

PG II

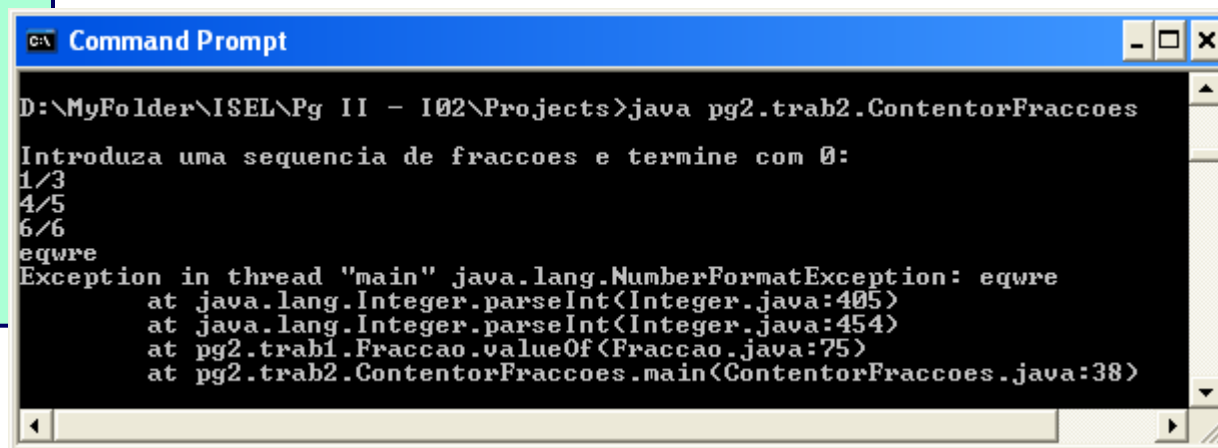
Programação Orientada aos Objectos em Java

Excepções:

- Definição de Excepções;
- Mecanismo de tratamento de Excepções;
- Implementação de novas Excepções;
- Hierarquia de Excepções;
- Derivar de `Exception` <> `RuntimeException`;
- Recuperação da execução de programas;
- `IOException` (construtor `In(String filename)`).

O que é uma excepção?

```
Fracao fr;  
String aux;  
  
while (!IO.cin.eof()){  
    aux = IO.cin.readLine();  
    fr = Fracao.valueOf(aux);  
    cnt.add(fr);  
    if (cnt.isFull()) break;  
}
```



```
C:\ Command Prompt  
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.trab2.ContentorFraccoes  
Introduza uma sequencia de fraccoes e termine com 0:  
1/3  
4/5  
6/6  
eqwre  
Exception in thread "main" java.lang.NumberFormatException: eqwre  
    at java.lang.Integer.parseInt(Integer.java:405)  
    at java.lang.Integer.parseInt(Integer.java:454)  
    at pg2.trab1.Fracao.valueOf(Fracao.java:75)  
    at pg2.trab2.ContentorFraccoes.main(ContentorFraccoes.java:38)
```

- Uma excepção é um acontecimento que interrompe o fluxo normal de execução de um programa:
 - Erro de leitura;
 - Indexação fora dos limites do *array*;
 - Falta de memória (a referência fica com **null**);
 - Divisão inteira por zero
 - etc...
- **Objectivo:** Tornar os programas suficientemente robustos para que sigam o seu fluxo normal de execução.

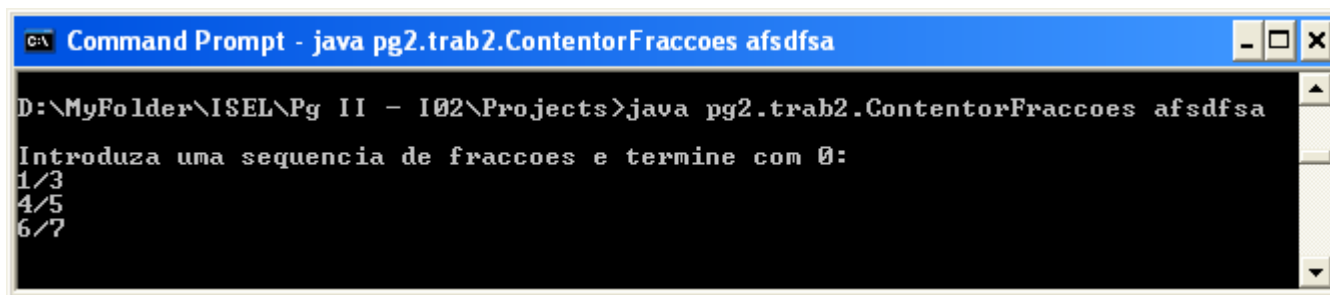
Controlo de Excepções com "if's"

