Trabajo Práctico N° 2

1) La clase A define un método denominado **metodoA**, indique:

a) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción *MioException*?
b) ¿Qué sentencias y en qué orden se ejecutan si no se produce la excepción *MioException*?
c) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción *TuException*?

```
public class A {

        public A(){}
        public void metodoA()
        {
                sentencia_1;
                sentencia_2;
                try{
                        sentencia_3;
                        sentencia_4;
                }
                catch(MioException e)
                {
                        sentencia_6;
                }
                sentencia_5;
        }
}
```

2) Dado el método **metodoB** de la clase B, indique:

¿qué sentencias y en qué orden se ejecutan si se produce la excepción *MioException*?
¿qué sentencias y en qué orden se ejecutan si no se produce la excepción *MioException*?
¿qué sentencias y en qué orden se ejecutan si se produce la excepción *TuException*?

```
class B {
        void metodoB() {
                sentencia_b1
                try {
                        sentencia_b2
                } catch (MioException e)
                {
                        sentencia_b3
                }
                finally
                        sentencia_b4
        }
}
```

3) Dado el método **metodoC** de la clase C, indique:

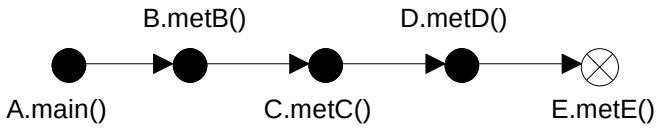
a) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción MioException?
b) ¿Qué sentencias y en qué orden se ejecutan si no se produce la excepción MioException?
c) ¿Qué sentencias y en qué orden se ejecutan si se produce la excepción TuException?

```
public class C {
        public C(){}
        public void metodoC() throws OtraException
        {
      sentencia_c1;
      try
      {
          sentencia_c2;
          sentencia_c3;
      }
      catch (MioException e)
      {
          sentencia_c4;
      }
          catch (TuException e)
          {  sentencia_c5;
                 throw new OtraException();
      }
      finally
```

```
        {
          sentencia_c6
        }
          }
}
```

4. En el método **metE** de la clase E se produce una excepción, al atrapar la excepción se ejecuta el método printStackTrace(), qué salida tiene el método si:

a) la excepción es atrapada en D.metD()
b) la excepción es atrapada en C.metC()
c) la excepción es atrapada en B.metB()
d) la excepción es atrapada en A.main()



5. Sistema de Archivos

Se debe desarrollar una aplicación para gestionar los archivos de una computadora. Como ya se sabe en un sistema de archivos hay diferentes clases de archivos. Todos los archivos se caracterizan por su nombre, extensión, tamaño (en bytes), fecha de la última modificación, permiso de escritura y permiso de lectura. El dato correspondiente a los permisos sólo admite dos valores, es decir si se puede o no leer/escribir. Hay archivos binarios y de texto.
El sistema de Archivos sobre el que va a trabajar la aplicación, se puede describir con las siguientes características: identificador del disco duro, capacidad del disco duro y lista de ficheros. Usar la clase Vector de java.util para implementar la lista de archivos.
Se pide:

a) Diseñar el diagrama de clases que represente el problema que se han descrito en el enunciado.
b) Implemente las clases del diagrama del punto A que da solución al problema utilice los conceptos del TP N°1.

La clase SistemaDeArchivos muestra por pantalla todos sus archivos, pero esta operación puede fallar. Es por ello que deben manejarse adecuadamente las excepciones que puedan surgir, Error1, Error2. Para ello deberá tenerse presente lo siguiente:

- En caso de que ocurra el Error1 (Si se muestra un archivo de tipo Binario) se indicará con un mensaje y continuará.
- Cuando ocurra el Error2 (Si no se puede seguir mostrando archivos) se indicará con otro mensaje y se detendrá la ejecución.

c) Agregue al diagrama de clases del punto A las clases para manejar los errores.
d) Implemente las clases nuevas.

El siguiente código fuente describe a grandes rangos la clase SistemaDeArchivos

```
import...
public class SistemaDeArchivo
{
        /*ATRIBUTOS*/
        Vector archivos;

        public SistemaDeArchivo(){}
        public void agregarArchivo(Archivo f){}
        public boolean eliminarArchivo(String id){}
        public void listarArchivo()
        {
                /*por cada archivo f en archivos*/
                InterfazGrafica.mostrar(f);
        }
```

```
}
```

La clase InterfazGrafica es una clase que da soporte a SistemaDeArchivo para listar sus archivos y presenta la siguiente estructura.

InterfazGrafica es una clase que tiene los siguientes métodos estáticos:

```
public class InterfazGrafica {
  /*otras cosas de la clase*/

  public static void mostrar (Archivo f) throws Error1, Error2{
  // Muestra un archivo por la pantalla
  // Lanza la excepción Error1 cuando el archivo que se pretende
mostrar es binario
  // Lanza la excepción Error2 cuando no puede seguir mostrando
archivos
  ...
  }
  public static void mensajeError (String s) {
  ...
  }
}
```

e) Implementar el método listarArchivos de SistemaDeArchivo.
f) Implementar la clase InterfazGrafica para poder complentar la implementación del punto E.
g) Escriba un programa en el que pruebe todas las funcionalidades.

