TEMA 5. POO AVANZADA Y EXCEPCIONES.

TEMA5. EXCEPCIONES.

Posibles fallos en tiempo de ejecución.

- **Situaciones no contempladas:**
  - Acceso a variables a nulo.
  - Acceso a posiciones de vectores/matrices fuera de rango.
  - División entre 0.
  - Apertura, lectura o escritura de ficheros.
  - Uso de la red.
  - Fallo conexión base de datos.
  - ….

TEMA5. EXCEPCIONES.

Definición: Un error en tiempo de ejecución que se produce de manera inesperada, las causas pueden ser ajenas al software o por falta de previsión de ciertas situaciones.

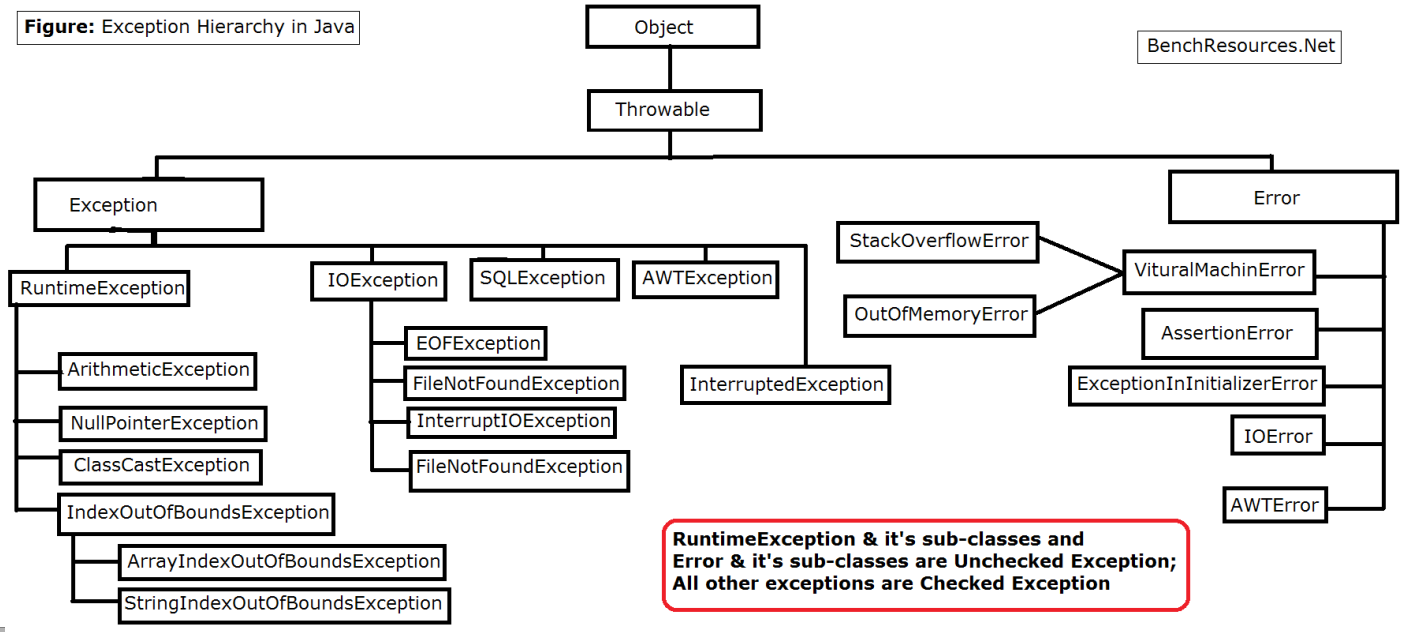
- **Situaciones no contempladas:**

```
public class Ejemplo1 {
   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      int dividendo,divisor,resultado;
      Scanner input= new Scanner(System.in);
      dividendo=input.nextInt();
      divisor=input.nextInt();
      resultado=dividendo/divisor;
      System.out.println("El resultado es "+resultado);
   }
}
```

```
5
0
Exception in thread "main"
java.lang.ArithmeticException: / by zero
     at
pedro.ieslaencanta.com.excepciones.Ejemplo1.main(
Ejemplo1.java:23)
Command execution failed.
```

Java.