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

        AbstractContainer cnt = null;

        if (args.length > 0)
            if (isInteger(args[0])) {
                int dim = Integer.parseInt(args[0]);
                cnt = new StaticContainer(dim);
            }
        else
            cnt = new DynContainer();

        ...
    }
}
```



```
C:\ Command Prompt - java pg2.trab2.ContentorFraccoes afsdfsa
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.trab2.ContentorFraccoes afsdfsa
Introduza uma sequencia de fraccoes e termine com 0:
1/3
4/5
6/7
```

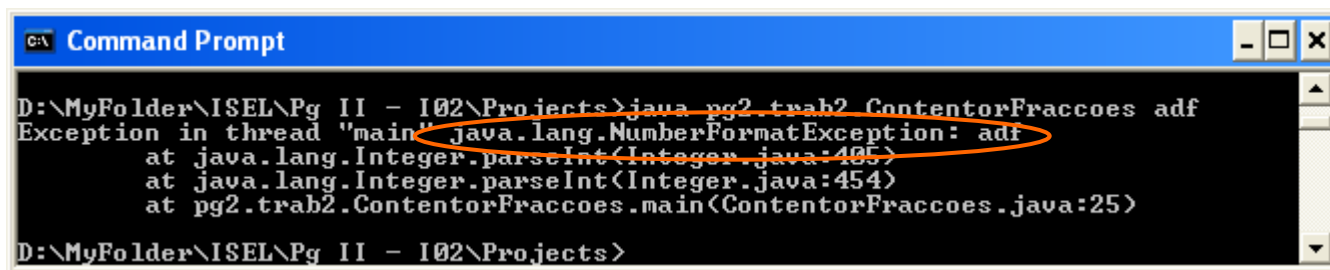
Controlo de Excepções

Sem a verificação feita anteriormente pela função "*isInteger*" teríamos a seguinte excepção:

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

        AbstractContainer cnt = null;

        if (args.length > 0)
            /*if (isInteger(args[0]))*/ {
                int dim = Integer.parseInt(args[0]);
                cnt = new StaticContainer(dim);
            }
        if (cnt == null)
            cnt = new DynContainer();
        ...
    }
}
```



```
C:\ Command Prompt
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.trab2.ContentorFraccoes adf
Exception in thread "main" java.lang.NumberFormatException: adf
    at java.lang.Integer.parseInt(Integer.java:405)
    at java.lang.Integer.parseInt(Integer.java:454)
    at pg2.trab2.ContentorFraccoes.main(ContentorFraccoes.java:25)
D:\MyFolder\ISEL\Pg II - I02\Projects>
```

Tratamento de Excepções

As **excepções** ao contrário dos **erros**, podem ser capturadas e processadas por forma a que o programa siga o seu fluxo normal de execução.

Assim, uma outra forma de resolver o problema anterior seria:

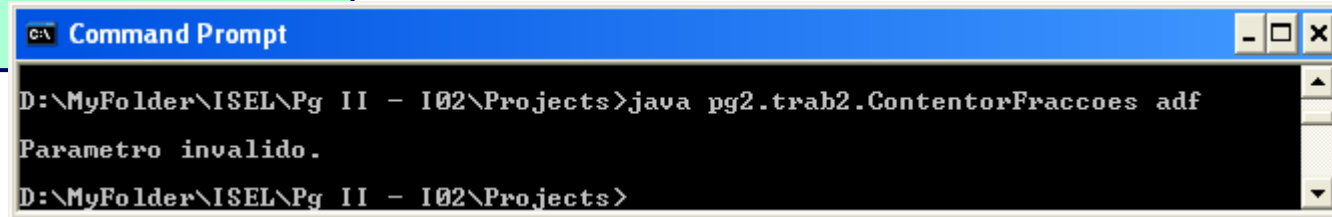
```
public class ContentorFraccoes {
    ...
    public static void main (String args[]){

        AbstractContainer cnt = null;

        if (args.length > 0)
            try {
                int dim = Integer.parseInt(args[0]);
                cnt = new StaticContainer(dim);
            }
            catch (NumberFormatException e){
                IO.cout.writeln("\nParametro invalido.");
                System.exit(1);
            }
        else
            cnt = new DynContainer();
    }
    ...
}
```

