iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 1: Programação em Linguagem Java

Aula 13
Exceções e Leitura e Escrita de Ficheiros (Scanner)

UPskill

Exceções

O que são **Exceções** em Java

O lançamento de exceções pode ser utilizado como um mecanismo para interromper a execução normal de um método, caso o objeto tenha sido utilizado de forma **incorreta ou indesejada**, por exemplo:

- Invocação de uma operação com argumentos inválidos.
- Sequência de invocações inválida.

As exceções elas próprias **são objetos** (com atributos, operações, e construtores).

Exemplos de **Exceções**

- Existem vários tipos já definidos de exceções em Java.
- Tipos relacionados com a utilização incorreta de objetos:
  - **IllegalArgumentException**: adequada quando um argumento inválido é utilizado na invocação de uma operação;
  - **NullPointerException**: adequada quando é passada uma referência null não permitida como argumento;
  - **IllegalStateException**: adequado quando é invocada uma operação não permitida dado o estado atual do objeto.

Lançamento de **Exceções**

IllegalArgumentException

```
class Point {
    final int x;
    final int y;

    Point(int x, int y) {
            if(x < 0 || y < 0){
                throw new IllegalArgumentException("Valores não negativos!”);
            }
            this.x = x;
            this.y = y;
    }
    //…
}
```

Estamos a lançar uma exceção caso o ponto tenha x ou y negativos. Utilizamos a IllegalArgumentException pois queremos representar uma exceção devido a argumentos inválidos.

Lançamento de Exceções

NullPointerException

```
class ImageUtils {

    static void invert(ColorImage img) {
        if(img == null){
            throw new NullPointerException("O argumento não pode ser null!");
        }
        //…
    }

    //…

}
```

Lançamento de Exceções

IllegalStateException

```
class IntSet {

     boolean isFull() {
          //…
     }

     void add(int element) {
          if(isFull())
               throw new IllegalStateException("O conjunto está cheio!”);
          //…
     }

}
```

Já sabemos **lançar** exceções. Mas... Para onde vão?

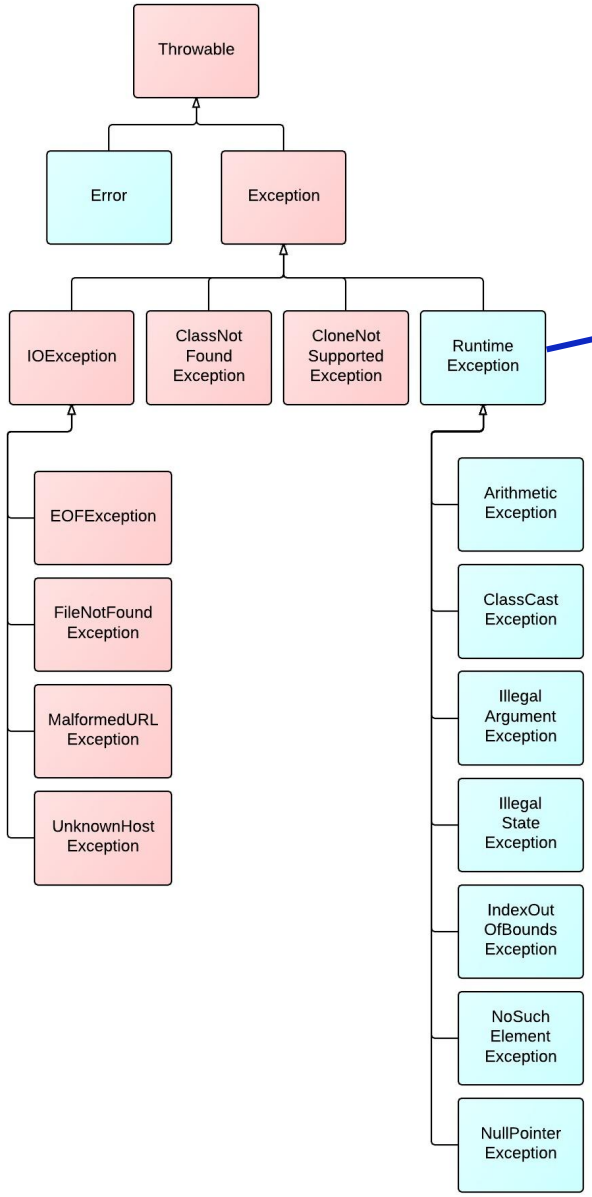
Quem é que **“apanha”** as exceções **lançadas**?

try - catch

Caso a exceção lançada aqui não tenha ninguém para a apanhar (ex. o bloco catch não for compatível com ela), é terminada a execução do programa.

```
try {
    //codigo a correr inicialmente
    //pode ser lançada uma exceção: throw new Exception();
} catch (Exception e) {
    //caso seja lançada uma uma excepção, é apanhada aqui
    //e é executado o código dentro deste bloco
} finally {
    //por fim... executa o código aqui
    //independentemente de ter havido uma exceção ou não
}
```

Tipos de **Exceções**



Exceções não verificadas, não é obrigatório apanhar.