- Jerarquía de excepciones, en función de la causa.



Java.

- **Todas heredan de Thowable.**
  - Contiene la pila de llamadas desde el método main hasta el punto en el que se produce el error, junto con un mensaje descriptivo.

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	addSuppressed(Throwable exception)	Appends the specified exception to the exceptions that were suppressed in order to deliver this exception.
Throwable	fillInStackTrace()	Fills in the execution stack trace.
Throwable	getCause()	Returns the cause of this throwable or null if the cause is nonexistent or unknown.
String	getLocalizedMessage()	Creates a localized description of this throwable.
String	getMessage()	Returns the detail message string of this throwable.
StackTraceElement[]	getStackTrace()	Provides programmatic access to the stack trace information printed by printStackTrace().
Throwable[]	getSuppressed()	Returns an array containing all of the exceptions that were suppressed, typically by the try-with-resources statement, in order to deliver this exception.
Throwable	initCause(Throwable cause)	Initializes the *cause* of this throwable to the specified value.
void	printStackTrace()	Prints this throwable and its backtrace to the standard error stream.
void	printStackTrace(PrintStream s)	Prints this throwable and its backtrace to the specified print stream.
void	printStackTrace(PrintWriter s)	Prints this throwable and its backtrace to the specified print writer.
void	setStackTrace (StackTraceElement[] stackTrace)	Sets the stack trace elements that will be returned by getStackTrace() and printed by printStackTrace() and related methods.
String	toString()	Returns a short description of this throwable.

Java.

- **De Thowable de forma directa heredan:**
  - Error: Error grave y de difícilmente tratamiento (no previsible).
    - Fallo de la máquina virtual.
    - Desbordamiento de la memoria.
    - Problemas con hilos.

    No se suele gestionar las situaciones que hacen lanzar este tipo de error, ya que no es responsabilidad del software y poco se puede hacer.

Java.

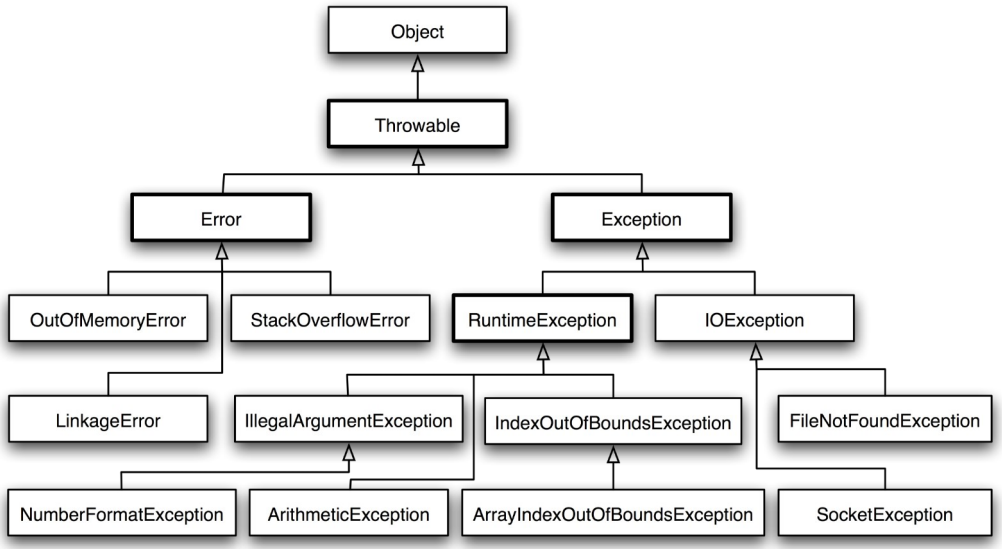
Ejemplo, desbordamiento de Heap (memoria dinámica limitada):

```
public static void main(String[]
args) {
        String str="desbordar";
        while(true){
            str+=str+"al montón";
        }
    }
```

```
Exception in thread "main"
java.lang.OutOfMemoryError: Overflow: String
length out of range
     at
java.base/java.lang.StringConcatHelper.checkOve
rflow(StringConcatHelper.java:57)
     at
java.base/java.lang.StringConcatHelper.mix(Stri
ngConcatHelper.java:138)
     at
pedro.ieslaencanta.com.excepciones.Desbordamien
toHeap.main(DesbordamientoHeap.java:19)
```

Java.

- **De Thowable de forma directa heredan:**
  - Exception: Errores producidos en tiempo de ejecución por el software, por ejemplo intentar acceder a una posición de un vector que no existe.



Estas si han de ser tratadas y previstas, al menos en los puntos críticos.

TEMA5. EXCEPCIONES.

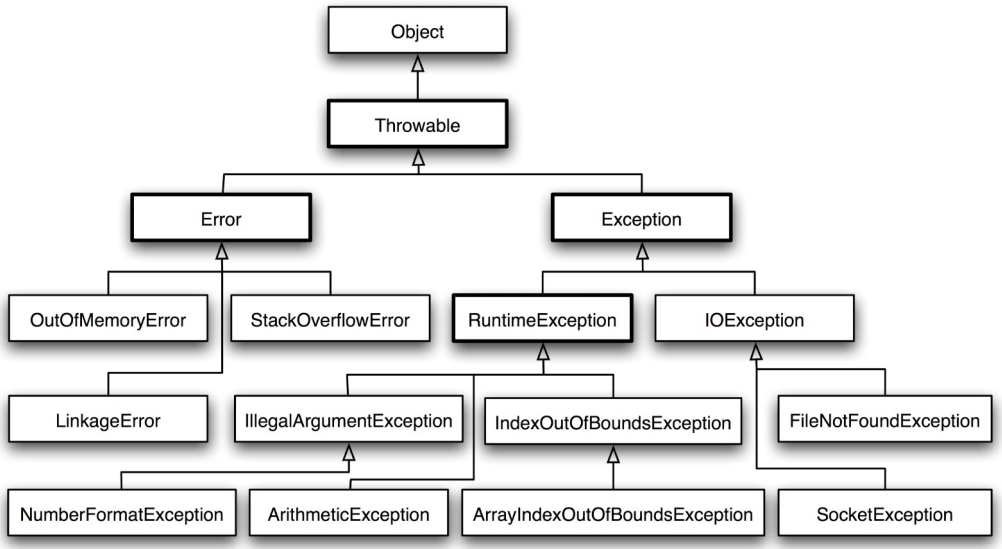
Java.

- **De Thowable de forma directa heredan:**
  - Exception: Errores producidos en tiempo de ejecución por el software, por ejemplo intentar acceder a una posición de un vector que no existe.

Estas si han de ser tratadas y previstas, al menos en los puntos críticos.

Se clasifican en 2 tipos:

- **RuntimeException: Relacionada con la programación-> x/0, fuera de índice…NO SE OBLIGA A SER CAPTURADAS.**
- **IOException: Relacionada con la I/O, no depende del programa, por ejemplo: HttpTimeOutException. ES OBLIGATORIO CAPTURARLAS. TIPO CHECKED.**



Java.

- **Ejemplo:**

```
FileWriter fichero = null;
fichero = new FileWriter(ruta);
fichero.write(this.aSVG());
fichero.close();
```

Compilar y también IDE:

```
*.java:89: error: unreported exception IOException; must be caught
or declared to be thrown
            fichero = new FileWriter(ruta);
*.java:90: error: unreported exception IOException; must be caught
or declared to be thrown
            fichero.write(this.aSVG());
*.java:91: error: unreported exception IOException; must be caught
or declared to be thrown
            fichero.close();
3 errors
```

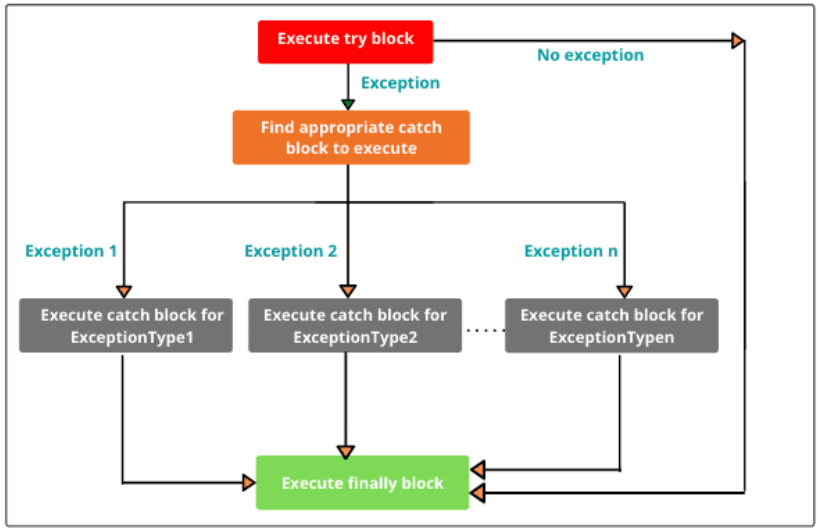
Java. Manejo y captura.

- Insertar código para capturar y manejar la excepción.
- Detectar los puntos en que se pueden producir e identificar el tipo o los tipos de Error/Excepción a tratar.
- Decidir si se trata en ese punto o se delega la responsabilidad en métodos superior.
- Se ha de indicar los tipos o subtipos de excepciones a tratar y como tratarlas de forma individual.

Java. Manejo y captura.

- **Estructura básica de manejo y captura.**

```
try{
        //código que puede hacer que se lance una exepción
}catch (Exception e1){
        //manejo de la excepción e1, que será alguno de los tipos que heredan de exception
}catch(Exception e2){
        //manejo de la excepción e2, diferente a e1, que será alguno de los tipos que heredan de exception
}catch(Exception….
        //se manejan todos los tipos que se deseee
finally{
        //código que se ejecuta se produzca o no la excepción
}
```



Java. Manejo y captura.

```
public void guardarDibujo(String ruta) {
  FileWriter fichero = null;
  try {
    fichero = new FileWriter(ruta);
    fichero.write(this.aSVG());
  } catch (IOException ex) {   System.err.println("Excepción por IO");
  } catch (Exception ex2) {     System.err.println("Excepción general, la más alta");
  } finally {
    try {
      fichero.close();
      System.out.println("Este código se ejeucta siempre, salte o no  salte");
    } catch (IOException ex) {
      Logger.getLogger(Dibujo.class.getName()).log(Level.SEVERE, null, ex);
  ....
```

- ¿Qué se ejecuta primero en caso de lanzarse una excepción ex o ex2?.
- ¿Qué codigo se ejecuta siempre, se lance o no se lance la excepción?
- ¿Cuál es la razón para que se cierre el fichero en finaly?
- ¿Qué consecuencias puede tener no hacer lo finally?

Recomendaciones

Nunca se ha de dejar sin código un catch ya que no se tendría constancia del fallo del programa, además es más que recomendable, casi obligatorio poseer algún tipo de sistema para realizar un log en fichero u otro medio que registre lo sucedido.

Java. Propagación.

Delegar el tratamiento de la posible excepción a métodos que llamen a este, de forma OBLIGATORIA.

Se define en la cabecera del método con la palabra reservada throws seguida del tipo de excepción a propagar.

```
public void guardarDibujo(String ruta) throws IOException {
        FileWriter fichero = null;
        fichero = new FileWriter(ruta);
        fichero.write(this.aSVG());
    }
```

Java. Excepciones personalizadas.

Las excepciones son clases que heredan de Exception.

Se pueden crear excepciones propias.

Capturar la excepción base y lanzarla con la nueva.

```
public class DivisionporCeroException extends ArithmeticException {
    public DivisionporCeroException(String msg){
        super(msg);
    }
    public DivisionporCeroException(){
        super("Fallo al dividir por 0");
    }
}
```

Java. Lanzar excepciones.

Uso incorrecto.

Situación no prevista. Ejemplo switch.

```
int dividendo,divisor,resultado;
Scanner input= new Scanner(System.in);
dividendo=input.nextInt();
divisor=input.nextInt();
try{
    resultado=dividendo/divisor;
}catch ( ArithmeticException e){
     throw  new
DivisionporCeroException("División por 0
personalizado "+e.getMessage());
}
System.out.println("El resultado es
"+resultado);
```

```
Exception in thread "main"
pedro.ieslaencanta.com.excepciones.DivisionporCer
oException: División por 0 personalizado / by
zero
     at
pedro.ieslaencanta.com.excepciones.Ejemplo1.main(
Ejemplo1.java:26)
```

Java. Aserciones.

Predicado. (expresión que se evalua a cierto o falso).

Se implementa desde la versión 1.4. Uso desaconsejado, solo utilizar en fase de desarrollo, no en producción.

```
public void ejemplo(Object parametro) {
    assert parametro!=null;
    if (null==parametro) {
        System.err.println("Parametro null");
        return;
    }
    ….
}
```

Ejecución:

```
java -ea MiAplicacion.jar
```

Java. Aserciones. Programación por contrato.

Condiciones y obligaciones.

Basado en aserciones.

Precondiciones: Condiciones que se han de cumplir para realizar la llamada. Es responsabilidad del elemento que hace uso del método cumplir con las condiciones.

Postcondiciones: Define que va a obtener el cliente, el método ha de garantizar de que se devuelva lo pactado.

Invariantes: Estado de un objeto que se ha de cumplir cuando se crea el objeto, así como antes y despueés de ejecutar cualquier método.

NO SOPORTADO DE FORMA NATIVA EN JAVA.

Java. Aserciones. Programación por contrato.

NO SOPORTADO DE FORMA NATIVA EN JAVA.

Librerías externas:

- Spring Framework.

```
public class Car {
    private String state = "stop";


    public void drive(int speed) {
        Assert.isTrue(speed > 0, "speed must be positive");
        this.state = "drive";
        // ...
    }
}
```

- Guava (Google). Varias como colecciones, almacenamiento, anotaciones...

Java. Aserciones. Programación por contrato.

Ejemplo Guava.

```
public Cuenta(String usuario, int saldo_inicial) {
        Preconditions.checkArgument(saldo_inicial > 0,
                "Illegal Argument passed: Negative value %s.", saldo_inicial);
        Preconditions.checkNotNull(usuario, usuario,
                "Please check the Object supplied, its null!");
        Preconditions.checkState(this.estado != ESTADO_CUENTA.ACTIVA,
                "Illegal state");
        this.usuario = usuario;
        this.saldo = saldo_inicial;
…
 public static void main(String[] args) {
        // TODO code application logic here
        Cuenta c= new Cuenta(null,10);
    }
Exception in thread "main" java.lang.NullPointerException: null [Please check the Object supplied, its null!]
     at com.google.common.base.Preconditions.checkNotNull(Preconditions.java:253)
     at pedro.ieslaencanta.com.precondicines.Cuenta.<init>(Cuenta.java:32)
     at pedro.ieslaencanta.com.precondicines.Principal.main(Principal.java:18)
```

Java. Aserciones. Programación orientada a aspectos.

Ampliación programación por contrato y aserciones.

Se desacopla partes transversales y comunes de la parte del dominio.

- Validación.
- Loggin.

Vídeo.