Vantagens face à programação por if's:

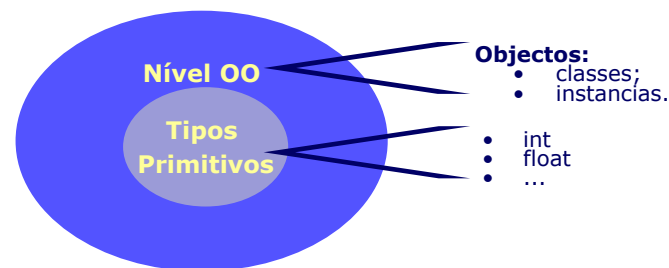
- Separação entre o tratamento de erros e o algoritmo;
- Propagação dos erros através do *stack de calls*;
- Divisão por grupos de erros organizados em hierarquia.



```
C:\> Command Prompt
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.trah2.ContentorFraccoes adf
Parametro invalido.
D:\MyFolder\ISEL\Pg II - I02\Projects>
```

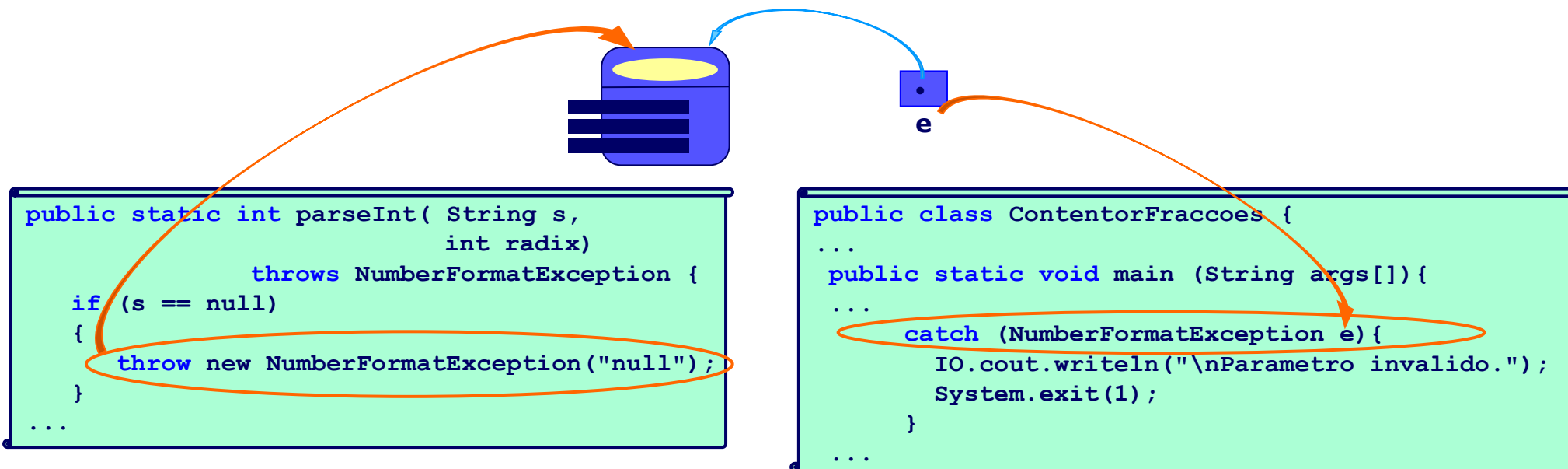
Como são definidas as Excepções em Java?

Já sabemos que em Java existem dois níveis de entidades:



Então e as excepções, que tipo de entidade são?

- As excepções são **objectos** (instâncias de uma classe específica);
- Lançar uma excepção corresponde a instanciar um determinado objecto;
- Capturar uma excepção é receber uma referência para o respectivo objecto (excepção).



... Como são definidas as Excepções em Java?

No lançamento de uma excepção em Java, é instanciado um objecto de uma classe que deriva de **java.lang.Exception**:

- class java.lang.Exception
 - class java.lang.ClassNotFoundException
 - class java.lang.CloneNotSupportedException
 - class java.lang.IllegalAccessException
 - class java.lang.InstantiationException
 - class java.lang.InterruptedException
 - class java.lang.NoSuchFieldException
 - class java.lang.NoSuchMethodException
 - class java.lang.RuntimeException
 - class java.lang.ArithmeticException
 - class java.lang.ArrayStoreException
 - class java.lang.ClassCastException
 - class java.lang.IllegalArgumentException
 - class java.lang.IllegalThreadStateException
 - class java.lang.NumberFormatException
 - class java.lang.IllegalMonitorStateException
 - class java.lang.IllegalStateException
 - class java.lang.IndexOutOfBoundsException
 - class java.lang.ArrayIndexOutOfBoundsException
 - class java.lang.StringIndexOutOfBoundsException
 - class java.lang.NegativeArraySizeException
 - class java.lang.NullPointerException
 - class java.lang.SecurityException
 - class java.lang.UnsupportedOperationException

Como é que lançamos as nossas próprias excepções?

Exemplo: contendor StackInt. Construtor:

```

StackInt TestStackInt
package pg2.aula05;
import pg2.io.IO;

public class TestStackInt {

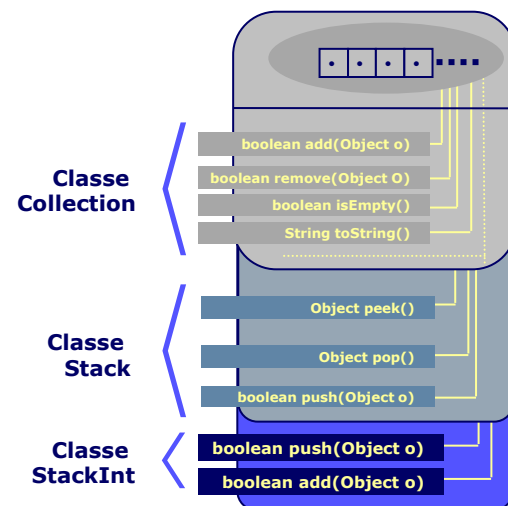
    public static void main (String args[]){
        StackInt opr=null;
        try {
            opr = new StackInt(Integer.parseInt(args[0]));
        }
        catch (Exception e){
            IO.cout.writeln("\n" + e);
            IO.cout.writeln("\n0 contendor foi inicializado com default 5\n");
            opr = new StackInt();
        }

        String str;
        //Testa inserção
        do{
            str = IO.cin.readLine();
        } while (opr.push(str));

        IO.cout.writeln("\n" + opr.toString() + "\n");

        //Testa remoção
        int i=0;
        while (!opr.isEmpty()) IO.cout.writeln(opr.pop().toString());
    }
}

```



```

C:\ Command Prompt
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula05.TestStackInt -7
java.lang.NegativeArraySizeException
0 contendor foi inicializado com dim 5
4
5
6
7
8
0
[<4><5><6><7><8>]
8
7
6
5
4
D:\MyFolder\ISEL\Pg II - I02\Projects>

```


... Como é que lançamos as nossas próprias excepções?

```

C:\ Command Prompt
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula05.TestStackInt -7
java.lang.NegativeArraySizeException:
  > 0 contendor deve ser inicializado com um valor positivo
0 contendor foi inicializado com default 5
2
3
4
5
6
0
[<2><3><4><5><6>]
6
5
4
3
2
D:\MyFolder\ISEL\Pg II - I02\Projects>

```

```

StackInt TestStackInt
package pg2.aula05;
/**
 * Title: Pilha de inteiros
 * @author MCarvalho
 * @version 1.1
 */
public class StackInt extends Stack {

    //Construtor
    public StackInt() {super();}

    public StackInt(int dim) throws RuntimeException {
        if (dim < 1)
            throw new NegativeArraySizeException("\n" +
            "  > 0 contendor deve ser inicializado com um valor positivo");
        else
            arr = new Object[dim];
    }
}

```

... Como é que lançamos as nossas próprias excepções?

Exemplo: contentor StackInt. Método Push.

```
StackInt TestStackInt

//Métodos de Instancia
public boolean add (Object o){
    if (!(StackInt.isInteger(o.toString())))
        return false;
    return super.add(Integer.valueOf(o.toString()));
}
public boolean push(Object o) {return this.add(o);}
```



```
//Métodos de Instancia
public boolean add (Object o){
    if (!(StackInt.isInteger(o.toString())))
        throw new NumberFormatException("\n" +
            "> Tentativa de insercao de um objecto não inteiro");
    return super.add(Integer.valueOf(o.toString()));
}
public boolean push(Object o) {return this.add(o);}
```

```
StackInt TestStackInt

String str;
//Testa inserção
loopLabel:
while(true) {
    str = IO.cin.readLine();
    try {
        if (!opr.push(str))break;
    }
    catch (NumberFormatException e) {
        IO.cout.writeln(e);
        continue loopLabel;
    }
}
```

```
Command Prompt

D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula05.TestStackInt 4
1
2
dsf
java.lang.NumberFormatException:
    > Tentativa de insercao de um objecto não inteiro
3
4
0

[<1><2><3><4>]

4
3
2
1
```

... Como é que lançamos as nossas próprias excepções?

