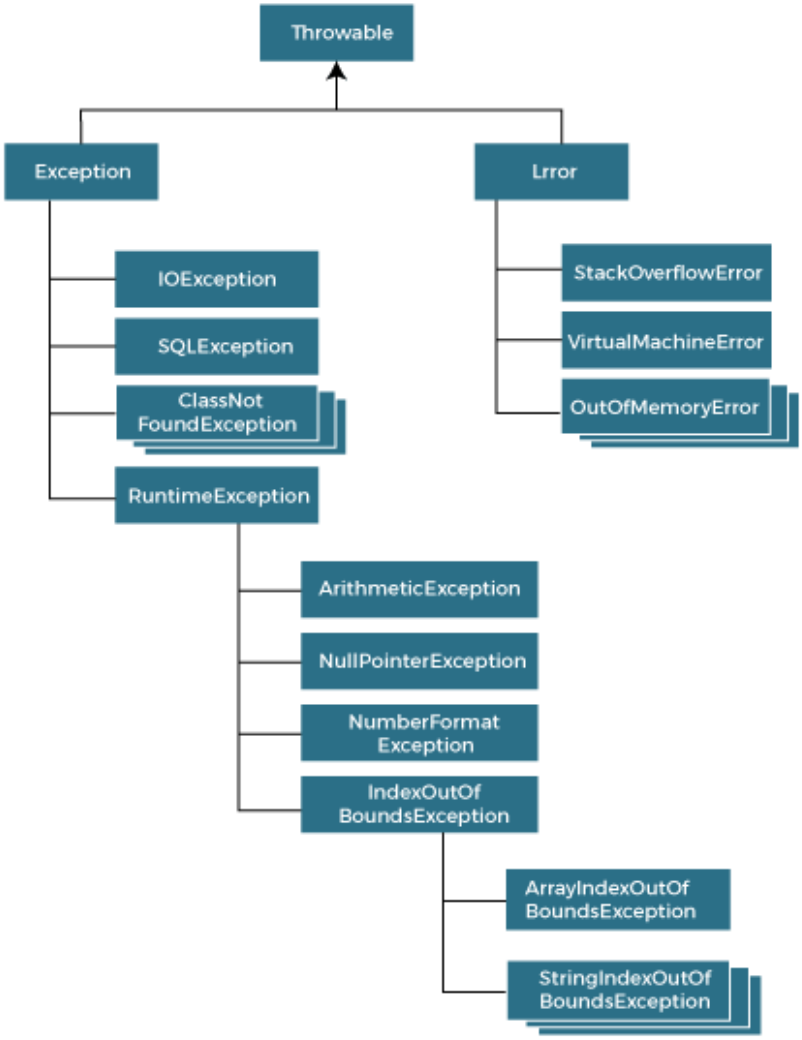
Ayudantía 9

Alexander Inostroza – alexander.inostroza@usm.cl

Exceptions

Las clases **InputStream** y **OutputStream** son clases abstractas del paquete **java.io**



Sr.No.	Method & Description
1	**public String getMessage()** Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor.
2	**public Throwable getCause()** Returns the cause of the exception as represented by a Throwable object.
3	**public String toString()** Returns the name of the class concatenated with the result of getMessage().
4	**public void printStackTrace()** Prints the result of toString() along with the stack trace to System.err, the error output stream.
5	**public StackTraceElement [] getStackTrace()** Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.
6	**public Throwable fillInStackTrace()** Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

```
class InsufficientFundsException extends Exception {
    private double amount;

    public InsufficientFundsException(double amount) {
       this.amount = amount;
    }

    public double getAmount() {
       return amount;
    }
 }
```

```
class CheckingAccount {
    private double balance;
    private int number;

    public CheckingAccount(int number) {
       this.number = number;
    }

    public void deposit(double amount) {
       balance += amount;
    }

    public void withdraw(double amount) throws InsufficientFundsException {
       if(amount <= balance) {
          balance -= amount;
       }else {
          double needs = amount - balance;
          throw new InsufficientFundsException(needs);
       }
    }

    public double getBalance() {
       return balance;
    }

    public int getNumber() {
       return number;
    }
 }
```

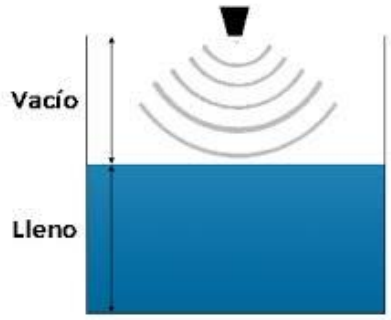
```java
public class BankDemo {

   public static void main(String [] args) {
      CheckingAccount c = new CheckingAccount(101);
      System.out.println("Depositando $500...");
      c.deposit(500.00);

      try {
         System.out.println("\nRetirando $100...");
         c.withdraw(100.00);
         System.out.println("\nRetirando $600...");
         c.withdraw(600.00);
      } catch (InsufficientFundsException e) {
         System.out.println("Para realizar la operación te faltan $" + e.getAmount());
         e.printStackTrace();
      }
   }
}
```

Sensor de tanque

Se tiene un tanque de 10 m de altura, el cual tiene un flujo de agua entrante y un flujo de agua saliente, ninguno de los dos es constante. De acuerdo al flujo máximo entrante y saliente, el nivel del agua sólo puede variar como máximo 2 m. El sensor entrega muchas mediciones falsas, de las que debe preocuparse en software.
*__El código del sensor y del tanque están en el repositorio.__



a) Escriba un programa main que le pregunte constantemente a un sensor, y si se entregara un valor no válido, termine con un WrongMeasurementException. Puede asumir que el primer valor medido es válido, siempre y cuando sea coherente.
b) Modifique el programa main que escribíó para manejar la exception lanzada. Sólo debe tomar las mediciones que sean válidas. Si el estanque se llenase, debe lanzar una alerta por consola, para que los operadores puedan cerrar el flujo de agua entrante; después de la alerta puede terminar el programa, asumiendo que los operadores van a reiniciar el software.