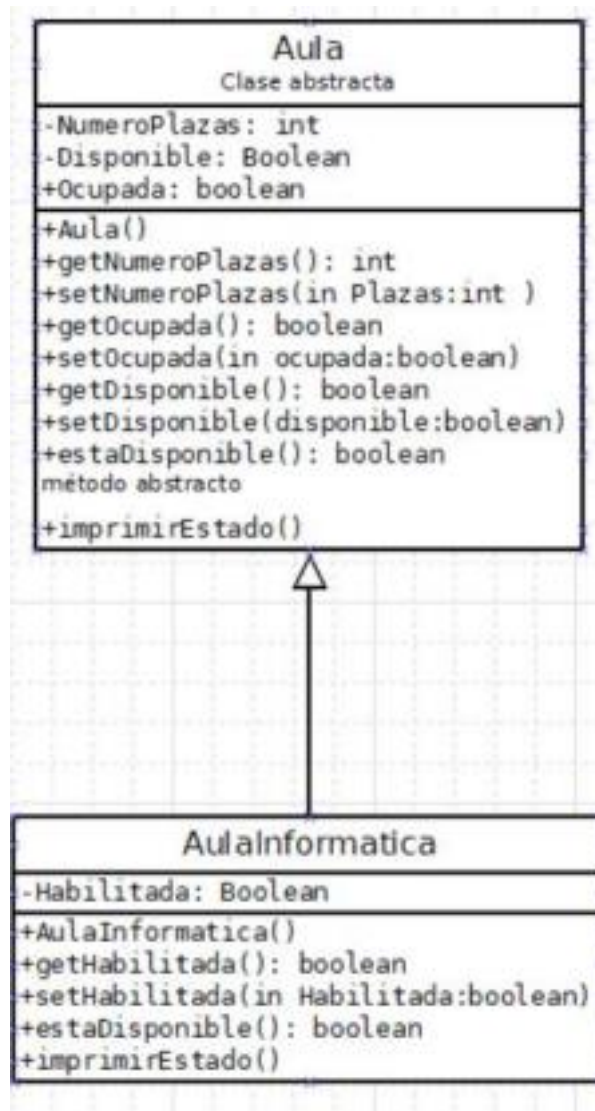


Ejercicio. Overriding

Tenemos la siguiente relación de herencia, representada mediante un diagrama UML:



Hay que realizar las siguientes tareas:

- Implementar las clases:
 - Todos los datos del enunciado se codificarán según lo mostrado en el diagrama UML.
 - El método imprimirEstado de la clase Aula muestra un mensaje por consola indicando si el aula está Disponible o no. La condición de disponible es: NumeroPlazas > 0 y Ocupada es falso.
 - El método imprimirEstado de la clase AulaInformatica muestra un mensaje por consola indicando si el aula es Apta o no. La condición para que el aula sea Apta es: NumeroPlazas > 0 y Ocupada es falso y Habilitada es verdad.
 - Los valores por defecto de los atributos son : 0 y falso.
- Implementar el método estadiponible() en la clase AulaInformatica.
- Redefinir el método imprimirEstado() en la clase AulaInformatica.
- Añadir una clase, llamada Flow, que tenga el método “main”. Implementar lo siguiente:
 - Crear un objeto de tipo AulaInformatica.
 - Cambiar el número de plazas a 24.
 - Cambiar el valor del atributo “Habilitada” a true.
 - Mostrar el estado del aula de informática.
 - Cambiar el valor del atributo “Ocupada” a true.
 - Mostrar otra vez el estado del aula de informática.

Clase Aula:

```
public abstract class Aula {
    // Atendiendo al diagrama UML me fijo en la visibilidad y en los metodos
    private int numPlazas;
    private boolean disponible;
    public boolean ocupada;
    public Aula() {
        this.numPlazas = 0;
        this.disponible = false;
        this.ocupada = false;
    }
    public int getNumPlazas() {
        return numPlazas;
    }
    public void setNumPlazas(int numPlazas) {
        this.numPlazas = numPlazas;
    }
    public boolean isDisponible() {
        if (numPlazas > 0 && ocupada == false) {
            disponible = true;
        } else {
            disponible = false;
        }
        return disponible;
    }
    public void setDisponible(boolean disponible) {
        this.disponible = disponible;
    }
    public boolean isOcupada() {
        return ocupada;
    }
    public void setOcupada(boolean ocupada) {
        this.ocupada = ocupada;
    }
    public abstract void estaDisponible();

    public void imprimirEstado() {
        if(numPlazas>0 && ocupada==false) {
            disponible=true;
        }else {
            disponible=false;
        }
    }
}
```