É **obrigatório** apanhar exceções verificadas e é **opcional** apanhar exceções não verificadas (*RuntimeException*).

try - catch

```
Scanner keyboard = new Scanner(System.in);
System.out.println("Quantos anos tens?");
Integer age = null;
while(age == null) {
    try {
        age = keyboard.nextInt();
    } catch (Exception e) {
        System.out.println("Idade inválida. Tenta outra vez.");
    }
}
System.out.println("Tens " + age + " anos.");
```

Usamos o **Integer** em vez do primitivo **int** porque queremos inicializar com o valor vazio **null**.

É **lançada** uma excepção que pode ser tratada para pedir novo input.

Exceção lançada é **apanhada** neste bloco.

try - catch

```
try {
    String abc = 123;
} catch (Exception e) {
    //...
}
```

Não é possível apanhar erros de compilação! Apenas erros verificados durante a execução.

Throws

- Por vezes, não queremos apanhar a excepção e queremos antes delegar o seu tratamento para o método que invocou o método onde a excepção foi lançada.

- Fazemos isto quando “não temos forma de resolver o problema” no método atual, e como tal não vale a pena apanhar a excepção, pois não podemos fazer nada.

Throws - Exemplo

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

   try {
       trataFicheiro(new File("ficheiro.txt"));
   } catch (FileNotFoundException e) {
       //tratamento da excepção
   }

}

public static void trataFicheiro(File f) throws FileNotFoundException {
   Scanner s = new Scanner(f);
   //...
}
```

Neste método, aqui sim, conhecemos o ficheiro e podemos ter uma solução para a *FileNotFoundException*

Neste caso, recebemos o ficheiro por argumento. **O que podemos fazer se não existir? Nada...**
Assim, devemos delegar a excepção para "cima"

Leitura e Escrita de Ficheiros

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

Leitura e Escrita de Ficheiros

- Classe “`Scanner`”
  - Para leitura
  - Usada anteriormente para ler do teclado
  - Estabelece *fluxo* (interno) de **entrada** do ficheiro
- Classe “`PrintWriter`”
  - Para escrita
  - Interface semelhante ao **System.out**
  - Estabelece *fluxo* (interno) de **saída** para ficheiro
- Classe “`File`”
  - Representa o ficheiro

Revisão: **Scanner**

```
public class Main {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Como te chamas?");
        String name = keyboard.nextLine();
        System.out.println("Quantos anos tens?");
        int age = keyboard.nextInt();
        System.out.print("Olá " + name + "! ");
        System.out.println("Tens " + age + " anos.");
    }

}
```

Leitura de Ficheiros: **Scanner**

É necessário encapsular a inicialização do Scanner de ficheiros num bloco “try catch” porque existe a possibilidade de lançar uma excepção verificada: **FileNotFoundException.**

```
try {
   Scanner fileScanner = new Scanner(new File("test.txt"));
   String primeiraLinha = fileScanner.nextLine();
   System.out.println(primeiraLinha);
   fileScanner.close();
} catch (FileNotFoundException e) {
   e.printStackTrace();
}
```

É recomendado “fechar” o scanner de ficheiros quando já não é necessário.

Escrita de Ficheiros: **PrintWriter**

```
try {
    PrintWriter fileWriter = new PrintWriter(new File("novo_ficheiro.txt"));
    fileWriter.println("Primeira linha");
    fileWriter.println("Segunda linha!");
    fileWriter.println(12345);
    fileWriter.close();
} catch (FileNotFoundException e) {
    System.out.println("Não foi possível criar o ficheiro!");
}
```

Parecido ao System.out.println()

É **necessário** usar o .close() para o PrintWriter escrever no ficheiro e finalizar.

Exercício 1

Crie uma lista de Pessoas com a informação recolhida no ficheiro pessoas.txt. O conteúdo do ficheiro pessoas.txt está representado abaixo:

```
João;23;Lisboa
Maria;10;Porto
Rita;21;Gaia
José;39;Aveiro
Manel;25;Portalegre
Ana;33;Alenquer
Alex;19;Sintra
Jacinto;30;Guarda
Vanderlei;45;Portimão
```

Dica: use o método `split()` da classe String

Exercício 2

Crie um programa que começa por pedir ao utilizador para inserir o seu nome e idade e escreve essa informação para um ficheiro txt, com uma estrutura definida por si.

De seguida, deve ler o ficheiro que acabou de escrever e imprimir para a consola o nome e idade do utilizador.

Exercício 3

Implemente uma classe *ContaBancaria* que contém um método para adicionar a uma lista de movimentos, um novo movimento com a assinatura **`(String descrição, double valor)`** e um método para guardar num ficheiro as informações relativas a todos os movimentos feitos na conta.

Os movimentos devem ser guardados num ficheiro .txt com a seguinte formatação (descrição, valor):

```
almoço;12
lanche;5
```

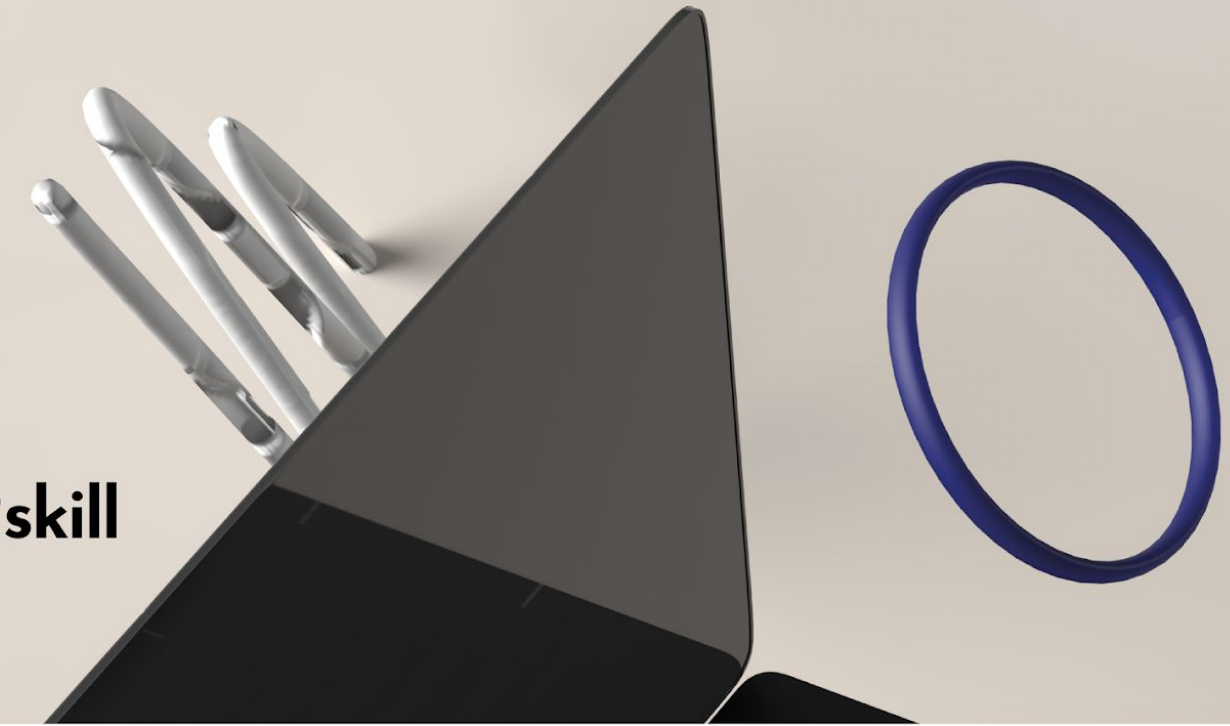
Por fim, crie um construtor alternativo que receba apenas o nome do ficheiro e construa o objecto ContaBancaria com base nos valores do ficheiro.

Exercício 4

Escreva um programa que seja capaz de ler um ficheiro que representa um mapa (em baixo, à esquerda) e, com base na informação lida, imprima no écran uma versão simplificada (em baixo, à direita) onde apenas aparecem os '#'s substituídos pelo caracter 'W'.

```
##########                    WWWWWWWWWW
#b  X   X#                    W        W
#        #                    W        W
#   C    #                    W        W
#        #          ==>       W        W
#      # #                    W      W W
#  C   #C#                    W      W W
#     O# #                    W      W W
#  X   #E#                    W      W W
##########                    WWWWWWWWWW
```

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill