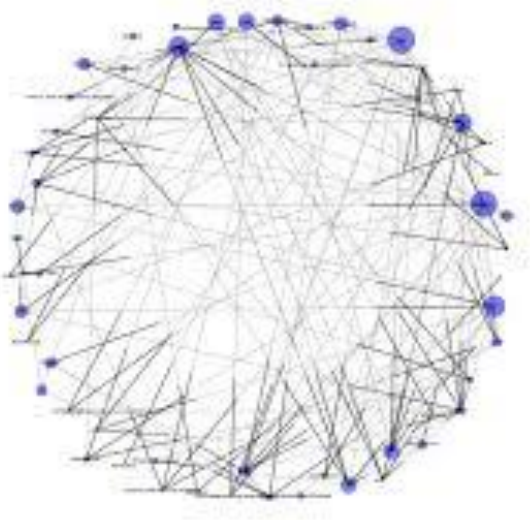
DAM
ACCESO A DATOS



UD1
MANEJO DE FICHEROS EN JAVA
7. EXCEPCIONES Y TRATAMIENTO

Excepciones

- Una excepción es un evento que sucede durante la ejecución de un programa e interrumpe el flujo normal de ejecución.
- Cuando una excepción no es capturada por el programa, la captura el gestor de excepciones por defecto enviando un mensaje a la consola y deteniendo abruptamente la ejecución del programa.
- **Ejemplo**: El siguiente código provoca una excepción al intentar realizar una división entre 0.

```
14    public static void main(String[] args) {
15
16        int num = 10, denom = 0, cociente;
17         cociente = num / denom;
18         System.out.println("Resultado:" + cociente);
19
20    }
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at gal.teis.general.Principal.main(Principal.java:17)
Command execution failed.
```

Mensaje que se muestra por consola al ejecutar la división entre 0. **AritmeticException**

Excepciones, paquetes y clases

- En Java, las excepciones están representadas por clases agrupadas en el paquete **java.lang.Exception**
- Todas las excepciones derivan de la clase **Throwable**.
- Derivadas de **Throwable** podemos encontrar dos clases **Error** y **Exception**.
  - **Error** → Se encarga de los errores que se produzcan en la máquina virtual, no en nuestros programas, por ejemplo, errores de memoria, disco duro roto, etc. Esta clase es usada por el runtime de Java para gestionar errores que ocurren en tiempo real.
  - **Exception** → Gestiona los errores que se producen en los programas, por ejemplo, error por división por cero, error por acceso a un array fuera de su memoria asignada, etc.; es la clase padre de las excepciones que debemos tratar en nuestros proyectos.
- Cuando se produce una excepción se **genera un objeto** asociado a la misma. Este objeto es de la clase **Exception** o de alguna de sus clases descendientes.
- También se pueden crear **excepciones personalizadas** en los programas.

Excepción	Significado
`IOException`	Problema de entrada o salida de datos
`ArithmeticException`	Desbordamiento o división entera por cero
`NumberFormatException`	Conversión ilegal de un string a un tipo numérico
`IndexOutOfBoundsException`	Acceso a un elemento inexistente de un vector o de un `String`
`NegativeArraySizeException`	Intento de creación de un vector de longitud negativa
`NullPointerException`	Intento de uso de una referencia nula
`SecurityException`	Violación de la seguridad en tiempo de ejecución

Tratamiento de excepciones

- Las excepciones se pueden capturar con una combinación de las instrucciones **try** y **catch**.
- Si alguna de las instrucciones en la parte **try** de la aplicación provoca una excepción, la ejecución salta al bloque **catch**.
- Las instrucciones del bloque **try** que hay después de la instrucción en la que se produce la excepción no son ejecutadas.

```
try {
    int num = 10, denom = 0, cociente;
    cociente = num / denom;
    System.out.println("Resultado:" + cociente);
} catch (Exception e) {
     System.out.println("Se ha producido una excepción");
}
```

La instrucción System… no se llega a ejecutar pues la excepción salta al realizar la división entre 0

Bloque finally

- El bloque **try/catch** puede tener un último bloque opcional denominado **finally**.
- Las instrucciones dentro del bloque **finally** tienen la garantía de que serán ejecutadas al final del bloque **catch**, se produzca una excepción o no.

```
public static void main(String[] args) {
        int num = 10, denom = 0, cociente;
        Scanner sc = new Scanner (System.in);
         try {
            num = sc.nextInt();
            cociente = num / denom;
            System.out.println("Resultado:" + cociente);
        } catch (Exception e) {
            System.out.println("Se ha producido una excepción");
        }finally{
            sc.close();
        }
 }
```

Se garantiza el cierre de la instancia de Scanner

Captura de varias excepciones

- Si las instrucciones del bloque **catch** pueden dar lugar a distintas excepciones, podemos capturar cada una de ellas de forma diferenciada implementando varios **catch**, uno por cada tipo de excepción.
- La captura múltiple de excepciones debe ordenarse de subclase a superclase.
- La captura de **Exception** debe ser el último catch, sino nunca se capturarían las excepciones que estuviesen capturadas después.

```
try {
    // Código regular del programa
    // Puede producir excepciones
} catch(TipoDeExcepcion1 e1) {
    // Código que trata las excepciones de tipo
    // TipoDeExcepcion1 o subclases de ella.
    // Los datos sobre la excepción los encontraremos
    // en el objeto e1.
} catch(TipoDeExcepcion2 e2) {
    // Código que trata las excepciones de tipo
    // TipoDeExcepcion2 o subclases de ella.
    // Los datos sobre la excepción los encontraremos
    // en el objeto e2.
...
} catch(Exception eN) {
    // Código que trata las excepciones de tipo
    // TipoDeExcepcionN o subclases de ella.
    // Los datos sobre la excepción los encontraremos
    // en el objeto eN.
}
```

Mostrar el contenido del objeto capturado por la excepción.

- En el bloque **catch** pueden ser útiles algunos métodos de la excepción (que podemos ver en la API de la clase padre **Exception**):

```
String getMessage()
void printStackTrace()
```

- Con **getMessage** obtenemos una cadena descriptiva del error (si la hay).
- Con **printStackTrace** se muestra por la salida estándar la traza de errores que se han producido (en ocasiones la traza es muy larga y no puede seguirse toda en pantalla con algunos sistemas operativos).

```
try {
    ... // Aqui va el codigo que puede lanzar una excepcion
} catch (Exception e) {
    System.out.println ("El error es: " + e.getMessage());
    e.printStackTrace();
}
```

Derivar el tratamiento de excepciones

- Se puede derivar el tratamiento de excepciones al método que ha llamado a uno dado utilizando la instrucción **throws** en la cabecera del método.
- Por ejemplo:

```
public void lee_fichero()  throws IOException, FileNotFoundException
{
   // Instrucciones que pueden generar las excepciones indicadas en throws
}
```

- En el caso de que en **lee_fichero()** se produzca una **IOException** o una **FileNotFoundException**, se envía las excepciones al **método que llamó a lee_fichero()** para que sean tratadas.
- El método que trata las excepciones es el que tiene que tener definido bloque **try/catch**.
- Las excepciones pueden ser derivadas en cadena, es decir, si el método A() lanza excepciones al método B(), este a su vez puede enviarlas al método C(). Suponemos que el método C(), llama a B() y éste a A().
- En último término siempre tendrán que ser tratadas en el bloque **main()**.

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

    try {
        leerFicheroBytes();

    } catch (FileNotFoundException ex) {
        System.out.println("No existe el fichero");

    }catch (IOException ex){
        System.out.println("Se ha producido la excepción “
                           +ex.getMessage());
    }
}

static void leerFicheroBytes()
                    throws FileNotFoundException, IOException {
    InputStream miFIS = new FileInputStream("texto.txt");
    int valor = miFIS.read();
    while (valor != 1) {
        System.out.print((char) valor);
        valor = miFIS.read();
        double a = .8_9;
    }
}
```

Trata las excepciones que se producen en **leerFicheroBytes()**

leerFicheroBytes() no trata las excepciones sino que las lanza a **main()**

Lanzamiento de excepciones por el programador

- Es posible utilizar la instrucción **throw** para lanzar excepciones.

  throw <objeto Exception>

- La excepción que se genera es un objeto que debe ser creado como cualquier otro objeto usando la palabra reservada **new**.
- Las excepciones que se pueden lanzar con **throw** pueden ser:
  - Objetos de la clase **System.Exception** o sus clases derivadas.
    - El programador lanza una excepción con la sintaxis

      throw new ***ClaseExceptionODerivada***

    - Las excepciones que se lanzan con **throw** no las provoca el sistema, sino que el programador la lanza para utilizarlas como mecanismo para detectar algún error de ejecución que no tiene que ver con las excepciones.
  - Objetos de clases creadas por el programador.
    - El programador debe crear una clase derivada de **Exception** que implementa como se captura una excepción o un error.
    - Las aplicaciones suelen definir sus propias subclases de la clase **Exception** para representar situaciones excepcionales específicas de cada aplicación.

Lanzamiento de excepciones por parte del programador:

Objetos de la clase **System.Exception** o sus clases derivadas

Lanzamiento de excepciones: Objetos de la clase **System.Exception** o sus clases derivadas. **Ejemplo1**

```
public class EjemploExcepciones {

    static int minutos;
    static boolean correcto = false;
    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        do {
            try {
                leerMinuto();
            } catch (InputMismatchException e) {
                System.out.println("El tipo de datos introducidos no es correcto,"
                        + " se debe introducir un entero");
            } catch (Exception e) {
                System.out.println("El valor introducido debe estar entre 1 y 60");
            }finally{
                sc.nextLine();
            }
        } while (!correcto);
    }

    public static void leerMinuto() throws InputMismatchException, Exception {
        System.out.println("Introduce los minutos");
        minutos = sc.nextInt();
        if (minutos < 1 || minutos >= 60) {
            throw new Exception();
        }
        correcto = true;
    }
}
```

Trata las excepciones que se producen en leer**Minuto()**

leerMinuto() no trata las excepciones sino que las lanza a **main()**

Lanza una excepción cuando minutos no está en el rango establecido

Ejemplo de excepción Java lanzada por el programador

Lanzamiento de excepciones: Objetos de la clase **System.Exception** o sus clases derivadas. **Ejemplo2**

```
public class DemoException {

    static void miMetodo(int n) throws Exception {
        try {
            switch (n) {
                case 1:
                    System.out.println("1er case");
                    break;
                case 2:
                    System.out.println("2er case");
                    throw new RuntimeException("2!");
                case 3:
                    System.out.println("3er case");
                    throw new Exception("3!");
            }
        } catch (RuntimeException e) {
            System.out.print("RuntimeException: ");
            System.out.println(e.getMessage());
        } finally{
            System.out.println("try-block ejecutado");
        }
    }
}
```

Ejemplo de excepción Java lanzada por el programador

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

    for (int i = 1; i < 4; i++) {
        try {
            DemoException.miMetodo(i);
        } catch (Exception ex) {
            System.out.println("Excepción capturada "+
                    ex.getMessage());
        }
    }
}
```

Lanzamiento de excepciones por parte del programador:

Objetos de clases creadas por el programador

- Los pasos a seguir para crear una excepción propia son:
  1. Crear una clase que extienda de la clase **RuntimeException** o **Exception**.
  2. Configurar la clase con sus constructores y aportarle la información que debe mostrarse en el método **getMessage()**.
  3. Se aconseja llamar a la clase con un nombre que describa e problema que detecta y finalice con ..**Exception**
- Ejemplo:

```
public class NumeroFueraRangoException extends Exception{

   public NumeroFueraRango(int i_rango, int f_rango) {
       super("Número fuera de rango, debe introducir un nº entre "
             + i_rango+" y "+f_rango);
    }
}
```

Excepción creada por el programador. Clase que extiende de Exception

```
public static void main(String[] args) {
    try {
        int x = 10;
        if (x < 20 || x > 100) {
            throw new NumeroFueraRangoException(20,100);
        }
    } catch (NumeroFueraRango e) {
        System.out.println(e.getMessage());
    }
}
```

Lanzamiento de la excepción creada por el programador

- **Práctica**
  - Modificar la librería **LibreriaDam** que contiene un control de rango para que el mensaje de fuera de rango se emita con una excepción propia que deberá incluirse también el proyecto.
  - Crear un proyecto que incorpore **LibreriaDam** con las modificaciones y que muestre un menú con 2 opciones.
  - En el momento de introducir la opción del menú es cuando se debe capturar la excepción.
  - Debe repetir la petición de la opción hasta que se introduzca un valor correcto.

- **Consulta**
- https://personales.unican.es/corcuerp/java/Labs/LAB_15.htm
- https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?punto=86&codigo=166&inici