A Simple Exception Example

```
public class Ejemplo1 {
   public static void main(String args[]) {
      int nums[] = new int[4];
      try {
         System.out.println("Antes de que se genere la excepción");
         // Generando an index out-of-bounds exception.
         nums[7] = 10;
         System.out.println("Este mensaje no se emitirá");
      } catch (ArrayIndexOutOfBoundsException exc) {
         // catch the exception
         System.out.println("Index out-of-bounds!");
      }
      System.out.println("Mensaje posterior al cacth");
   }
}
```

```
      class Prueba {
      // Generate an exception.
         static void generaException() {
               int nums[] = new int[4];
               System.out.println("Antes de ser generada");
               // generate an index out-of-bounds exception
               nums[7] = 10;
               System.out.println("Este mensaje no se emitirá");
            }
      }
      class Ejemplo2 {
            public static void main(String args[]) {
               try {
                  Prueba.generaException();
               }
               catch (ArrayIndexOutOfBoundsException exc) {
                     // catch the exception
                     System.out.println("Index out-of-bounds!");
               }
               System.out.println("Mensaje posterior a catch");
            }
      }
```

```
class Ejemplo3 {
   public static void main(String args[]) {
      int numerador[] = {4, 8, 16, 32, 64, 128, 256, 512}; //Longitud 8
      int denominador[] = {2, 0, 4, 4, 0, 8}; //Longitud 6
      try {
         for (int i = 0; i < numerador.length; i++) {
            System.out.println(numerador[i] + " / " + denominador[i] + " is " +
numerador[i] / denominador[i]);
         }
      } catch (ArithmeticException exc) {
         System.out.println("No se puede dividir por cero!");
      } catch (ArrayIndexOutOfBoundsException exc) {
         System.out.println("No existe la celda ");
      }
   }
}
```

```
class Ejemplo4 {
   public static void main(String args[]) {
      // Here, numer is longer than denom.
      int numer[] = {4, 8, 16, 32, 64, 128, 256, 512};
      int denom[] = {2, 0, 4, 4, 0, 8};
      for (int i = 0; i < numer.length; i++) {
         try {
            System.out.println(numer[i] + " / " + denom[i] + " is "+ numer[i] / denom[i]);
         } catch (ArrayIndexOutOfBoundsException exc) { //Subclase debe ir antes
            // catch the exception
            System.out.println("No se ha encontrado el denominador");
         } catch (Throwable exc) { //Superclase
            System.out.println("Recoge cualquier excepción");
         }
      }
   }
}
```

```
class Ejemplo5 {
```

```
    public static void main(String args[]) {
      int numer[] = {4, 8, 16, 32, 64, 128, 256, 512};
      int denom[] = {2, 0, 4, 4, 0, 8};
      try { // Exterior
        for (int i = 0; i < numer.length; i++) {
          try { // Anidado falla en indice 6 y no tiene captura se
propaga al bloque externo
            System.out.println(numer[i] + " / "+ denom[i] + " is "+ numer[i] / denom[i]);
          } catch (ArithmeticException exc) {
            // catch the exception
            System.out.println("Can't divide by Zero!");
          }
        }
      } catch (ArrayIndexOutOfBoundsException exc) {
          // catch the exception
        System.out.println("No matching element found.");
        System.out.println("Fatal error - program terminated.");
      }
    }
```

```
class Ejemplo6 {
  public static void main(String args[]) {
    try {
      System.out.println("Antes de la excepción");

      throw new ArithmeticException("Mi mensaje de error");
      //throw new ArithmeticException();

    } catch (ArithmeticException exc) {
      // catch the exception
      exc.printStackTrace();
    }
    System.out.println("Después de try/catch");
  }
}
```

otro ejemplo

```
package excepciones;
import java.util.*;

class ejemplo7 {

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

```
        int n=(int)(Math.random()*10+1);

          try {

               System.out.println("Antes de la excepción");

               if (n%2==0)

throw new Throwable("");

               //throw new ArithmeticException();


          } catch (Throwable exc) {

                System.out.println("el numero era "+n+" conseguí mi
excepcion");

              // exc.printStackTrace();

          }

          System.out.println("Después de try/catch");

