Práctica 6

Ejercicio 1.

Determine si el siguiente código es correcto. Si produce un error, observe de qué tipo es y soluciónelo.

```
class Excepcion1 extends Exception{}
class Excepcion2 extends Exception1{}
public class Test1 {
   public static void main(String[] args) {
     try {
             throw new Exception2();
     } catch(Excepcion1 e1) {
System.out.println("Se capturó la Excepción1");
     } catch( Excepcion2 e2) {
System.out.println("Se capturó la Excepción2");
     }
   }
}
```

Como esta el catch de Excepcion1 primero, el cual es el más general, ya que es superclase de Excepcion2, el segundo catch queda inalcanzable.

Ejercicio 2.

Ejecute el siguiente código. ¿Cuál es el resultado?. Elimine los comentarios y vuelva a ejecutarlo. ¿Cuál es el resultado?.

```
public class Test2 {
public int unMetodo(){
    //  try {
        System.out.println("Va a retornar 1");
        return 1;
    // } finally {
        System.out.println("Va a retornar 2");
        return 2;
    // }
  }
  public static void main(String[] args) {
     Test2 res = new Test2();
     System.out.println(res.unMetodo());
  }
}
```

Como hay un return, el segundo return es inalcanzable y no se puede ejecutar ya que da error de compilación. Si se comenta lo que esta adentro del finally comentado, se retorna 1. Si se descomenta todo lo comentado, se retorna 2, ya que el finally se ejecuta siempre, sobrescribiendo el return 1 del bloque try.

Ejercicio 3.

Ejecute el siguiente código. ¿Cuál es la salida del programa?

```
public class Test3 {
   public static void main(String[] args) {
          System.out.println("Test3");
          try {
                System.out.println("Primer try");
                try {
                      throw new Exception();
                } finally {
                      System.out.println("Finally del 2° try");
                }
          } catch (Exception e) {

                   System.out.println("Se capturó la Excepción ex del 1° Primer try");
          } finally {
                System.out.println("Finally del 1° try");
          }
    }
}
```

```
Test3
Primer try
Finally del 2º try
Se capturó la Excepción ex del 1º Primer try
Finally del 1º try
```

Ejercicio 4.

Analice el siguiente código y determine si es correcto. Si hay errores, escriba el motivo de cada uno y proponga una solución.

```
class FutbolException extends Exception{}

class Falta extends FutbolException{}

class EquipoIncompleto extends
                        FutbolException{}

class ClimaException extends Exception{}

class Lluvia extends ClimaException{}

class Mano extends Falta{}

class Partido {
  Partido() throws FutbolException{}
  void evento() throws FutbolException{}
  void jugada() throws EquipoIncompleto,
                       Falta{}
  void penal(){}
}

inteface Tormenta {
  void evento() throws Lluvia;
  void diluvio() throws Lluvia;
}
```

```
public class Encuentro extends Partido
                  implements Tormenta {

Encuentro() throws Lluvia,
                   FutbolException{..}
Encuentro (String fecha) throws Falta,
                   FutbolException{..}

void penal() throws Mano{..}
public void evento() throws Lluvia{..}
public void diluvio() throws Lluvia{..}
public void evento(){..}
void jugada() throws Mano{..}

public static void main (String[] args) {
     try {
        Encuentro enc = new Encuentro();
        enc.jugada();
     } catch(Mano e) {
     } catch(Lluvia e) {
     } catch(FutbolException e) {
     try {
        Partido par = new Encuentro();
        par.jugada();
     } catch(EquipoIncompleto e) {
     } catch(Falta e) {
     } catch(Lluvia e) {
     } catch(FutbolException e) {}
  }
}
```

void penal() throws Mano {}: penal no debe tirar ninguna excepción, o el penal de partido debe tirar una excepción superclase de Mano.

public void evento(): evento de Evento no debe tirar ninguna excepción, ya que esta sobreescribiendo dos métodos que tiran excepciones diferentes.

El nombre de los parámetros el try catch dentro del último catch debe ser distinto.

Ejercicio 5.

Analice el siguiente código:

```java
public class Suma {
    public static void main(String[] args){
      int suma=0;
      for(int i=0;i<=args.length;i++)
         suma+= Integer.parseInt(args[i]);
      System.out.print("La suma es:"+suma);
    }
}
```

a) Ejecútelo ingresando al menos 2 valores.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2
        at LaboratorioDeSoftware/practica6.ej5.Suma.main(Suma.java:7)
```

Tengo que sacar al <= del for.

b) Ahora ejecútelo ingresando: 2 3 four. ¿Qué pasó?. Solucione el problema de manera que los datos no numéricos sean impresos en la consola con un mensaje y descartados antes de ser sumados.

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "four"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:588)
        at java.base/java.lang.Integer.parseInt(Integer.java:685)
        at LaboratorioDeSoftware/practica6.ej5.Suma.main(Suma.java:7)
```

c) ¿Por qué no fue necesario capturar la excepción en el inciso a) ?

Porque todos los elementos ingresados eran validos para hacer el casteo a Integer.