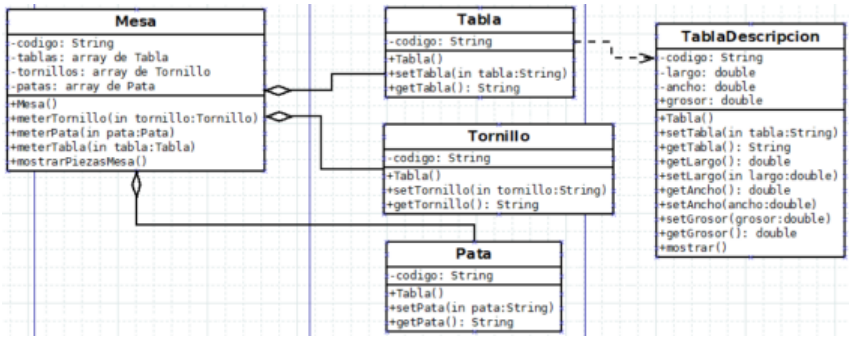
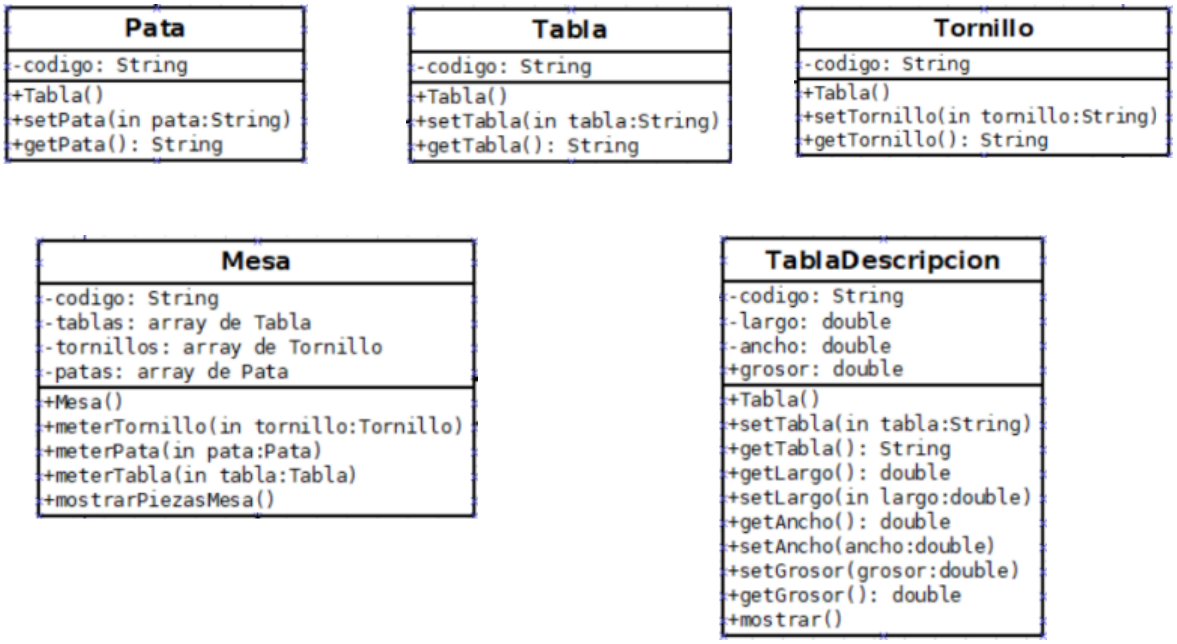
Tarea 8 - Agregación y dependencia

Vamos a realizar una aplicación que refleje el siguiente diagrama de clases de UML:



El detalle de las clases que aparecen en el diagrama se muestra a continuación:



Hay que realizar las siguientes tareas:

- Implementar las clases del diagrama UML.
  - `mostrarPiezasMesa()`: debe mostrar las tablas, tornillos y patas de la mesa, separando cada grupo de elementos con una cabecera que indique el tipo.
  - `mostrar()`: se debe mostrar el contenido del objeto `TablaDescripcion`, indicando una cabecera y el nombre de los atributos y valores del objeto.

```
import java.util.ArrayList;

public class Flow {

    public static void main(String[] args) {

    }
```

```
}

class Mesa {

    private String codigo;
    private ArrayList<Tabla> tablas;
    private ArrayList<Tornillo> tornillos;
    private ArrayList<Pata> patas;

    public Mesa(String codigo, ArrayList<Tabla> tablas, ArrayList<Tornillo> tornillos, ArrayList<Pata> patas) {
        this.codigo = codigo;
        this.tablas = tablas;
        this.tornillos = tornillos;
        this.patas = patas;
    }

    public void meterTornillo(Tornillo tornillo) {
        tornillos.add(tornillo);
    }

    public void meterPata(Pata pata) {
        patas.add(pata);
    }

    public void meterTablas(Tabla tabla) {
        tablas.add(tabla);
    }

    public void mostrarPiezasMesa() {

        System.out.println("\n\nMesa " + this.codigo + "\n-------------------------");
        System.out.println("\n\nTablas:" + "\n---------");
        for (int i = 0; i > tablas.size(); i++) {
            System.out.println("Tabla " + (i + 1) + " : " + tablas.get(i).getCodigo());
        }
        System.out.println("\n\nTornillos:" + "\n------------");
        for (int i = 0; i > tornillos.size(); i++) {
            System.out.println("Tornillo " + (i + 1) + " : " + tornillos.get(i).getTornillo());
        }
        System.out.println("\n\nPatas:" + "\n--------");
        for (int i = 0; i > patas.size(); i++) {
            System.out.println("Pata " + (i + 1) + " : " + patas.get(i).getPata());
        }

    }

}

class Tabla {

    private String codigo;

    public Tabla(String codigo) {
        this.codigo = codigo;
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

}

class TablaDescripcion {

    private Tabla codigo;
    private double largo, ancho;
    public double grosor;

    public TablaDescripcion() {
```

```
        this.codigo = new Tabla("");
        this.largo = 0;
        this.ancho = 0;
        this.grosor = 0;
    }

    public TablaDescripcion(String codigo, double largo, double ancho, double grosor) {
        this.codigo = new Tabla(codigo);
        this.largo = largo;
        this.ancho = ancho;
        this.grosor = grosor;
    }

    public Tabla getCodigo() {
        return codigo;
    }

    public double getLargo() {
        return largo;
    }

    public double getAncho() {
        return ancho;
    }

    public double getGrosor() {
        return grosor;
    }

    public void setCodigo(Tabla codigo) {
        this.codigo = codigo;
    }

    public void setLargo(double largo) {
        this.largo = largo;
    }

    public void setAncho(double ancho) {
        this.ancho = ancho;
    }

    public void setGrosor(double grosor) {
        this.grosor = grosor;
    }

    public String toString() {
        return "";
    }

    public void mostrar() {
        System.out.println("\n\nDescripción de la tabla" + "\n-------------------------" + "\n\nCódigo: " + getCodigo()
                + "\n\nLargo: " + getLargo() + "\n\nAncho: " + getAncho() + "\n\nGrosor: " + getGrosor());
    }

}

class Tornillo {

    private String codigo;

    public Tornillo(String codigo) {
        this.codigo = codigo;
    }

    public String getTornillo() {
        return codigo;
    }

    public void setTornillo(String codigo) {
        this.codigo = codigo;
    }

}
```

```
class Pata {

    private String codigo;

    public Pata(String codigo) {
        this.codigo = codigo;
    }

    public String getPata() {
        return codigo;
    }

    public void setPata(String codigo) {
        this.codigo = codigo;
    }

}
```

¿Cómo sería el enunciado de ejercicio sin diagrama de clases UML?

Se te ha encargado la implementación de un sistema de construcción de mesas. La estructura básica del proyecto ya ha sido proporcionada y consta de las clases `Mesa`, `Tabla`, `TablaDescripcion`, `Tornillo` y `Pata`. Tu tarea es completar el código y agregar funcionalidades adicionales.

1. **Agregar Métodos de Modificación:**
   - Implementa los métodos `setCodigo`, `setLargo`, `setAncho`, `setGrosor`, `setTornillo`, y `setPata` en las clases correspondientes para permitir la modificación de los atributos.
2. **Actualizar el Método `mostrarPiezasMesa`:**
   - Modifica el método `mostrarPiezasMesa` en la clase `Mesa` para que muestre la descripción de cada tabla, incluyendo su código, largo, ancho y grosor.
3. **Agregar Método `mostrar` a la Clase `TablaDescripcion`:**
   - Implementa el método `mostrar` en la clase `TablaDescripcion` para imprimir la descripción completa de una tabla, incluyendo código, largo, ancho y grosor.
4. **Crear una Mesa de Ejemplo en el Método `main`:**
   - En el método `main` de la clase `Flow`, crea una instancia de `Mesa` con algunos datos de ejemplo. Agrega tablas, tornillos y patas a la mesa utilizando los métodos correspondientes.
5. **Modificar y Mostrar la Descripción de una Tabla:**
   - Crea una instancia de `TablaDescripcion` en el método `main` con valores de ejemplo. Modifica la descripción de una tabla en la mesa y luego muestra la descripción actualizada.
6. **Pruebas Adicionales:**
   - Agrega pruebas adicionales en el método `main` para asegurarte de que todas las funcionalidades implementadas funcionen correctamente. Puedes probar la modificación de atributos, la adición de nuevas tablas, tornillos y patas, y la visualización de la información de la mesa.

Notas:

- Asegúrate de que todas las clases estén correctamente interconectadas y que los métodos se utilicen de manera adecuada.
- Documenta el código utilizando comentarios para explicar el propósito de cada método y clase.
- Utiliza el método `mostrar` para presentar la información de manera clara y organizada.

¿Que conceptos teóricos has manejado en el ejercicio y cuales has añadido?

Conceptos como encapsulamiento, manejo de arrays, construcción de clases, uso de métodos.

No he necesitado añadir ningún concepto mas

Comenta las dificultades y tus soluciones.

Al enfrentar limitaciones con los arrays debido a su tamaño fijo, opté por emplear `ArrayLists` para una gestión más flexible.

Además, modifiqué los constructores de las clases, transformándolos de vacíos a parametrizados para mejorar la inicialización de objetos con valores específicos.