    }

}
```

```
        class PruebaExcepción {
          static void generaException() {
            int nums[] = new int[4];
            System.out.println("Mensaje anterior a la excepción");
            // generate an index out-of-bounds exception
            nums[7] = 10;
            System.out.println("Este mensaje no se emitirá");
          }
        }

        class Ejemplo7 {
          public static void main(String args[]) {
            try {
              PruebaExcepción.generaException();
            } catch (ArrayIndexOutOfBoundsException exc) {
              System.out.println("Mensaje estándar");
              System.out.println(exc);
              System.out.println("\nTraza de la excepción");
              exc.printStackTrace();
            }
            System.out.println("Mensaje posterior a catch");
          }
        }
```

```
class UseFinally {
   public static void genException(int valor) {
      int t;
      int nums[] = new int[2];
      System.out.println("Receiving " + valor);
      try {
         switch (valor) {
            case 0:
               t = 10 / valor; // Genera división por 0
               break;
            case 1:
               nums[4] = 4; // Genera error de índice
               break;
            case 2:
               System.out.println("Una ejecución normal, sin error");
               break;
            case 3:
               return; // Sale del bloque try-catch

         }
      } catch (ArithmeticException exc) {
         System.out.println("Can't divide by Zero!");
         return; // sale del método, luego no pasa por println("Others")
      } catch (ArrayIndexOutOfBoundsException exc) {
         System.out.println("No matching element found.");
      } finally {
         //Siempre se ejecuta aunque haya un return
         System.out.println("SIEMPRE PASA POR AQUÍ AUNQUE HAYA UN
RETURN ANTES (Case 0)");
      }
      System.out.println("Others"); //Siempre se ejecuta a no ser que haya un
return previo
   }
}
```

```
class Ejemplo8 {

   public static void main(String args[]) {
      for (int i = 0; i <= 4; i++) {
         UseFinally.genException(i);
         System.out.println();
      }
   }
}
```

Using throws

*En algunos casos, si un método genera una excepción que no maneja, debe declarar esa excepción en una cláusula **throws***

```
class Ejemplo9 {
  public static char prompt(String str) throws java.io.IOException { //Se
recogerá fuera
    System.out.print(str + ": ");
    return (char) System.in.read();
  }

  public static void main(String args[]) {
    char ch;
    try {
      ch = prompt("Escriba una letra");
    } catch (java.io.IOException exc) {
      System.out.println("Error de entrada salida");
      ch = 'X';
    }
    System.out.println("Has escrito " + ch);
  }
}
```

```
class Ejemplo10 {
  public static void main(String args[]) {
    int a = 88, b = 0;
    int result;
    char chrs[] = {'A', 'B', 'C'};
    for (int i = 0; i < 2; i++) {
      try {
        if (i == 0) {
          result = a / b; // generate an ArithmeticException
        } else {
          chrs[5] = 'X'; // generate an ArrayIndexOutOfBoundsException
        }
      } catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception caught: " + e);
      }
    }
    System.out.println("After multi-catch.");
  }}
```

Tabla 1 (No es obligatorio lanzarlas,son implicitas)

Exception	Meaning
ArithmeticException	Arithmetic error, such as integer divide-by-zero.
ArrayIndexOutOfBoundsException	Array index is out-of-bounds.
ArrayStoreException	Assignment to an array element of an incompatible type.
ClassCastException	Invalid cast.
EnumConstantNotPresentException	An attempt is made to use an undefined enumeration value.
IllegalArgumentException	Illegal argument used to invoke a method.
IllegalMonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException	Environment or application is in incorrect state.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	Array created with a negative size.
NullPointerException	Invalid use of a null reference.
NumberFormatException	Invalid conversion of a string to a numeric format.
SecurityException	Attempt to violate security.
StringIndexOutOfBoundsException	Attempt to index outside the bounds of a string.
TypeNotPresentException	Type not found.
UnsupportedOperationException	An unsupported operation was encountered.

Tabla 2 (explicitas, Es obligatorio lanzarlas si se quieren manipular)

Exception	Meaning
ClassNotFoundException	Class not found.
CloneNotSupportedException	Attempt to clone an object that does not implement the **Cloneable** interface.
IllegalAccessException	Access to a class is denied.
InstantiationException	Attempt to create an object of an abstract class or interface.
InterruptedException	One thread has been interrupted by another thread.
NoSuchFieldException	A requested field does not exist.
NoSuchMethodException	A requested method does not exist.
ReflectiveOperationException	Superclass of reflection-related exceptions.
IOException	Produced by failed or interrupted I/O operations
FileNotFoundException	Attempt to open the file denoted by a specified pathname has failed