6. La **impresora** es un periférico que puede fallar (cuando se ordena la impresión de un documento) ante las siguientes condiciones:

a) fuera de línea
b) no conectada
c) falta papel
d) falta tinta

La clase **Printer** tiene las siguientes operaciones básicas:

```
cargarTinta (double);
cargarPapel(int);
conectar();
desconectar();
on_line () ;
of_line () ;
print(Document);// esta operación puede fallar si falta papel o tinta o no esta
                //conectada o está fuera de línea.
```

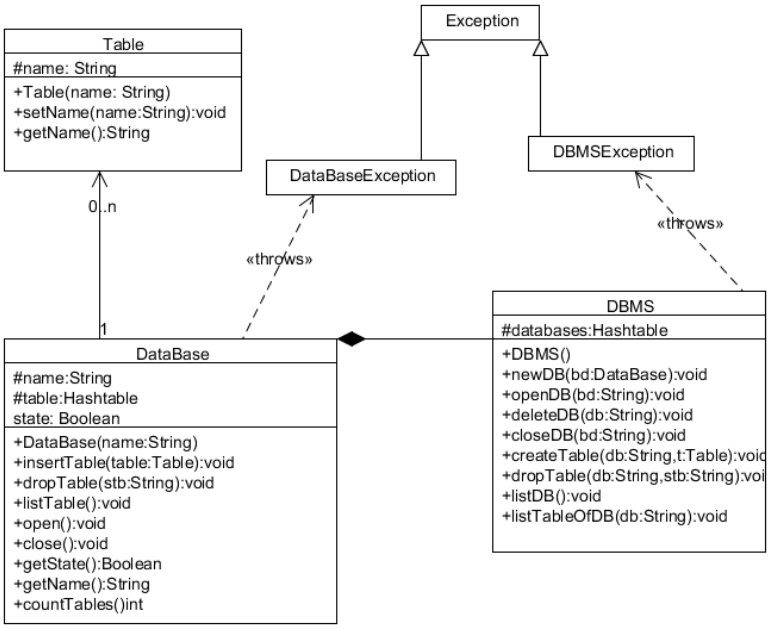
La clase **Document** (cantidad de hojas) representa el documento a imprimir.
La clase **PrinterException** representa todas las posibles excepciones de la clase **Printer**.

Tener en cuenta:

- Al enviar a imprimir un documento se decrementa la cantidad de hojas y tinta correspondiente de la impresora.
- Si la cantidad de tinta o papel son mayores que 0 pero son insuficientes para imprimir completamente un documento, se imprimirán las hojas hasta que la cantidad de papel y/o tinta alcance y luego se produce la excepción.
- Si la cantidad de tinta y/o papel es/son igual a 0 al enviar a imprimir un documento, no se imprime nada y solo se produce la excepción.
- Para poder imprimir la impresora debe estar conectada y en línea.
- Una carga de tinta imprime 125 hojas. Es decir una hoja insume 1/125 carga de tinta.
- El usuario puede cargar más de una carga de tinta. (hay cartuchos de distintos tamaños).

Codificar un programa que utilice la impresora (use un menú).

7. Un gestor de base de datos (DBMS) es una aplicación que administra bases de datos. Una base de datos es un conjunto de tablas. Cada base de datos tiene su nombre, como así cada una de las tablas. Las operaciones del DBMS son: crear, eliminar y consultar, abrir, cerrar las bases de datos disponibles. A cada base de datos se le pueden agregar, remover y listar sus tablas (estas operaciones solo se pueden realizar si la base de datos existe y está abierta).



a) Crear la clase DataBaseManagementSystem, DataBase y Table teniendo en cuenta que:

Operación	Condición de falla
CreateDB (db)	Falla si db ya existe
OpenDB(db)	Falla si db no existe
CloseDB(db)	Falla si db no existe
CreateTable (db, t)	Falla si db no está abierta o t ya existe en db.
RemoveDB(db)	Falla si db no está abierta.
QueryTable(db)	Falla si db no está abierta
RemoveTable(db, t)	Falla si db no está abierta o t no existe en db.

Usar como contenedor para las bases de datos que gestiona DataBaseManagementSystem la clase Hashtable (del API de Java). Idem para las tablas que gestiona DataBase.

b) Crear las clases DBException y TableException para manejar las excepciones.
c) Pruebe las clases con un menú con las siguientes opciones.
   - Crear Base de Datos
   - Eliminar Base de Datos
   - Abrir Base de Datos
   - Cerrar Base de Datos
   - Agregar una Tabla a una Base de Datos
   - Eliminar una Tabla de una Base de Datos
   - Consultar las Tablas de una Base de Datos.