PARADIGMAS DE PROGRAMAÇÃO

2023/2024

P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Aula 11

1. Palavras Reservadas
2. O que é uma Excepção?
3. Tratamento de Excepções
4. Criar uma Excepção
5. Lançar uma Excepção
6. Palavras Reservadas Usadas
7. Links Úteis

Ricardo Santos | rjs@estg.ipp.pt

Palavras Reservadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

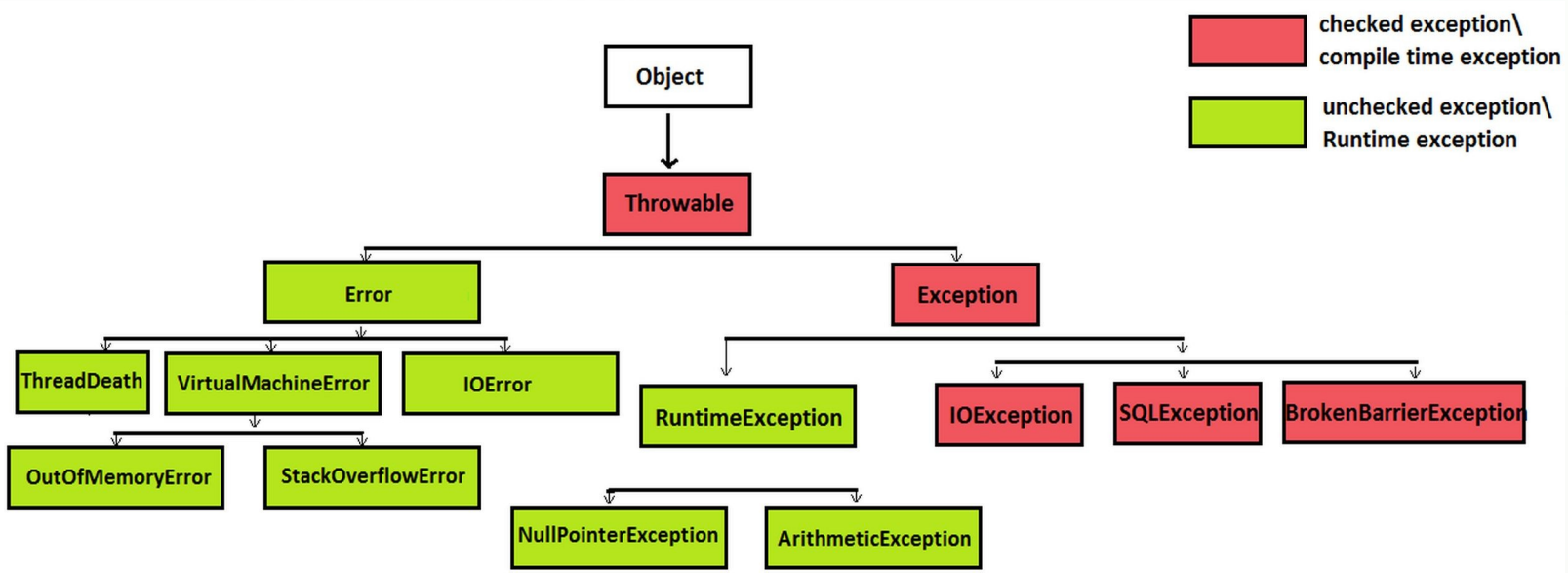
* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

O que é uma Excepção?

- Uma excepção 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 excepções do Java podemos, de uma forma estruturada e controlada, tratar os erros em tempo de execução

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


checked exception\
compile time exception
unchecked exception\
Runtime exception
Object
Throwable
Error
Exception
ThreadDeath
VirtualMachineError
IOError
RuntimeException
IOException
SQLException
BrokenBarrierException
OutOfMemoryError
StackOverflowError
NullPointerException
ArithmeticException

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Tratamento de Excepções

- No ***core*** do tratamento de excepções 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`:

```
try {
        ...                    1
} catch(Tipo_de_excepção1 var1){
        ...                    2
} catch(Tipo_de_excepção2 var2){
        ...                    3
} finally {
    ...                        4
}
```

1. Código normal do programa, onde podem ser detectadas excepções
2. Código de tratamento da excepção do tipo declarado; `var1` é a instância da excepção, que pode ser usada aqui
3. Código de tratamento de outra excepção, do tipo declarado; `var2` é a instância desta excepção, que pode ser usada aqui

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

   Há excepção: - 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 excepção: - controlo 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.
StringIndexOutOfBounds	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.

```
public class ExException0
{
   public static void main(String args[]) {
      int numero[] = new int[4];
      System.out.println("Antes da excepção ser” +
           “gerada.");
      // gera uma excepção 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 nao sera" +
               "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.");
        }
      }
   }
}
```

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

Criar uma Excepção

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

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

```
class MinhaExcepcao extends Exception {

   MinhaExcepcao() {
      super();
   }

   MinhaExcepcao(String s)
   {
      super(s);
   }
}
```

Lançar uma Excepção

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

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

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

- A única forma que um programador tem de saber que um determinado método pode lançar uma excepção, 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**" excepção 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 excepção, deve precaver-se e ter disponível código de tratamento para essa excepção. Se não o fizer, então esse método deve explicitamente (na sua cláusula de `throws`) declarar que também pode libertar uma excepção desse tipo

Palavras Reservadas Usadas

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Links Úteis

- http://docs.oracle.com/javase/tutorial/essential/exceptions/index.html