



#Reskilling4Employment
Software Developer

Programação Estruturada

Exceções

Vitor Santos



Conteúdo



- Definição de Exceção
- Tratamento de Exceções
- Criar Exceções
- Lançar Exceções

Exceções



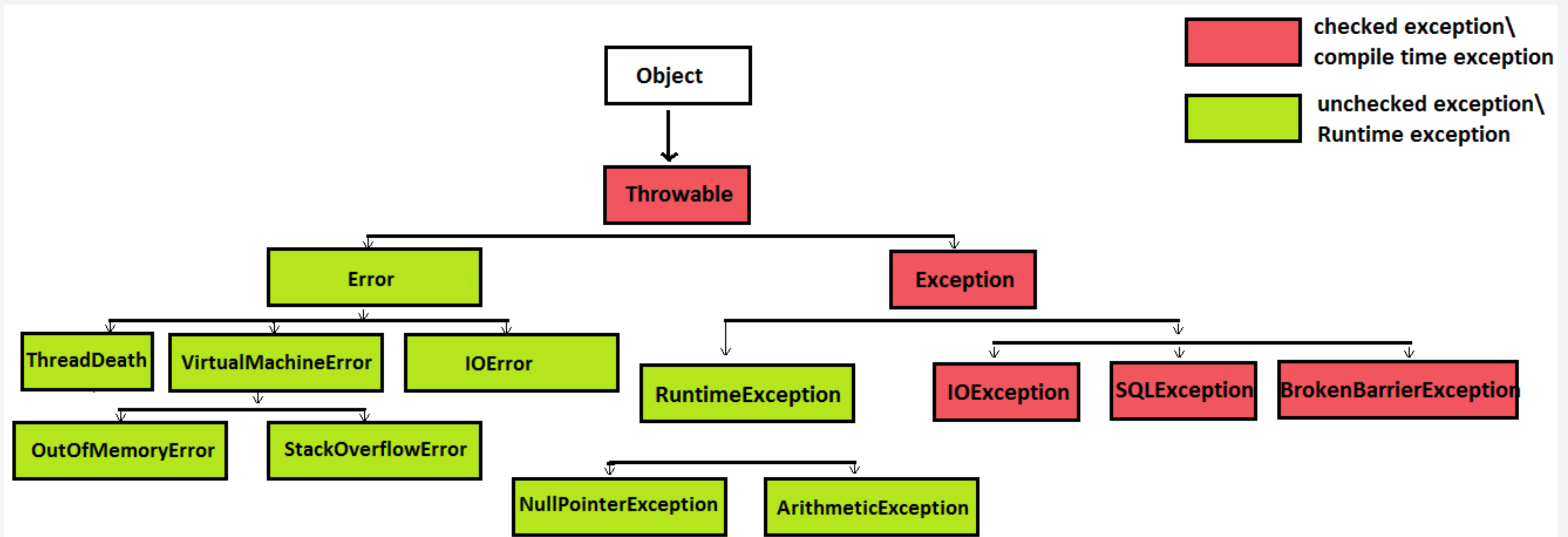
- Uma exceção é um evento que ocorre durante a execução de um programa que interrompe o ciclo normal das instruções.
- Ao usar o subsistema de tratamento de exceções do Java podemos, de uma forma estruturada e controlada, tratar os erros em tempo de execução.

Exceções



- Em Java as exceções são representadas por classes.
- Quando ocorre uma exceção num programa é criado um objecto de uma determinada classe.
- Existem três tipos de exceções:
 - Checked Exception
 - Error
 - Runtime Exception

Hierarquia de Exceções em Java



Tratamento de Exceções

- No core do tratamento de exceções está o **try** e o **catch**.
- Estas palavras reservadas são utilizadas em conjunto não podendo ser usada uma sem a outra.
- De seguida é apresentada a forma geral do try-catch.

Tratamento de Exceções

```
try {  
    ...  
} catch ( Tipo_de_exceção1 var1 ) {  
    ...  
} catch ( Tipo_de_exceção2 var2 ) {  
    ...  
} finally {  
    ...  
}
```

1

2

3

4

1.Código normal do programa, onde podem ser detectadas exceções

2.Código de tratamento da exceção do tipo declarado; var1 é a instância da exceção, que pode ser usada aqui.

3.Código de tratamento de outra exceção, do tipo declarado; var2 é a instância desta exceção, que pode ser usada aqui.

4.Código sempre executado, quer ocorra uma exceção ou não no bloco try.

Tratamento de Exceções

- Há exceção:
 - se houver catch local apropriado, este é executado e só depois o bloco finally;
 - se não houver catch, é executado o bloco finally e depois o catch externo, se existir, por exemplo no método invocador.
- Não há exceção: controle de execução transferido do bloco try quando termina sequência de instruções ou quando é executada uma instrução **return**, **break** ou **continue**.