Clase AulaInformática

```
public class AulaInformatica extends Aula {
    private boolean Habilitada;
    public AulaInformatica() {
        super();
    }
    public boolean isHabilitada() {
        return Habilitada;
    }
    public void setHabilitada(boolean habilitada) {
        Habilitada = habilitada;
    }
    @Override
    public void estaDisponible() {
        // TODO Auto-generated method stub
        if (getNumPlazas() > 0 && !isOcupada() && isHabilitada()) {
            System.out.println("El aula de informática está disponible.");
        } else {
            System.out.println("El aula de informática NO está disponible.");
        }
    }
    public void imprimirEstado() {
        if (getNumPlazas() > 0 && !isOcupada() && isHabilitada()) {
            System.out.println("El aula de informática es APTA");
        } else {
            System.out.println("El aula de informática NO es APTA");
        }
    }
}
```

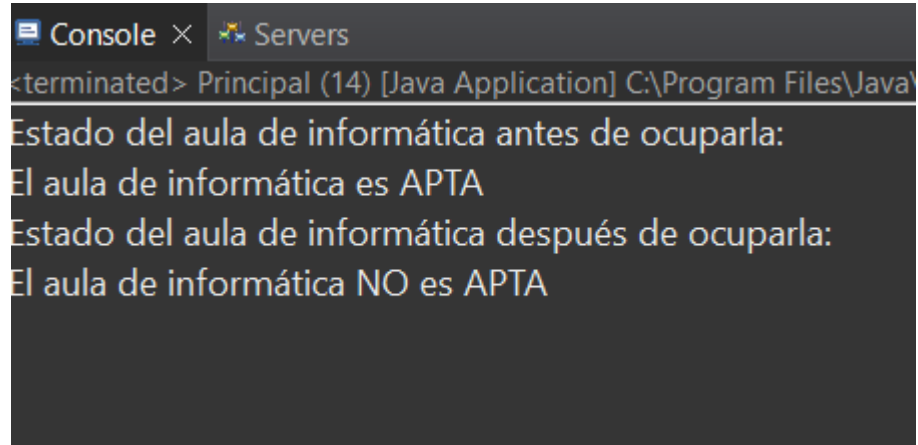
Clase Principal(Main):

```
public class Principal {
    public static void main(String[] args) {
        AulaInformatica infor = new AulaInformatica();
        //llamando a los setter lo que hago es asignarle los valores que me piden el
        enunciado porque por defecto ya los tengo predefinidos
        //en el constructor Aula.

        infor.setNumPlazas(24); // Cambia el número de plazas a 24
        infor.setHabilitada(true); // Cambia Habilitada a true
        System.out.println("Estado del aula de informática antes de ocuparla.");
    }
}
```

```
        infor.imprimirEstado();// aqui el estado es Apta  
        infor.setOcupada(true); // Cambia Ocupada a true  
        System.out.println("Estado del aula de informática después de ocuparla:");  
        infor.imprimirEstado();  
    }  
}
```

La lectura por consola:



```
Console × Servers  
<terminated> Principal (14) [Java Application] C:\Program Files\Java\jre-  
Estado del aula de informática antes de ocuparla:  
El aula de informática es APTA  
Estado del aula de informática después de ocuparla:  
El aula de informática NO es APTA
```

¿Qué dificultades has encontrado? ¿Cómo lo has solucionado? ¿Has obtenido alguna conclusión?

Al principio en la clase Flow(Principal en nuestro caso) siempre nos aparecía que el aula de informática no era apta.

Entonces lo que hicimos fue revisar paso a paso el diagrama UML, entonces escribimos tal cual los métodos, con su visibilidad, las clases abstractas e implementamos el código y funcionó correctamente.

La conclusión que sacamos es que conocer los diagramas UML es muy importante a la hora de implementar el código, ya que aparecen los métodos, la visibilidad, atributos. permitiendo realizar el código paso a paso y sin perderse. También la importancia de las clases abstractas, que es como una plantilla base para otras clases que extienden de ella. En este caso creamos un método abstracto que luego le vamos a implementar en la clase que lo hereda que es AulaInformática.

Para ello usamos el override en El método estaDisponible cambiándole el cuerpo o body con el contenido que necesitábamos.

Tarea realizada por: Marta Tirador e Irene Verdeja.