

## **6 – Exceções**

- Quando um programa viola as restrições semânticas da linguagem, a JVM assinala um erro ao programa, sob a forma de exceção.

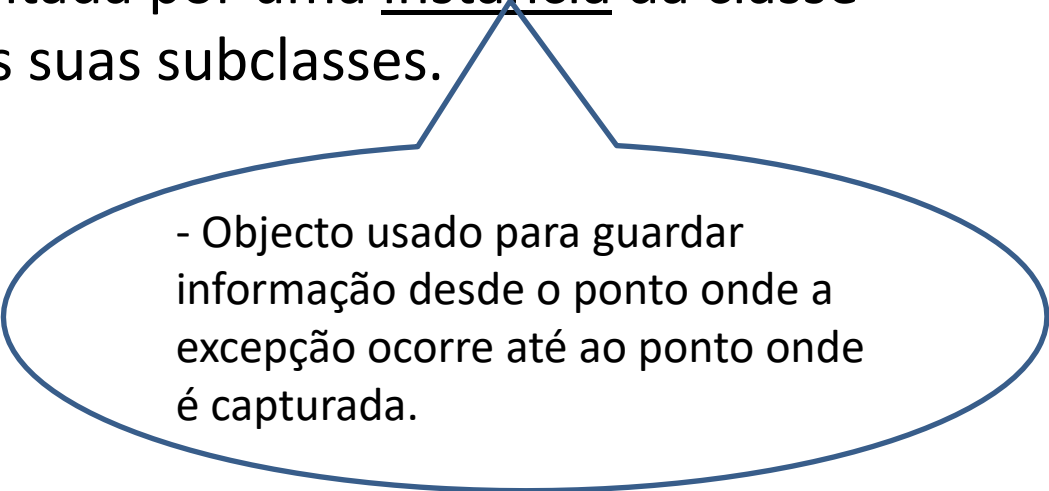
*Uma exceção é um erro recuperável*

- O controlo da execução do programa é transferido do ponto onde ocorre a exceção para um ponto que pode ser especificado pelo utilizador.

*? Sintaxe versus Semântica ?*

## 6 – Exceções

- Uma exceção diz-se lançada (**thrown**) no ponto onde ocorre e diz-se capturada (**caught**) no ponto para onde o controlo de execução é transferido
- Cada exceção é representada por uma instância da classe Throwable ou de uma das suas subclasses.



- Objecto usado para guardar informação desde o ponto onde a excepção ocorre até ao ponto onde é capturada.

## *Programação Orientada a Objectos - P. Prata, P. Fazendeiro*

O tratamento de uma excepção é definido pela cláusula **catch** de uma instrução **try**:

```
try { // bloco de instruções, nas quais queremos ter a  
// possibilidade de detectar a ocorrência de erros recuperáveis.  
  
} catch ( <classe da excepção> <instância da excepção gerada> ) {  
// instruções para tratamento da excepção considerada  
  
[ ... outras cláusulas catch ]*  
  
} [ finally {  
// instruções que serão executadas, ocorra ou não uma  
exceção no bloco try  
} ]*
```

\*[ **opcional** ]

## 6 – Exceções

Uma excepção pode ser lançada porque:

- A JVM detecta uma violação da semântica da linguagem

Exemplos - divisão por zero

- o limite de um dado recurso foi ultrapassado

- Foi executada uma instrução **throw**

*throw – geração explícita de uma excepção definida ou não pelo utilizador*

**throw new** < construtor da classe Exception ou de uma sua subclasse >

## **6 – Exceções**

Existem dois tipos de exceções:

### **A - Exceções verificáveis pelo compilador**

*- O compilador verifica se o programa trata as exceções que poderão ocorrer no código.*

Para cada exceção verificável, o método onde essa exceção pode ocorrer deve:

- prever o tratamento da exceção (try – catch)

ou

## 6 – Exceções

### A - Excepções verificáveis pelo compilador (cont)

...

ou

lançar a exceção, através da cláusula throws, para que seja tratada no método invocador ou noutro mais externo.

