

PROGRAMMING  
LANGUAGE

JAVA

FILE HANDLING  
FUNCTIONS  
MULTITHREADING  
OPERATORS  
OOP  
**BASICS**  
INHERITANCE  
OPERATORS  
ARRAYS  
WRAPPER CLASSES  
JAVA PACKAGES

## T10 – Java Exceptions

# EJERCICIOS

## 1.

Escribe un programa, utilizando para ello el paradigma de POO, que juegue con el usuario a adivinar un número. Debe cumplir los siguientes requerimientos:

1. El ordenador debe generar un número entre 1 y 500, y el usuario tiene que intentar adivinarlo.
2. Cada vez que el usuario introduce un valor, el ordenador debe decirle al usuario si el número que tiene que adivinar es mayor o menor que el que ha introducido el usuario.
3. Cuando consiga adivinarlo, debe indicárselo e imprimir en pantalla el número de veces que el usuario ha intentado adivinar el número.
4. Si el usuario introduce algo que no es un número, debe indicarlo en pantalla, y contarlo como un intento indicando que no ha conseguido reconocer la entrada lanzando la excepción **InputMismatchException**.

# EJERCICIOS

## 2.

Escribe un programa , utilizando para ello el paradigma de POO, que lance y capture una excepción customizada. Crea para ello una package diferente que puedas reutilizar para el resto de tus proyectos.

### Recomendaciones:

El programa abre un bucle `try{}` en el que comienza mostrando un mensaje por pantalla. A continuación, crea un objeto de la clase *Exception*, indicando en su constructor un mensaje explicativo.

El resultado debe ser similar a esta captura de pantalla:

```
Mensaje mostrado por pantalla  
Excepcion capturada con mensaje: Esto es un objeto Exception  
Programa terminado
```

# EJERCICIOS

## 3.

Escribe un programa , utilizando para ello el paradigma de POO, que genere un número aleatorio e indique si el número generado es par o impar. El programa utilizará para ello el lanzamiento de una excepción.

### Recomendaciones:

1. El programa utiliza la clase Random() para obtener un número aleatorio entre 0 y 999 (por poner un rango cualquiera).
2. Se determina si el número es par o impar y se lanza una excepción con el correspondiente mensaje para indicarlo (se limitará a mostrar el mensaje asociado a la excepción capturada).

El resultado debe ser similar a esta captura de pantalla:

```
Generando número aleatorio...  
El numero aleatorio generado es: 203  
Es impar
```

# EJERCICIOS

## 4.

Escribe un programa , utilizando para ello el paradigma de POO, que nos permita realizar cálculos simples (suma, resta, multiplicación, potencia, raíz cuadrada, raíz cubica y división). El programa ha de estar preparado para gestionar los posibles errores de calculo. Has de utilizar para ello el control de excepciones de JAVA.

### **Recomendaciones:**

1. Utiliza siempre que sea posible las Excepciones definidas en la API de Java 8.
2. Puedes utilizar como interfaz visual Scanner o JOptionPane.
3. Estructura correctamente el código en diferentes packages.

# EJERCICIOS

**5)** Haz una clase llamada **Password** que siga las siguientes condiciones:

- Que tenga los atributos **longitud** y **contraseña** . Por defecto, la longitud sera de 8.
- Los constructores serán los siguiente:
  - ✓ Un constructor por defecto.
  - ✓ Un constructor con la longitud que nosotros le pasemos. Generara una contraseña aleatoria con esa longitud.

Los métodos que implementa serán:

- **esFuerte()**: devuelve un booleano si es fuerte o no, para que sea fuerte debe tener mas de 2 mayúsculas, mas de 1 minúscula y mas de 5 números.
- **generarPassword()**: genera la contraseña del objeto con la longitud que tenga.
- Método get para contraseña y longitud.
- Método set para longitud.

# EJERCICIOS

Ahora, crea una clase clase ejecutable:

- Crea un array de Passwords con el tamaño que tu le indiques por teclado.
- Crea un bucle que cree un objeto para cada posición del array.
- Indica también por teclado la longitud de los Passwords (antes de bucle).
- Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).

Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior). Usa este simple formato:

```
contraseña1 valor_booleano1  
contraseña2 valor_bololeano2
```

# In case of fire



1. `git commit`



2. `git push`



3. leave building