Acceso a Datos

Tema 1

Excepciones

Excepciones

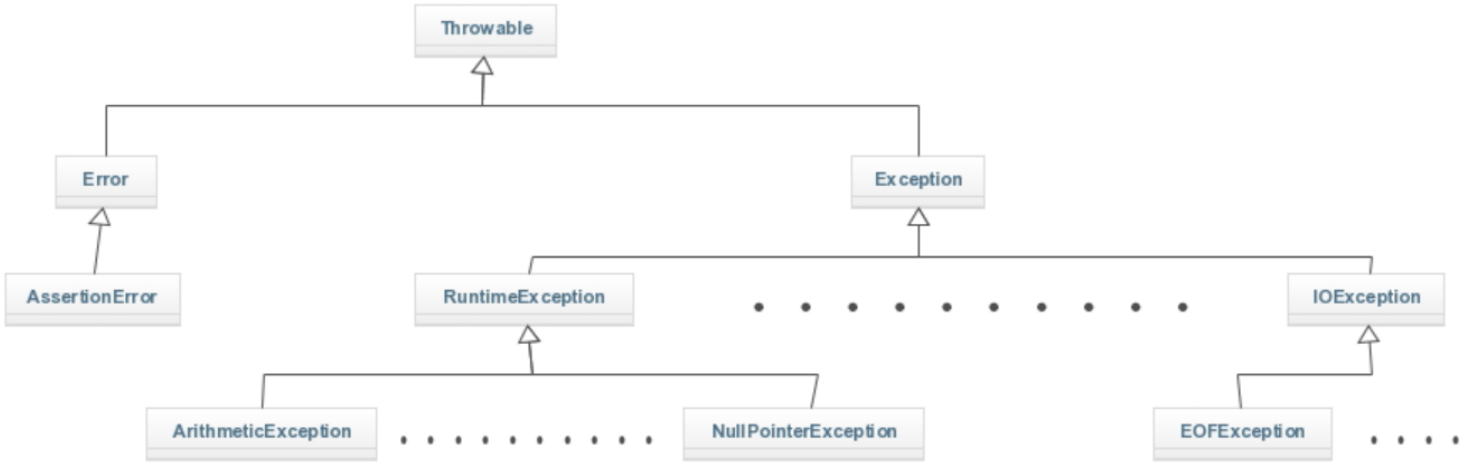
1. Excepciones
2. Control de excepciones
3. Excepciones Propias

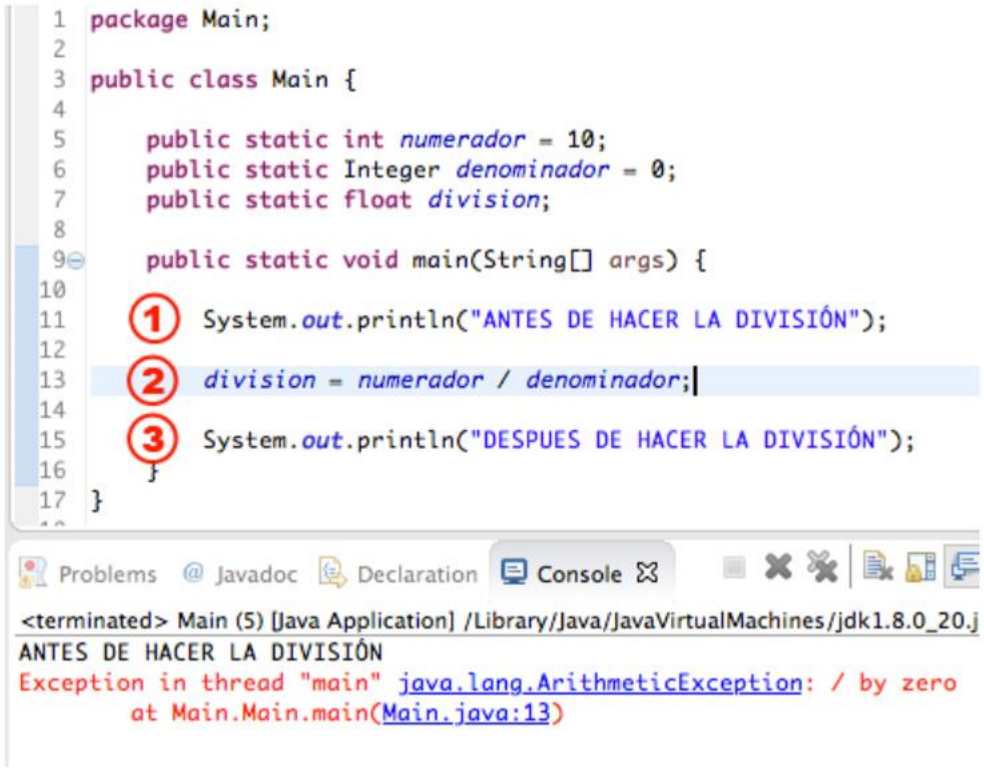
Definición

- Error en tiempo de ejecución
  - Se produce cuando una instrucción ejecuta una operación incorrecta
  - Ejemplo:
    - Una división por cero
    - Llamada a un método con un objeto null
    - Un índice fuera de rango
    - etc