Ex.los

```
public void metodoExemplo () throws <nome da classe da exceção>  
{ ...
```

```
public static void main (String [] args ) throws IOException { ...
```

## **6 – Exceções**

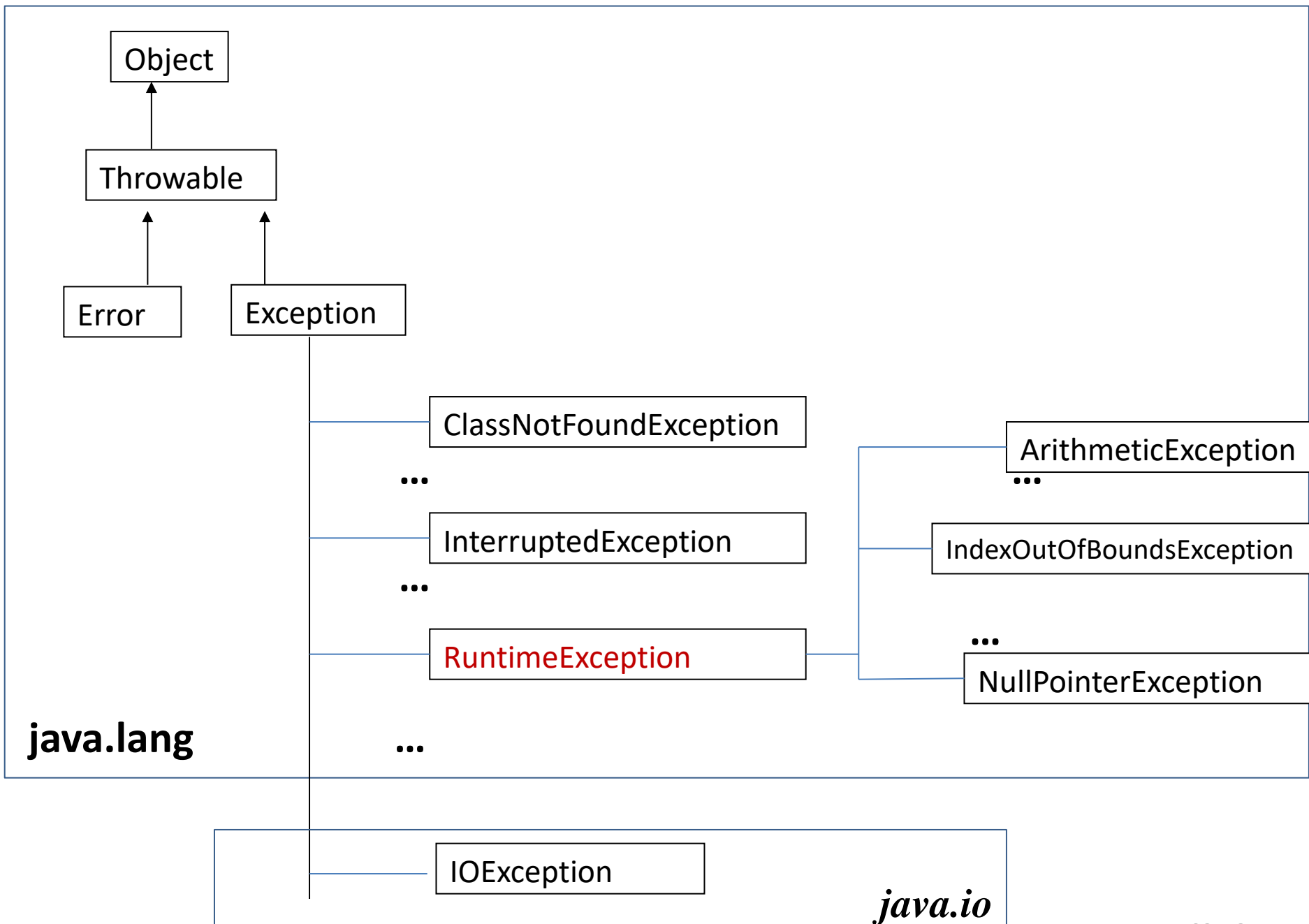
### **B - Exceções não verificáveis pelo compilador**

*- São objetos das classes **RuntimeException**, **Error** e respectivas subclasses*

RuntimeException – exceções cuja ocorrência é difícil de ser verificável pelo programador

Error – erros não recuperáveis

Hierarquia das classes de excepção em Java: →





## 6 – Exceções

### Exemplo 1: Definir uma exceção

Subclasse de Exception

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

```

public class Teste1 {
    public static void explode () throws Bomba {
        //throws - lança a exceção para ser tratada num método que invocar o método explode()
        throw new Bomba(); // gera a exceção Bomba
    }
    public static void main (String []args) {
        try {
            explode();
        }
        catch ( RuntimeException e ) {
            System.out.println ("Runtime Explode"+ e.getMessage());
        }
        catch ( Bomba b ) {
            System.out.println ( "Bomba" );
        }
    }
} // main
} // Teste1

```

A exceção que ocorre não é compatível com a classe RuntimeException mas é compatível com a classe Bomba.