```
class NonIntResultException extends Exception {
  int n;
  int d;
  NonIntResultException(int i, int j) {
    n = i;
    d = j;
  }
  public String toString() {
    return "Result of " + n + " / " + d + " is non-integer.";
  }
```

```
}
class Ejemplo12 {
   public static void main(String args[]) {
      int numer[] = {4, 8, 15, 32, 64, 127, 256, 512};
      int denom[] = {2, 0, 4, 4, 0, 8};
      for (int i = 0; i < numer.length; i++) {
         try {
            if ((numer[i] % 2) != 0) {//Si no es par aviso de que el resultado no
será entero
               throw new NonIntResultException(numer[i], denom[i]);
                  //Se crea la excepción y es desde try-catch desde se invoca la
            visualización del
                  //mensaje de error
            }
            System.out.println(numer[i] + " / " + denom[i] + " is " + numer[i] /
denom[i]);
         } catch (ArithmeticException exc) {
            System.out.println("Can't divide by Zero!");
         } catch (ArrayIndexOutOfBoundsException exc) {
            System.out.println("No matching element found.");
         } catch (NonIntResultException exc) {
            System.out.println(exc);
         }
      }
   }
}
```

Ejercicios para hacer vosotros

1.Supongamos la siguiente clase:

```
public class Ejemplo1 {

   public static void main(String[] args) {
      final int NUM = 5;
      int[] enteros = new int[NUM];
      int posicion = 0;
      Scanner teclado = new Scanner(System.in);
      int cont = 0;
      int divisor = 0;
      while (cont < NUM) {
         System.out.println("Introduce una posición del array:");
         posicion = Integer.parseInt(teclado.nextLine());
         System.out.println("Introduce un divisor:");
```

```
            divisor = Integer.parseInt(teclado.nextLine());
            enteros[posicion] = 5 / divisor;
            cont++;
        }
        System.out.println("El contenido del array de enteros es:");
        for (int valor : enteros) {
            System.out.println(valor);
        }
    }
}
```

Modifica el código para que el programa capture las excepciones que se indican y correspondientes mensajes para después continuar con la ejecución del programa:

IndexOutOfBoundsException, NumberFormatException (valor no entero), ArithmeticException (division por 0)

2.Crea una clase Alumno con los atributos:

```
private String nombre;
private int edad;
private float nota;
```

Y el siguiente constructor, que deberá lanzar una excepción propia cuando la edad sea inferior a 10 (**EdadNoValidaException** con su propio mensaje de error).

```
public Alumno(String nombre, int edad) …
```

La clase tendrá los siguientes métodos:

```
public float getNota()
public void setNota(float nota)
public String getNombre()
public int getEdad()
```

El método setNota, no deberá asignar notas inferiores a 0 o superiores a 10, y se protegerá lanzando una excepción cuando la nota sea errónea (**NotaNoValidaException**, con su propio mensaje de error).

Crea una clase con un método main que lea desde teclado un array de 5 alumnos con sus notas y posteriormente muestre la información de cada uno de ellos. Se deberán capturar todas las excepciones que se pudieran generar
y mostrar los mensajes correspondientes a cada tipo de excepción.

3.a)crea una excepcion propia llamada Limites

b)crear 2 excepciones derivadas de Limites demasiadoCalor, y demasiadoCansado

c)crea un main que recoja la temperatura y las horas de sueño, si más de 40º o menos de 8 horas lance las excepciones

Recomendaciones finales

1. Limpia los recursos en el bloque Finally
2. Utiliza Excepciones específicas
3. Lanza excepciones con mensajes descriptivos
4. Captura la excepción más específica primero
5. No captures con Throwable.
6. No ignores las excepciones, aunque sea un mensaje