



INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo



Unidad de Aprendizaje:

Programación Orientada a Objetos

Grupo:

2CV8

Profesor:

Daniel Cruz García

Práctica 2

“Relación entre clases”

Alumnos:

Luciano Espina Melisa

Dávila García Rivas Emiliano

Ramos Mesas Edgar Alaín

Fecha de entrega:

09/03/2018

Marco Teórico

Una clase se ve como una plantilla para construir objetos; eventualmente nos daremos cuenta de que dichas plantillas pueden necesitar de otras plantillas para generar dichos objetos de alguna u otra manera. Si nosotros nos encontramos con la parte de diseño del sistema entonces es necesario conocer las siguientes relaciones y saber diferenciarlas: [1]

Asociación:

Indica cuando una clase está ligada a otra, esto es que, una primera clase necesita de cierta información de la segunda clase, pero su existencia no se ve afectada si una de estas clases no existe. Se hace uso de la cardinalidad, que indica la cantidad de objetos que podrán ser utilizados. [1]

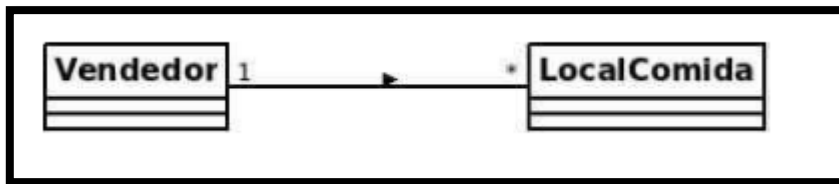


Imagen 1

La imagen 1 muestra la asociación entre clases porque el vendedor puede tener cero locales y estar en el sistema hasta que tenga los recursos para crear uno.

Dependencia

Es una relación que indica cuando una clase depende de otra clase para funcionar. [1]

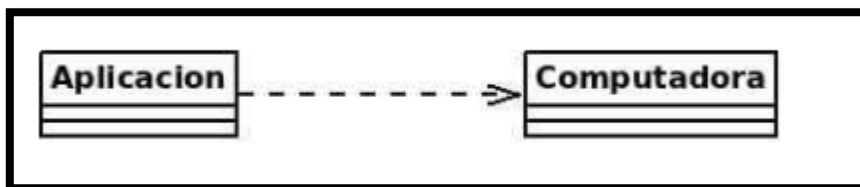


Imagen 2

En la imagen 2 se muestra la asociación de esas clases porque sería imposible correr una aplicación en un navegador que no existe.

Agregación

Es utilizada cuando una clase se compone de otras clases, aunque si se quita alguna de ellas, la primera seguirá funcionando normalmente. [1]

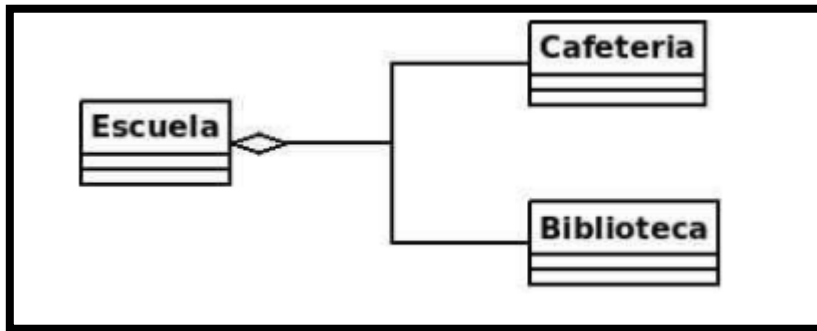


Imagen 3

En la imagen 3 hay agregación entre las clases porque si se quita, por ejemplo, la cafetería seguirá siendo escuela

Composición

Es muy similar a la relación anterior, con la diferencia de que si una clase hace falta entonces la principal se verá afectada. [1]

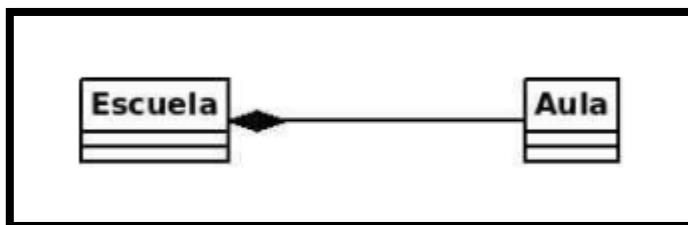


Imagen 3

En la imagen 3 hay composición entre la clase aula y escuela, ya que el aula compone a la escuela y si no estuviera no existiría escuela.

Análisis del problema

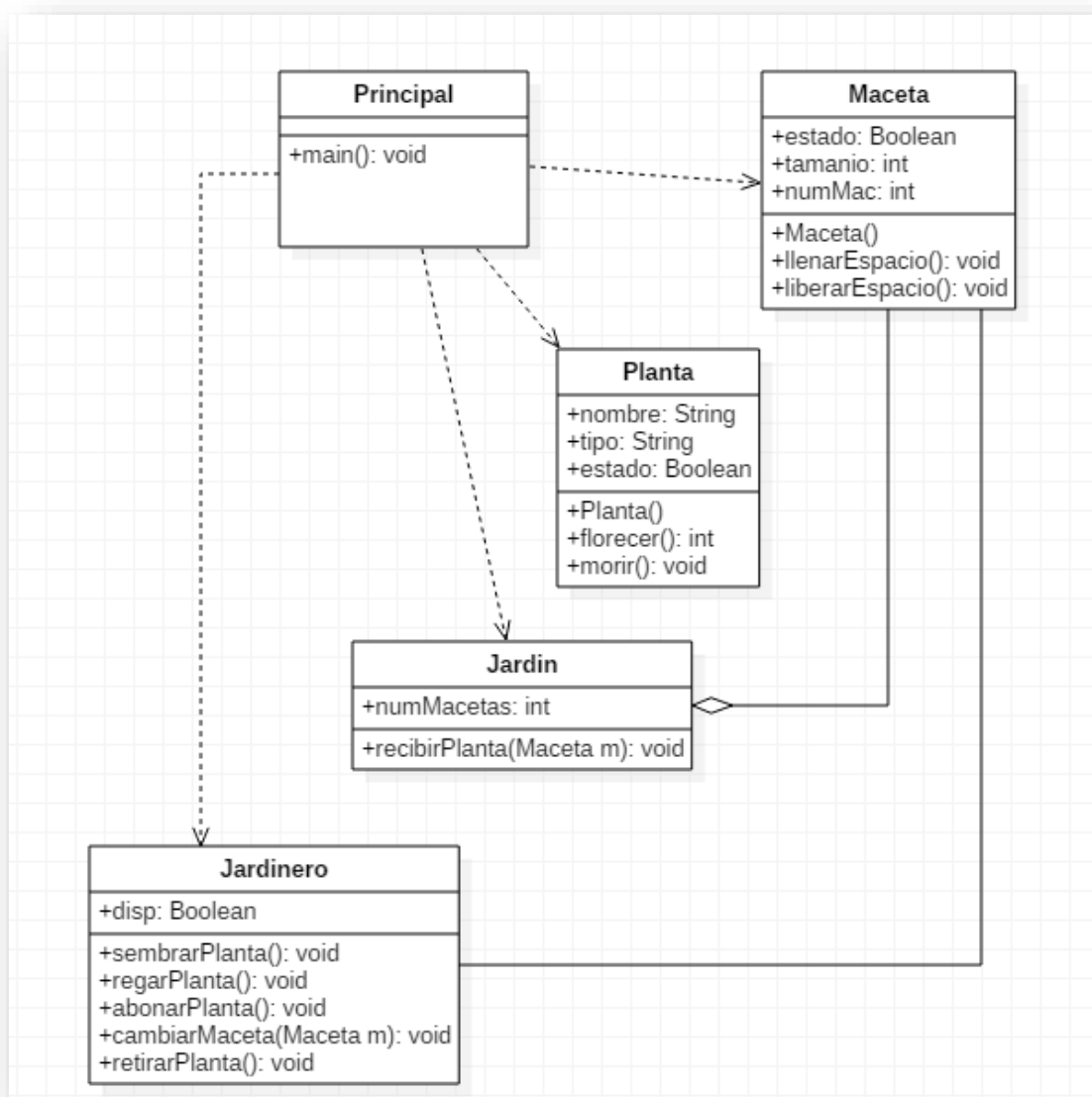
Problema que resolver:

Se desea construir un jardín en el cual se pueda interactuar para agregar plantas cuidar de ellas, incluso poder cambiarla de sitio.

Identificar los objetos que ayudarán a resolverlo

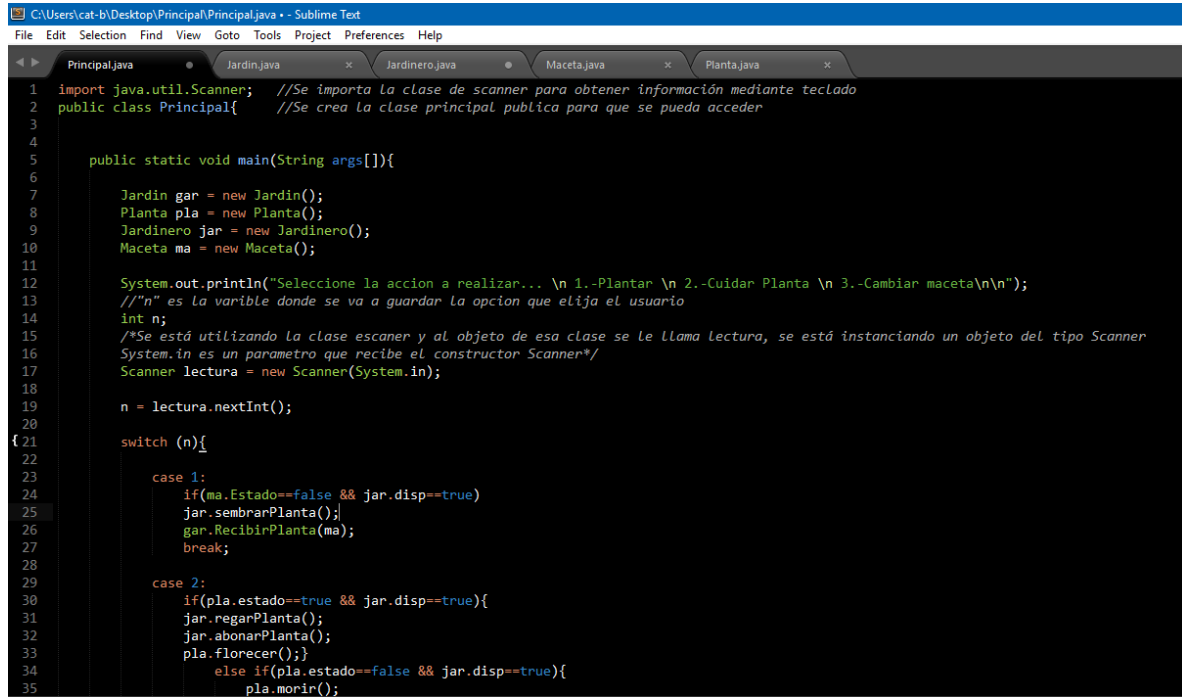
Con la planta, maceta, jardinero y el jardín, se puede resolver el problema, ya que los elementos planta, maceta y jardinero interactuarán en el jardín. Por ejemplo, el jardinero puede agregar la maceta al jardín, agregar una planta a la maceta y cuidarla si el usuario así lo desea.

Diagramas de clase

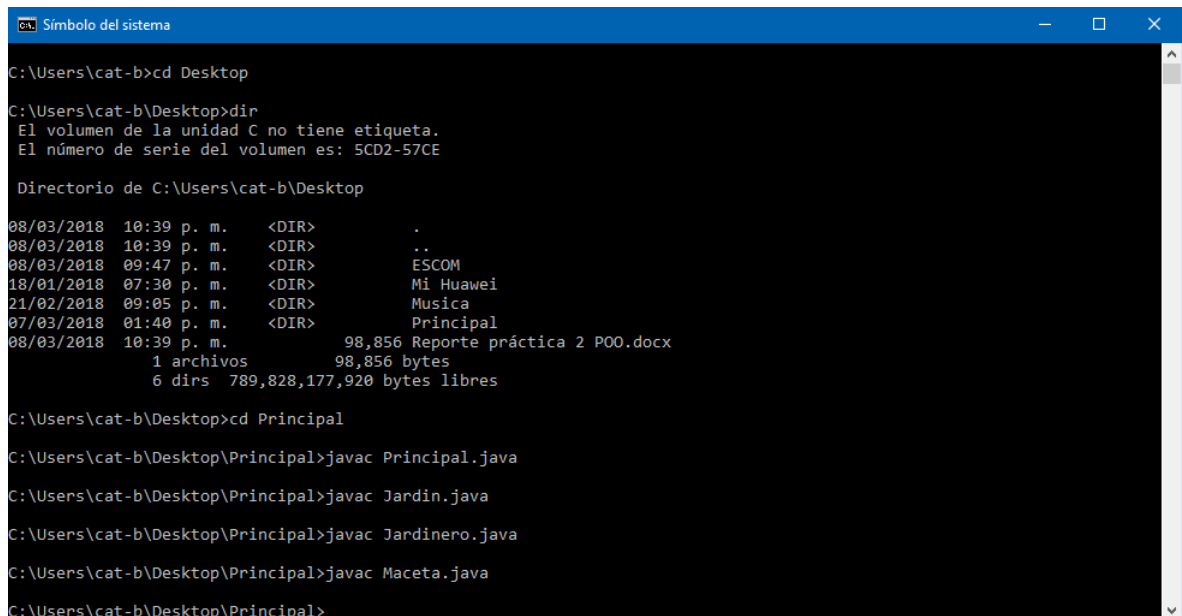


Pruebas y resultados:

En esta imagen se muestra la clase principal del programa



```
1 import java.util.Scanner; //Se importa la clase de scanner para obtener información mediante teclado
2 public class Principal{ //Se crea la clase principal publica para que se pueda acceder
3
4
5     public static void main(String args[]){
6
7         Jardin gar = new Jardin();
8         Planta pla = new Planta();
9         Jardinero jar = new Jardinero();
10        Maceta ma = new Maceta();
11
12        System.out.println("Seleccione la accion a realizar... \n 1.-Plantar \n 2.-Cuidar Planta \n 3.-Cambiar maceta\n\n");
13        // "n" es la variable donde se va a guardar la opcion que elija el usuario
14        int n;
15        /*Se está utilizando la clase scanner y al objeto de esa clase se le llama lectura, se está instanciando un objeto del tipo Scanner
16        System.in es un parametro que recibe el constructor Scanner*/
17        Scanner lectura = new Scanner(System.in);
18
19        n = lectura.nextInt();
20
21        switch (n){
22
23            case 1:
24                if(ma.Estado==false && jar.disp==true)
25                    jar.sembrarPlanta();
26                    gar.RecibirPlanta(ma);
27                    break;
28
29            case 2:
30                if(pla.estado==true && jar.disp==true){
31                    jar.regarPlanta();
32                    jar.abonarPlanta();
33                    pla.floreecer();
34                } else if(pla.estado==false && jar.disp==true){
35                    pla.morir();
36                }
```



```
C:\Users\cat-b>cd Desktop
C:\Users\cat-b\Desktop>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 5CD2-57CE

Directorio de C:\Users\cat-b\Desktop

08/03/2018  10:39 p. m.      <DIR>          .
08/03/2018  10:39 p. m.      <DIR>          ..
08/03/2018  09:47 p. m.      <DIR>          ESCOM
18/01/2018  07:30 p. m.      <DIR>          Mi Huawei
21/02/2018  09:05 p. m.      <DIR>          Musica
07/03/2018  01:40 p. m.      <DIR>          Principal
08/03/2018  10:39 p. m.             98,856 Reporte práctica 2 P00.docx
                   1 archivos             98,856 bytes
                   6 dirs  789,828,177,920 bytes libres

C:\Users\cat-b\Desktop>cd Principal
C:\Users\cat-b\Desktop\Principal>javac Principal.java
C:\Users\cat-b\Desktop\Principal>javac Jardin.java
C:\Users\cat-b\Desktop\Principal>javac Jardinero.java
C:\Users\cat-b\Desktop\Principal>javac Maceta.java
C:\Users\cat-b\Desktop\Principal>
```

```
CS: Símbolo del sistema - java Principal
Directorio de C:\Users\cat-b\Desktop

08/03/2018 10:39 p. m. <DIR> .
08/03/2018 10:39 p. m. <DIR> ..
08/03/2018 09:47 p. m. <DIR> ESCOM
18/01/2018 07:30 p. m. <DIR> Mi Huawei
21/02/2018 09:05 p. m. <DIR> Musica
07/03/2018 01:40 p. m. <DIR> Principal
08/03/2018 10:39 p. m. 98,856 Reporte práctica 2 P00.docx
1 archivos 98,856 bytes
6 dirs 789,828,177,920 bytes libres

C:\Users\cat-b\Desktop>cd Principal
C:\Users\cat-b\Desktop\Principal>javac Principal.java
C:\Users\cat-b\Desktop\Principal>javac Jardin.java
C:\Users\cat-b\Desktop\Principal>javac Jardinero.java
C:\Users\cat-b\Desktop\Principal>javac Maceta.java
C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta
```

```
CS: Símbolo del sistema
08/03/2018 09:47 p. m. <DIR> ESCOM
18/01/2018 07:30 p. m. <DIR> Mi Huawei
21/02/2018 09:05 p. m. <DIR> Musica
07/03/2018 01:40 p. m. <DIR> Principal
08/03/2018 10:39 p. m. 98,856 Reporte práctica 2 P00.docx
1 archivos 98,856 bytes
6 dirs 789,828,177,920 bytes libres

C:\Users\cat-b\Desktop>cd Principal
C:\Users\cat-b\Desktop\Principal>javac Principal.java
C:\Users\cat-b\Desktop\Principal>javac Jardin.java
C:\Users\cat-b\Desktop\Principal>javac Jardinero.java
C:\Users\cat-b\Desktop\Principal>javac Maceta.java
C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta

1
La planta ha sido sembrada en la maceta
La maceta 1 esta ocupada
C:\Users\cat-b\Desktop\Principal>
```

```
Símbolo del sistema
C:\Users\cat-b\Desktop\Principal>javac Jardinero.java
C:\Users\cat-b\Desktop\Principal>javac Maceta.java
C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta

1
La planta ha sido sembrada en la maceta
La maceta 1 esta ocupada

C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta

2
La planta ha sido regada
La planta ha sido abonada
La planta esta floreciendo... Siga cuidandola... :)

C:\Users\cat-b\Desktop\Principal>
```

```
Símbolo del sistema
C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta

2
La planta ha sido regada
La planta ha sido abonada
La planta esta floreciendo... Siga cuidandola... :)

C:\Users\cat-b\Desktop\Principal>java Principal
Seleccione la accion a realizar...
1.-Plantar
2.-Cuidar Planta
3.-Cambiar maceta

3
La planta ha sido cambiada de maceta
La maceta 1 esta libre
La maceta 2 esta ocupada

C:\Users\cat-b\Desktop\Principal>
```


Conclusiones

Dávila García Rivas Emiliano

A medida que abstraía los objetos para diseñar las clases que eran destinadas a resolver el problema especificado, se presentaron algunos problemas imprevistos. Estos causados tanto por mi inexperiencia con el lenguaje de programación, como mi entendimiento en cuanto a la relación de clases (Tanto, agregación como composición). Un claro ejemplo del segundo tipo de situación fue el uso "incorrecto de la agregación (En realidad no fue incorrecto, sino que realmente no era agregación, sino uso.) lo cual remarco la necesidad de estudio en cuanto a esos temas.