1.- Excepción

- **Ejecución**
  - Cuando se ejecuta una excepción se crea un objeto ***"Throwable"***
  - Proporciona métodos para obtener información del error




1 package Main;
2
3 public class Main {
4
5 public static int numerador = 10;
6 public static Integer denominador = 0;
7 public static float division;
8
9 public static void main(String[] args) {
10
11 1 System.out.println("ANTES DE HACER LA DIVISIÓN");
12
13 2 division = numerador / denominador;
14
15 3 System.out.println("DESPUES DE HACER LA DIVISIÓN");
16 }
17 }
Problems Javadoc Declaration Console
<terminated> Main (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.j
ANTES DE HACER LA DIVISIÓN
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Main.Main.main(Main.java:13)

2.- Control de excepciones

- **Java permite controlar las instrucciones "peligrosas"**
  - Se evita que el programa rompa **"ABRUPTAMENTE"**
    - Aunque se produzca una excepción
  - Estructura "try-catch-finally"

2.- Control de Excepciones

```
1  package Main;
2
3  public class Main {
4
5      public static int numerador = 10;
6      public static Integer denominador = 0;
7      public static float division;
8
9      public static void main(String[] args) {
10         System.out.println("ANTES DE HACER LA DIVISIÓN");
11         try {
12             division = numerador / denominador;
13         } catch (ArithmeticException ex) {
14             division = 0; // Si hay una excepción doy valor '0' al atributo 'division'
15             System.out.println("Error: "+ex.getMessage());
16         } finally {
17             System.out.println("División: "+division);
18             System.out.println("DESPUES DE HACER LA DIVISIÓN");
19         }
20     }
21 }
```

Problems @ Javadoc Declaration Console

```
<terminated> Main (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (1
ANTES DE HACER LA DIVISIÓN
Error: / by zero
División: 0.0
DESPUES DE HACER LA DIVISIÓN
```

2. Control de Excepciones

```java
public class Main {

    public static int numerador = 10;
    public static Integer denominador = null;
    public static float division;

    public static void main(String[] args) {
        System.out.println("ANTES DE HACER LA DIVISIÓN");
        try {
            division = numerador / denominador;
        } catch (ArithmeticException ex) {
            division = 0; // Si hay una excepción doy valor '0' al atributo 'division'
            System.out.println("Error: "+ex.getMessage());
        } catch (NullPointerException ex) {
            division = 1; // Si si la excepción es de un null doy valor '1' al atributo 'division'
            System.out.println("Error: "+ex.getMessage());
        } finally {
            System.out.println("División: "+division);
            System.out.println("DESPUES DE HACER LA DIVISIÓN");
        }
    }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> Main (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (12/10/2014 16:
ANTES DE HACER LA DIVISIÓN
Error: null
División: 1.0
DESPUES DE HACER LA DIVISIÓN
```

3. Excepciones Propias

- **Errores en tiempo de ejecución propios de la aplicación**
  - Ejemplo:
    - Insertar nuevo cliente con dni duplicado
    - Eliminar un cliente que no existe
    - Insertar un precio de producto negativo
    - Formato incorrecto de datos , según el diseño de la aplicación

3. Excepciones Propias

- Una nueva clase que hereda de la clase **Exception**
  - Recibe un Srting como mensaje
    - Parámetro único de la clase padre

```
public class ExcepcionIntervalo extends Exception {
    public ExcepcionIntervalo(String msg) {
        super(msg);
    }
}
```

3.- Excepciones Propias

- **Un método que "lanza" una excepción propia**
  - Un método que lanza una excepción se le añade a continuación de la declaración la palabra reservada **throws**

```
static void rango(int num, int den)throws ExcepcionIntervalo{
   if((num>100)||(den<-5)){
       throw new ExcepcionIntervalo("Números fuera del intervalo");
   }
```

3.- Excepciones Propias

- **Captura de excepciones**
  - Incluyendo la llamada al método que lanza la excepción dentro de un bloque try-catch

```
public static void main(String[] args) {
    String str1="120";
    String str2="3";
    String respuesta;
    int numerador, denominador, cociente;
    try{
        numerador=Integer.parseInt(str1);
        denominador=Integer.parseInt(str2);
        rango(numerador, denominador);
        cociente=numerador/denominador;
        respuesta=String.valueOf(cociente);
    }catch(NumberFormatException ex){
        respuesta="Se han introducido caracteres no numéricos";
    }catch(ArithmeticException ex){
        respuesta="División entre cero";
    }catch(ExcepcionIntervalo ex){
        respuesta=ex.getMessage();
    }
    System.out.println(respuesta);
}
```