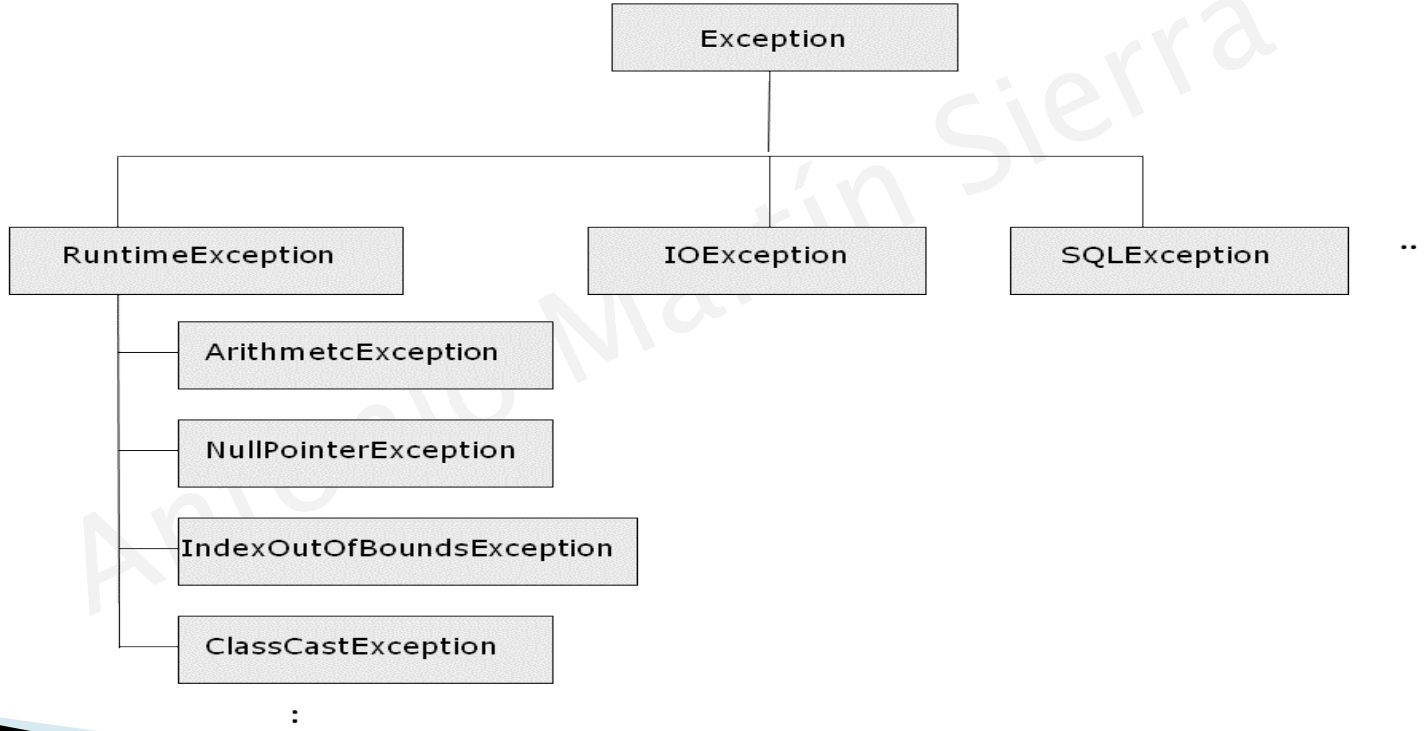
Excepciones

Concepto

- Una excepción es una situación anómala que se puede producir durante la ejecución de un programa
- Se producen debido a fallos de programación unas veces, y otras son debidas a situaciones que escapan al control del programador (error en la introducción de un dato de usuario, corrupción de un fichero, etc.)
- Un programa puede recuperarse de una excepción a través de una gestión de excepciones

Antonio Martín Sierra

Clases de excepción



Antonio Martín Sierra

Clasificación

- En función de su naturaleza, una excepción puede ser:
  - Unchecked. Conocidas también como excepciones de sistema, son todas las excepciones subclases de RuntimeException. Se producen por errores de programación y no es obligatorio capturarlas.
  - Checked. Son lanzadas por métodos de clases del API Java, específicas de cada clase. Es obligatorio capturarlas

Antonio Martín Sierra

Excepciones unchecked (I)

- ArrayIndexOutOfBoundsException. Intento de acceder fuera de los límites de un array:

```
int [] datos=new int[10];
datos[10]=3; //arrayIndexOutOfBoundsException
```

- NullPointerException. Acceso a métodos de un objeto con referencia a null:

```
class Test{
        String s; //inicializada a null
        public static void main(String[] args){
                int n=s.length();//NullPointerException
        }
}
```

- SecurityException. Producida por una violación de seguridad

Antonio Martín Sierra

Excepciones unchecked (II)

- **ClassCastException. Error al realizar una conversión de tipos:**

```
Object ob=new String("34");
Integer in=(Integer)ob;//ClassCastException
```

- **ArithmeticException. Operación matemática incorrecta:**

```
int n=5/0;//ArithmeticException
double r=3/0.0;//ok
```

- **IllegalArgumentException. Un método recibe como parámetro un valor que no es válido**

```
Thread.sleep(-100); //IllegalArgumentException
```

Antonio Martín Sierra

Errores

- A diferencia de una excepción, un error es una situación que se produce en un programa de la que este no se puede recuperar, como un fallo en la JVM, falta de espacio en memoria, etc.

- Sin embargo, los errores también están representados por clases que heredan Error: OutOfMemoryError, StackOverFlowError, InternalError, etc.

Antonio Martín Sierra

Captura de excepciones

Consideraciones

- **No puede haber ninguna instrucción entre los bloques try y catch:**

```
try{..}
System.out.println("hello"); //error de compilación
catch(…){}
```

- **Si se capturan varios tipos de excepciones que tienen relación de herencia entre ellas, los catch de las subclases deben ir antes que los de las superclases:**

Compilación correcta

```
catch(FileNotFoundException ex){
..
}
catch(IOException ex){
..
}
```

Error compilación

```
catch(RuntimeException ex){
..
}
catch(ArithmeticException ex){
..
}
```

Antonio Martín Sierra

Multicatch

- Si los catch de varias excepciones van a realizar la misma tarea, podemos agruparlos en un multicatch:

```
catch(IOException ex){
        System.out.println("error");
}
catch(SQLException ex){
        System.out.println("error");
}
```

multicatch

```
catch(IOException|SQLException ex){
        System.out.println("error");
}
```

- Las excepciones del multicatch no pueden tener relación de herencia, se produciría un error de compilación

Antonio Martín Sierra