## 6 – Exceções

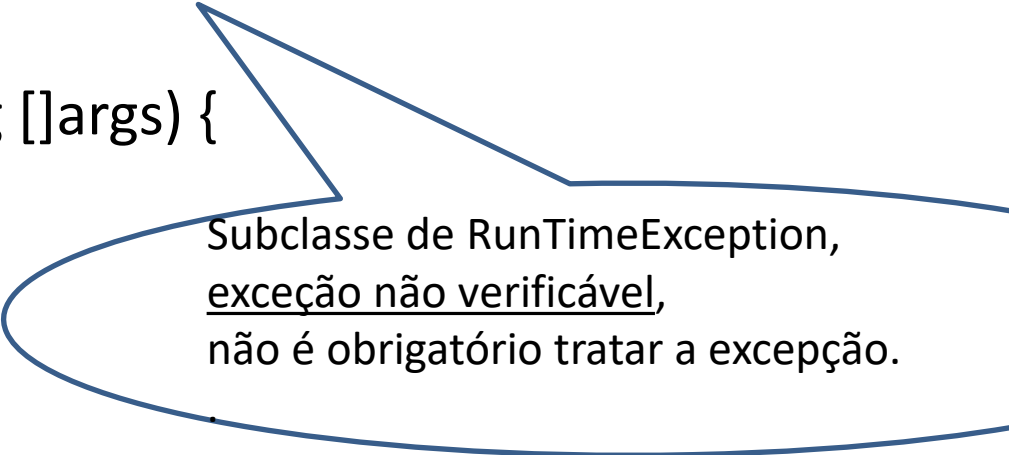
Output do programa:



Bomba

## Exemplo 2: Cláusula finally

```
public class Teste2 {  
    public static void explode () {  
        throw new NullPointerException();  
    }  
    public static void main (String []args) {  
        try {  
            explode();  
        }  
        catch ( Bomba e ) {  
            System.out.println ("Bomba");  
        }  
        finally {  
            System.out.println ("Exceção não capturada");  
        }  
    }  
} // main  
} // Teste2
```



Subclasse de RuntimeException,  
exceção não verificável,  
não é obrigatório tratar a exceção.

## 6 – Exceções

Output do programa:

*Exceção não capturada*

```
java.lang.NullPointerException  
at Teste2.Explode (Teste2.java:7)  
at Teste.main ( Teste2.java:11)
```

## 6 – Exceções

### Herança e cláusula throws

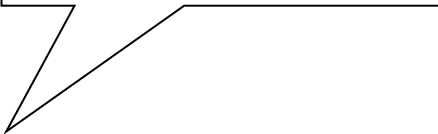
- Um método que sobrepõe (“overrides”) outro não pode declarar lançar mais exceções do que o método que é sobreposto.

#### Exemplo:

```
public class C1 {  
    public void m2() throws Exception {  
        System.out.println (“Método 2 da classe C1 ”);  
    }  
}
```

## 6 – Exceções

```
public class C2 extends C1 {  
    public void m2()* throws InterruptedException,  
                                                ClassNotFoundException{  
  
        System.out.println (“Método 2 da classe C2 ”);  
    }  
}
```



Válido

*\* O método m2 da classe C2 sobrepõe m2 da classe C1*

## **6 – Exceções**

### **Herança e cláusula throws (cont ...)**

- Cada exceção declarada em m2 da classe C2 tem que ser do mesmo tipo (classe) de uma exceção lançada em m2 de C1 ou de um seu subtipo (subclasse).
  
- No método m2 da classe C1 tem que ser lançada a mesma exceção que em m2 de C2, ou uma exceção de uma sua superclasse.