Exception	Meaning
ArithmeticException	Arithmetic error, such as divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.
ArrayStoreException	Assignment to an array element of an incompatible type.
ClassCastException	Invalid cast.
IllegalArgumentException	Illegal argument used to invoke a method.
IllegalMonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException	Environment or application is in incorrect state.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
SecurityException	Attempt to violate security.
StringIndexOutOfBoundsException	Attempt to index outside the bounds of a string.
TypeNotPresentException	Type not found. (Added by J2SE 5.)
UnsupportedOperationException	An unsupported operation was encountered.

ClassNotFoundException	Class not found.
CloneNotSupportedException	Attempt to clone an object that does not implement the Cloneable interface.
IllegalAccessException	Access to a class is denied.
InstantiationException	Attempt to create an object of an abstract class or interface.
InterruptedException	One thread has been interrupted by another thread.
NoSuchFieldException	A requested field does not exist.
NoSuchMethodException	A requested method does not exist.

Exemplos de Exceções

```
public class ExException0 {  
    public static void main(String args[]) {  
        int numero[ ] = new int[4];  
        System.out.println("Antes da exceção ser gerada.");  
        // gerar uma exceção index out of bounds  
        numero[7] = 10;  
    }  
}
```

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 7
at ExException0.main(ExException0.java:9)

```
public class ExException1 {  
    public static void main(String args[ ]) {  
        int numeros[ ] = new int[4];  
        try {  
            System.out.println("Antes da excepcao ser gerada.");  
            //Gerar uma excepção index out-of-bounds.  
            numeros[7] = 10;  
            System.out.println("esta mensagem não será apresentada");  
        } catch (ArrayIndexOutOfBoundsException exc) {  
            // apanhar a excepção  
            System.out.println("Index for dos limites!!");  
        }  
        System.out.println("Depois do catch.");  
    }  
}
```

```
public class ExException2 {  
    public static void main(String args[]) {  
        int numero[ ] = { 4, 8, 16, 32, 64, 128 };  
        int denominador[ ] = { 2, 0, 4, 4, 0, 8 };  
        for(int i=0; i<numero.length; i++) {  
            try {  
                System.out.println(numero[i] + " / " +  
                    denominador[i] + " é " + numero[i]/denominador[i]);  
            }  
            catch (ArithmeticException exc) {  
                // apanhar a excepção  
                System.out.println("Nao se pode dividir por zero!");  
            }  
        }  
    }  
}
```

```
public class ExException3 {  
    public static void main(String args[]) {  
        int numero[] = { 4, 8, 16, 32, 64, 128, 256, 512 };  
        int denominador[] = { 2, 0, 4, 4, 0, 8 };  
        for(int i=0; i<numero.length; i++) {  
            try {  
                System.out.println(numero[i] + " / " + denominador[i] + " é " +  
                    numero[i] / denominador[i]);  
            }  
            catch (ArithmeticException exc) {  
                System.out.println("Não é possível div. por" + "zero!");  
            }  
            catch (ArrayIndexOutOfBoundsException exc) {  
                System.out.println("Não foi encontrado o" +  
                    "elemento correspondente.");  
            }  
        }  
    }  
}
```

...

Criar uma Exceção

- O mecanismo de exceções em Java, permite que o programador construa as suas próprias classes de exceção, de forma muito simples, por herança da superclasse das exceções, a classe **Exception**.

Criar uma Exceção

- O código necessário para criar uma classe de exceções toma a seguinte forma:

```
class MinhaExcepcao extends Exception {  
    MinhaExcepcao() {  
        super();  
    }  
  
    MinhaExcepcao(String s)  
    {  
        super(s);  
    }  
}
```


Lançar uma Exceção

- Quando existe uma situação de erro, por exemplo ao retirar um elemento da coleção quando ele não existe, o programador pode explicitamente sinalizar o envio de um erro que deve ser posteriormente tratado.

Lançar uma Exceção

- Para tal apenas é necessário criar uma instância da exceção pretendida, da seguinte forma:

...

```
throw new <nomeClasseExcepcao>();  
//construtor vazio
```

...

...

```
throw new <nomeClasseExcepcao>("mensagem que" +  
"queira enviar");  
//construtor parametrizado por uma String
```

Criar uma Exceção

- A única forma que um programador tem de saber que um determinado método pode lançar uma exceção, é estar ao explicitamente reflectido na assinatura do método.
- Para tal a assinatura do método deve ser feita de forma a indicar que o método pode “libertar alguma” exceção que deve ser feito de acordo com a seguinte sintaxe:

```
public <tipoDados><nomeMetodo>(<listaParametros>)  
throws <listaClassesExcepcao>
```

- Um método que invoque outro que pode lançar uma exceção, deve precaver-se e ter disponível código de tratamento para essa exceção. Se não o fizer, então esse método deve explicitamente (na sua cláusula de throws) declarar que também pode libertar uma exceção desse tipo.



#Reskilling4Employment
Software Developer

Programação Estruturada

Exceções

Vitor Santos

