1.- Excepciones (Basado en https://www.tutorialspoint.com/java/java_exceptions.htm)

Una excepción ocurre cuando durante la ejecución de un programa se produce un evento o situación inesperada. Cuando sucede una excepción la ejecución del programa se ve alterada y, si el programador no la ha previsto, el programa termina.

Una excepción puede ocurrir por muchos motivos, por ejemplo:

- Un usuario introduce datós no válidos.
- Se intenta acceder a un archivo que no existe.
- La red se interrumpe en el medio de una conexión.
- ...

Algunas de estas situaciones excepcionales se producen por errores de los usuarios, otras por errores del programador y otras durante el acceso a recursos (ficheros, red...)

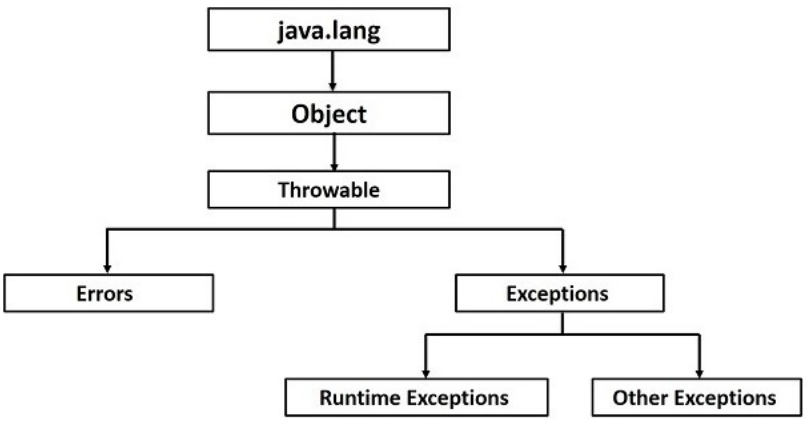
Existen 3 tipos de excepciones:

1. **Checked excepcions**: excepciones que son comprobadas por el compilador en tiempo de compilación. El programador tiene que especificar obligatoriamente qué quiere hacer cuando una excepción de este tipo sucede.

2. **Unchecked exceptions**: estas excepciones ocurren en tiempo de ejecución. Estas excepciones no pueden ser detectadas por el compilador en tiempo de compilación.

3. **Errors**. No son excepciones sino problemas que están fuera del control del programador. Los Error normalmente no se capturan en el código porque son situaciones excepcionales acerca de las que poco o nada puede hacer el programador.

Java captura las excepciones o los errores y nos los muestra a su vez como objetos. Las clases que se utilizan son:



Como se ve en la imgen, la clase base de todas las excepciones y errores es **Throwable**. Sólo las instancias de Throwable o de alguna clase que herede de ella son “lanzadas” (thrown) por la JVM o pueden ser “lanzadas” utilizando la claúsula throw.

Para poder “capturar” una excepción y decidir qué hacemos Java utiliza las palabras reservadas: **try** y **catch** de la siguiente manera:

```
try {
   // Protected code
} catch (ExceptionName e1) {
   // Catch block
}
```

Por ejemplo:

public class ExcepTest {

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

         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);  // línea 5

         ...
         ... código que no se ejecuta al producirse la excepción en línea 5.

      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Exception thrown  :" + e);
      }

      System.out.println("Out of the block");
   }
}
```

Puede darse el caso de que el mismo bloque de código lance varios tipos de excepciones. Tenemos que capturar cada tipo y decidir qué hacemos. Por ejemplo:

```
try {
   file = new FileInputStream(fileName);
   x = (byte) file.read();
} catch (FileNotFoundException f) {   // Se captura la subclase excepción
   i.printStackTrace();
   return -1;
} catch (IOException i) {     // Se captura la superclase excepción
   f.printStackTrace();
   return -1;
}
```

En este caso FileNotFoundException es una subclase de IOException. Siempre tenemos que declarar el bloque catch de las subclases antes que el de las superclases, o lo que es lo mismo, captura primero siempre las excepciones más específicas.

Si dentro de un método tenemos un bloque de código que lanza una excepción pero no queremos capturarla en el mismo método, podemos añadir la cláusula **throws** al final de la declaración del método.

```
public void miMetodo() throws IOExcepcion {
}
```

Esto hará que no necesitemos un bloque try-catch en nuestro método, pero tendremos que añadirlo en aquellos puntos de nuestro código que llamen a este método.

Una mala práctica, que debemos evitar, consiste en capturar pero ignorar una excepción, por ejemplo:

```
public void doNotIgnoreExceptions() {
        try {
                // do something
        } catch (NumberFormatException e) {
                // Aquí tenemos que hacer algo con la excepción
        }
}
```

Por último tenemos el bloque **finally**, este bloque se ejecuta siempre, tanto si se ha producido una excepción como si no. Por ejemplo:

```
public class ExcepTest {

   public static void main(String args[]) {
      int a[] = new int[2];
      try {
         System.out.println("Access element three :" + a[3]);

      } catch (ArrayIndexOutOfBoundsException e) {

         System.out.println("Exception thrown  :" + e);

      } finally {
         a[0] = 6;
         System.out.println("First element value: " + a[0]);
         System.out.println("The finally statement is executed");
      }
   }
}
```