## 6 – Exceções

**Exemplo 3: Qual o output do seguinte programa?**

```
package excecoes;
```

```
public class TesteException extends Exception {
```

```
    public TesteException (){
```

```
        super();
```

```
    }
```

```
    public TesteException ( String s ){
```

```
        super ( s );
```

```
    }
```

```
}
```

```
public class Teste {  
    public static int atirador (String s) throws TesteException {  
        if ( s.equals ("dividir") ) {  
            int i= 0 ;  
            return i/i;  
        }  
        if ( s.equals( "null" ) ) {  
            s = null;  
            return s.length();  
        }  
        if ( s.equals ("teste") ) {  
            throw new TesteException ("Teste");  
        }  
        return 0;  
    }  
} // fim do atirador
```

TesteException é uma  
exceção verificável

cria uma instância  
de TesteException

...

```
public static void main (String []args) {  
    String [] txt = new String [4];  
    txt [0] = "dividir" ;  
    txt [1] = "null" ;  
    txt [2] = "não" ;  
    txt [3] = "teste" ;  
}
```



```
public static void main (String []args) {
```

```
...
```

```
for ( int i = 0; i < txt.length ; i++ ){
```

```
    try {
```

```
        atirador (txt [i] );
```

```
        System.out.println ( "Teste(" + txt[i] + ")" + " não lançou uma  
                                exceção");
```

```
    }
```

```
    catch (Exception e){
```

```
        System.out.println ( "Teste(" + txt[i] + ") lançou uma " +  
            e.getClass() + "\n com a mensagem " + e.getMessage() );
```

```
    }
```

```
finally {
```

```
    System.out.println ("****");
```

```
    System.out.println ("[Atirador(" + txt[i] + ") executado]");
```

```
}
```

```
} // main
```

```
} // Teste
```

Sempre executado,  
quer ocorra uma  
exceção quer não.

## 6 – Exceções

### Output:

```
Teste(dividir) lançou uma class java.lang.ArithmeticException  
com a mensagem / by zero
```

```
****
```

```
[Atirador(dividir ) executado]
```

```
Teste(null) lançou uma class java.lang.NullPointerException  
com a mensagem null
```

```
****
```

```
[Atirador(null ) executado]
```

```
Teste(não) não lançou uma exceção
```

```
****
```

```
[Atirador(não ) executado]
```

```
Teste(teste) lançou uma class excepcoes.TesteException  
com a mensagem Teste
```

```
****
```

```
[Atirador(teste ) executado]
```

## 6 – Exceções – Exercícios

Suponha a classe Exemplo:

```
import java.util.ArrayList;

public class Exemplo {
    private int dim; // dimensão do array
    private String[] listaEstatica;
    private ArrayList<String> listaDinamica;

    public Exemplo (int d){
        dim = d;
        listaEstatica = new String [dim];
        listaDinamica = new ArrayList<String>();
    }
}
```

## 6 – Exceções – Exercícios

Suponha a classe Exemplo: ...

```
public int getDim() {    return dim; }
public void setDim(int dim) {
    this.dim = dim;
}
public String[] getListaEstatica() {
    return listaEstatica;
}
public void setListaEstatica(String[] listaEstatica) {
    for (int i = 0; i < listaEstatica.length; i++) {
        this.listaEstatica[i] = listaEstatica[i];
    }
}
```

## 6 – Exceções – Exercícios

**Suponha a classe Exemplo: ...**

```
public ArrayList<String> getListaDinamica() {  
    return listaDinamica;  
}  
  
public void setListaDinamica (ArrayList<String> listaDinamica) {  
    this.listaDinamica = (ArrayList<String>)listaDinamica.clone();  
}  
  
public String toString() {  
    String s= "dim= " + dim + ", listaEstatica=";  
    for (int i = 0; i < listaEstatica.length; i++) {  
        s = s + listaEstatica[i] + " ";  
    }  
    s = s + ", listaDinamica=" + listaDinamica ;  
    return s;  
} }
```



## 6 – Exceções – Exercícios

Qual o output do programa abaixo? (1)

```
public class TesteExemplo {  
  
    public static void main(String[] args) {  
        Exemplo e1, e2;  
  
        e1 = new Exemplo (5);  
  
        System.out.println(e1);  
    }  
}
```

## **6 – Exceções – Exercícios**

- a) Para a classe Exemplo criar um método que devolva o primeiro elemento da listaDinamica.  
Se a lista estiver vazia deverá gerar a exceção ListaVaziaException, com a mensagem de erro “Erro: lista vazia”.  
A exceção deverá ser tratada no programa que invocar o método.
- Criar a classe de exceção ListaVaziaException.
  - Testar o método.

