna o tipo de exceção gerado e informa em que linha da classe ocorreu o erro.

Exceções

Exemplo:

```
public class Exemplo2 {
	public static void main(String args[]) {
		int x = 10;
		try {
			int y = x / 0; // gera uma exceção
		}
		catch (Exception erro) {
			System.out.println(erro.getMessage());
			erro.printStackTrace();
		}
	}
}
```

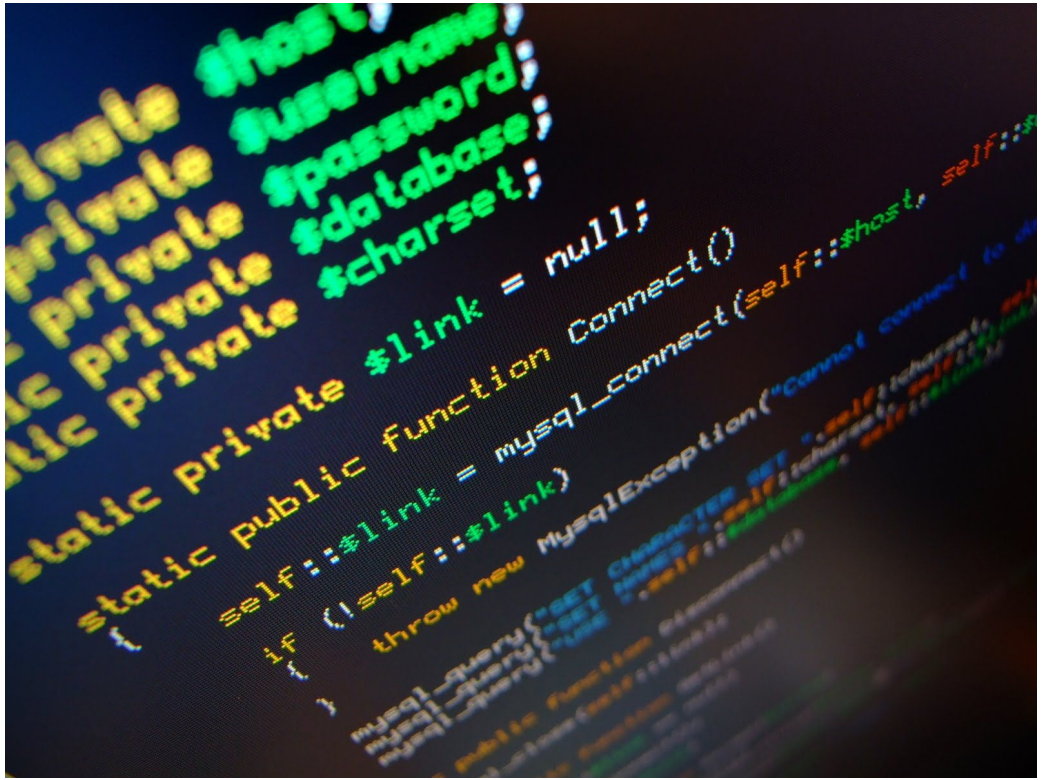
Exceções

De onde vêm as exceções?

Exceções são suspensas na classe Throwable, que faz parte do pacote java.lang. Eles estão no mesmo nível dos erros. Em outras palavras, Exceção e Erro são subclasses de Throwable.

Somente instâncias de objetos da classe Throwable podem ser lançadas pela instrução Java throw. Conforme declarado na documentação Java da classe Throwable, isso inclui uma captura instantânea da pilha de execução no momento da criação. Isso permite procurar a fonte da exceção (ou do erro), pois inclui o estado da memória do computador naquele momento. Um objeto jogável pode conter o motivo pelo qual foi construído. Isso é conhecido como recurso de exceção encadeada, porque um evento excepcional pode ser causado por uma certa cadeia de exceções.

Atividade prática - Exceções



Bibliografia

Cadenhead, Rogers; LEMAY, Laura, Aprenda em 21 dias Java 2. 4. ed. São Paulo: Campus, 2005.

DEITEL, Harvey H .; DEITEL, Paul J. Java: como Programar. 8. Ed. São Paulo: Pearson / Prentice-Hall, 2010.

Perguntas?

Obrigado!