Como é que se definem as próprias excepções de uma aplicação?

Resposta: Derivando uma das classes da Hierarquia **java.lang.Exception**

```
StackInt | TestStackInt | ElementNotInteger
package pg2.aula05;

public class ElementNotInteger extends RuntimeException {

    public ElementNotInteger() {
        super("\n  > Tentativa de insercao de um objecto não inteiro");
    }
}
```



```
StackInt | TestStackInt | ElementNotInteger

//Métodos de Instancia
public boolean add (Object o){
    if (!(StackInt.isInteger(o.toString())))
        throw new ElementNotInteger();
    return super.add(Integer.valueOf(o.toString()));
}
public boolean push(Object o) {return this.add(o);}
```

Hierarquia de Excepções

Porquê manter uma hierarquia de Excepções?

- Para programar de forma PPO e não procedimental;
- Tirar proveito do mecanismo de polimorfismo:
 - Fazendo a captura do tipo de excepção que está no topo da hierarquia.
- Se as excepções forem especificadas na própria instrução de lançamento, quando quisermos alterar as respectivas descrições temos que percorrer instrução a instrução...

- Ex: `throw RuntimeException ("Contentor Cheio")`

Numa hierarquia de Excepções apenas temos de alterar a respectiva Classe:

- Ex: `class ElementNotInteger extends RuntimeException`

Derivar de Exception <> RuntimeException

O que aconteceria se a excepção ElementNotInteger fosse derivada de **Exception** em vez de **RuntimeException**?

```
StackInt | TestStackInt | ElementNotInteger
package pg2.aula05;

public class ElementNotInteger extends Exception {

    public ElementNotInteger() {
        super("\n > Tentativa de insercao de um objecto não inteiro");
    }
}
```

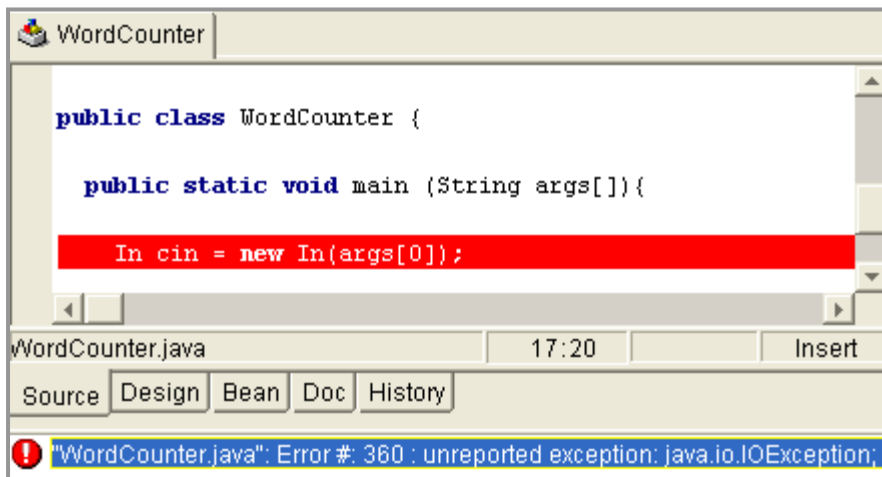
```
StackInt | TestStackInt | ElementNotInteger

//Métodos de Instancia
public boolean add (Object o){
    if (!(StackInt.isInteger(o.toString())))
        throw new ElementNotInteger();
    return super.add(Integer.valueOf(o.toString()));
}
public boolean push(Object o) {return this.add(o);}

StackInt.java 35:7 Insert
Source Design Bean Doc History
! "StackInt.java": Error #. 360 : unreported exception: pg2.aula05.ElementNotInteger; must be caught or declared to be thrown at line 35, column 7
Compiler
```

Classe In: construtor In(String filename)

A classe `pg2.io.In` disponibiliza ainda um outro construtor com parâmetro para que seja instanciado com um ficheiro de texto de entrada.