## 6 – Exceções – Exercícios

```
public String primeiroLD () throws ListaVaziaException{  
    if (listaDinamica.isEmpty() )  
        throw new ListaVaziaException( "Erro: Lista vazia" );  
    else  
        return listaDinamica.get(0);  
}
```

\_\_\_ Classe da Exceção // criar num ficheiro com o nome da classe

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

## **6 – Exceções – Exercícios**

### **Testar**

```
public static void main(String[] args) {  
    Exemplo e1, e2;  
    e1 = new Exemplo(5);  
    System.out.println(e1);  
  
    try {  
        String s = e1.primeiroLD();  
        System.out.println(s);  
    } catch (ListaVaziaException x) {  
        System.out.println(x.getMessage());  
    }  
}
```

**Qual o output? (2)**

## **6 – Exceções – Exercícios ... Testar**

```
public static void main(String[] args) {
```

```
    Exemplo e1, e2;
```

```
    e1 = new Exemplo(5);
```

```
    System.out.println(e1);
```

```
    ArrayList<String> l = new ArrayList<String>();
```

```
    l.add("XPTO1");
```

```
    l.add("XPTO2");
```

```
    e1.setListaDinamica(l);
```

```
    try {
```

```
        String s = e1.primeiroLD();
```

```
        System.out.println(e1);
```

```
        System.out.println(s);
```

```
    } catch (ListaVaziaException x) {
```

```
        System.out.println(x.getMessage());
```

```
    }
```

**Qual o output? (3)**

## 6 – Exceções – Exercícios

**Output:**

**(1)**

**dim= 5, listaEstatica=null null null null null , listaDinamica=[]**

**(2)**

**dim= 5, listaEstatica=null null null null null , listaDinamica=[]**

**Erro: Lista vazia**

**(3)**

**dim= 5, listaEstatica=null null null null null , listaDinamica=[]**

**dim= 5, listaEstatica=null null null null null , listaDinamica=[XPTO1,  
XPTO2]**

**XPTO1**

## **6 – Exceções – Exercícios**

b) Para a mesma classe criar um método que atribua um valor a uma dada posição da listaEstatica.

Se a posição for inválida o método deverá gerar a exceção `ArrayIndexOutOfBoundsException` (\*) com a mensagem de erro “Posição inválida”.

- Testar o método

**(\*) exceção não verificável do package `java.lang`**

## 6 – Exceções – Exercícios

### Classe Exemplo ...

```
public void adicionarLE (String valor, int pos) /* !!! */ {  
  
    if (pos < 0 || pos >= dim)  
  
        throw new ArrayIndexOutOfBoundsException("posição  
                                                    inválida");  
  
    listaEstatica[pos] = valor;  
}
```



## **6 – Exceções – Exercícios**

## **Testar ...**

```
public static void main(String[] args) {  
    Exemplo e1, e2;  
    e1 = new Exemplo(5);  
    System.out.println(e1);  
  
    try {  
        e1.adicionarLE("ABC", 3);  
    } catch (ArrayIndexOutOfBoundsException x) {  
        System.out.println(x.getMessage());  
    }  
    System.out.println(e1);  
  
    try {  
        e1.adicionarLE("XYZ", 10);  
    } catch (ArrayIndexOutOfBoundsException x) {  
        System.out.println(x.getMessage());  
    }  
    System.out.println(e1);  
}
```

**6 – Exceções – Exercícios**

**Testar ...**

**Output ...**

dim= 5, listaEstatica=null null null null null , listaDinamica=[]

dim= 5, listaEstatica=null null null ABC null , listaDinamica=[]

**posição inválida**

**dim= 5, listaEstatica=null null null ABC null , listaDinamica=[]**

## **6 – Exceções – Exercícios**

**Qual o output dos seguintes blocos de código?**

**a)**

```
for (int i =0; i < 5 ; i++) {  
    try {  
        System.out.println ( 10/i );  
    } catch (ArithmeticException x){  
        System.out.println (“Erro na iteração: ” + i);  
    }  
    finally {  
        System.out.println ( “Continua” );  
    }  
}
```

## **6 – Exceções – Exercícios**

**Qual o output dos seguintes blocos de código?**

**b)**

```
int i = 0;
try {
    for ( i = -1; i < 5 ; i++){
        System.out.println ( 10 / (i-1) );
    }
} catch (ArithmeticException x){
    System.out.println ("Erro na iteração: " + i);
}
finally {
    System.out.println ( "Continua" );
}
```