Además, los problemas en cuanto a la abstracción fueron más redundantes, uno más serio de ellos fue el hecho de tener una falta de comprensión en cuanto al nivel necesario de abstracción, es decir, que tan refinado debía ser en la creación de clases. Esto provocó una confusión, sin embargo, a mi parecer el nivel de abstracción, aunque sencillo, tiene el potencial (Con el conocimiento necesario) para resolver el problema original.

Luciano Espina Melisa

En el desarrollo de esta práctica fue importante recurrir a los apuntes vistos en clase e información en internet para poder hacerla correctamente, desde la parte donde se debía buscar el problema, hasta lograr que funcione la aplicación. Las relaciones que se deben de manejar son importantes, ya que con ellas podemos distinguir si un objeto necesita de otros dentro de una misma clase o de otras.

Se logró identificar las diferencias entre cada una de las relaciones y aplicarlas en nuestro programa.

Ramos Mesas Edgar Alaín

El proceso de abstracción de un ente es un proceso aplicable para todas las creaciones de clases, por lo cual nuevamente en esta práctica hemos realizado una abstracción para cada uno de los objetos.

Se decidió trabajar con el ejemplo de un Jardín ya que, aparentemente, es fácil relacionar todos los entes existentes en este entorno. El proceso de abstracción de un Jardinero se limitó más bien a lo técnico. En el caso de la planta, en cambio, se puede realizar un proceso de abstracción más complejo pues esta, a su vez, se compone de muchos entes. Se hizo uso del ente maceta, ya que, como es común, las plantas deben estar contenidas dentro de algún objeto de este tipo. La maceta, al igual que el jardinero, fue un objeto abstraído en un ámbito más práctico.

Una vez se hubo realizado el proceso de abstracción tuvimos que recurrir nuevamente, al igual que con el proceso de abstracción, a un diagrama de clases. Sin embargo, a diferencia de la primera práctica, en esta ocasión se tuvo que contemplar la forma en la que se relacionaban todos y cada uno de los entes entre sí.

El proceso de relación entre entes es un poco complicado, pues requiere de un análisis más profundo de cada uno de los entes, y obviamente, de la forma en la que interactúan con los otros.

Por ejemplo, en el caso del jardinero y la maceta, encontramos que el jardinero simplemente usa a la maceta, lo cual es muy parecido a la planta. Por otro lado, en el caso de agregación, podíamos encontrar que una maceta podía ser agregada a nuestro Jardín. El jardinero y la planta estaban relacionados en algún tipo de uso, ya que la planta solo requería cuidados del jardinero, sin embargo, si a un Jardinero se le asignaba alguna planta y /o maceta, se podría conseguir una relación de asociación. Nuevamente haciendo uso de la planta, pudimos generar la relación de composición, esto gracias a que una planta, está compuesta por hojas, frutos, etc.

Durante el desarrollo de la práctica pudimos notar que, si bien no es muy difícil abstraer un objeto, o relacionarlo con algún otro, pasa algo un poco diferente al nivel de programación pues es donde encontramos algunos problemas. Cuando se realizó el código se encontró que, de alguna manera, se habían omitido las relaciones señaladas, sin embargo, eso debido al hecho de error de análisis a nivel lógico, o de codificación.

Tanto el proceso de abstracción como de relación son procesos que requieren un cuidado especial pues al momento de codificar se pueden presentar confusiones en cuestión de análisis lógico. Es por eso por lo que se debe tomar en cuenta la forma de utilizar funciones, así como hacer referencias a las mismas, y hacer envío de información entre las mismas pues de realizarlo de forma equivocada podríamos cometer errores terribles en el código y en las aplicaciones.

Bibliografía

- [1] S/A Relaciones entre clases [UML] | NativeHex [Online] Available: <https://blog.nativehex.com/relaciones-entre-clases-uml>