- class java.lang.[Throwable](#) (implements java.io.[Serializable](#))
 - class java.lang.[Exception](#)
 - class java.io.[IOException](#)
 - class java.io.[CharConversionException](#)
 - class java.io.[EOFException](#)
 - class java.io.[FileNotFoundException](#)
 - class java.io.[InterruptedIOException](#)
 - class java.io.[ObjectStreamException](#)
 - class java.io.[InvalidClassException](#)
 - class java.io.[InvalidObjectException](#)
 - class java.io.[NotActiveException](#)
 - class java.io.[NotSerializableException](#)
 - class java.io.[OptionalDataException](#)
 - class java.io.[StreamCorruptedException](#)
 - class java.io.[WriteAbortedException](#)

... Classe In: construtor In(String filename)

WordCounter

```
public class WordCounter {
    public static void main (String args[]){
        In cin = null;
        int tries = 0;
        String file;
        if (args.length > 0){
            file = args[0];
            tries = 3;
        }
        else {
            IO.cout.writeln("Introduza o caminho e o nome para "
                + "o ficheiro de entrada.");
            file = IO.cin.readLine();
        }
        loopLabel:
        while (cin == null){
            try {
                cin = new In(file);
            }
            catch ( IOException e){
                IO.cout.writeln("\n" + e +
                    "\n > Nome de ficheiro ou caminho invalido.\n");
                if (tries++ > 2)
                    System.exit(1);
                IO.cout.writeln("Introduza o caminho e o nome para "
                    + "o ficheiro de entrada.");
                file = IO.cin.readLine();
                continue loopLabel;
            }
        }
    }
}
```

Command Prompt

```
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter alunos.txt
java.io.FileNotFoundException: alunos.txt (The system cannot find the file speci
fied)
> Nome de ficheiro ou caminho invalido.
D:\MyFolder\ISEL\Pg II - I02\Projects>
```

Command Prompt

```
D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter
Introduza o caminho e o nome para o ficheiro de entrada:
alunos.txt
java.io.FileNotFoundException: alunos.txt (The system cannot find the file speci
fied)
> Nome de ficheiro ou caminho invalido.
Introduza o caminho e o nome para o ficheiro de entrada:
pg2\aula07\alunos.txt
D:\MyFolder\ISEL\Pg II - I02\Projects>
```

... Classe In: construtor In(String filename)

```

package pg2.aula07;
import java.util.HashMap;
import pg2.io.*;
import java.io.*;

class Counter {
    int i =1;
    public String toString() {
        return Integer.toString(i) + "\n";
    }
}

```

```

public class WordCounter {
    public static void main (String args[]){

```

...

```

HashMap hm = new HashMap(5);
while (!cin.eof()) {
    String s = cin.readWord();
    if (hm.containsKey(s))
        ((Counter) hm.get(s)).i++;
    else
        hm.put(s, new Counter());
}
IO.cout.writeln(hm.toString());
}

```

alunos.txt ...

File Edit Format View Help

Ricardo Cordeiro
Joao Lourenco
Paulo Santos
Ivo Esteves
Nuno Galego
Sergio Santos
Manuel Barros
Paulo Fernando
Nuno Alves
Hugo Almeida
Hugo Moreira
Filipe Bileu
Carlos Santos
Luis Loureiro
Rodrigo Sousa
Jorge Lopes
Ramiro Marques
Pedro Dias
Pedro Maria
Wagner dias

Command Prompt

D:\MyFolder\ISEL\Pg II - I02\Projects>java pg2.aula07.WordCounter

Introduza o caminho e o nome para o ficheiro de entrada:
pg2\aula07\alunos.txt

(A)lexandre=1
Mario=1
Ricardo=1
Luis=1
Hugo=3
Aperta=1
Maria=1
Goncalo=1
Bileu=1
Loureiro=1
Moreira=1
Edgar=1
Nuno=2
Martins=1
Esteves=1
Jorge=1
Cabaco=1
Santos=4
Pedro=3
Lopes=1
Sergio=2
Barros=1
Lourenco=1
Pacheco=1
Joao=2
Alves=1
Manuel=1
Almeida=1
Marques=1
Luz=1
Wagner=1
Fernando=1
Sousa=1
Carlos=1
Ivo=1
Dias=3
Cordeiro=1
Filipe=2
Galego=1
Paulo=2
Rodrigo=1
Ramiro=1

D:\MyFolder\ISEL\Pg II - I02\Projects>