Patrón de diseño Abstract Factory

Trabajo escrito presentado a :

Profesor Jhon Cano

Presentado por:

Estudiante: Milton Giovanni Rodriguez Guerrero

Universidad Cooperativa de Colombia

Ingenieria en Sistemas VI

Cali, Colombia 2020

**Programa Servicios en Arte**

Este programa se diseño con el objetivo de usar el patrón de diseño AbstractFactory (o diseño de fabrica). La idea es que se supla una necesidad que tengo como artista y como empresario (tengo colegas a quienes se le asignara un proyecto de arte) donde estoy dando distintos servicios en arte de acuerdo a las habilidades en creación de retratos, caricaturas, y trabajos en oleo.

Primero diseñe el diagrama UML en el sitio web <https://app.diagrams.net/> (ver Fig. 1). En este diagrama cree dos paquetes uno que se llama (AbstractFactory ) y otro que se llama (Prueba).

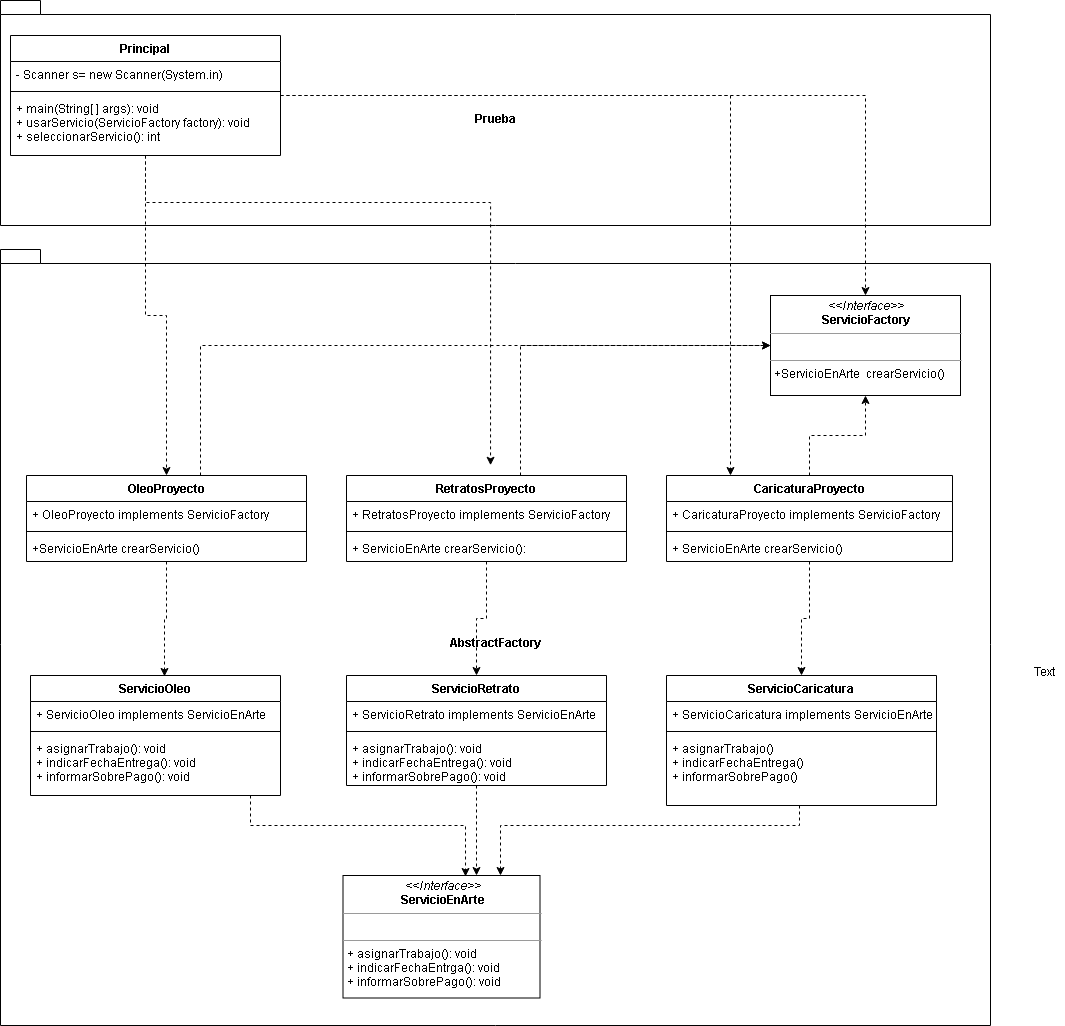


Fig. 1

En el paquete AbstractFactory es donde se desenvuelve el patrón de diseño el cual me permite la creación la instanciación de objetos de distintas clases.

En el paquete Prueba hago uso de las clases.

Como la necesidad de mi problema es crear objetos de distintas clases pero con métodos en común es decir mis productos que puedo ofrecer a mis clientes como dibujante serian:

Clases producto:

* ServicioOleo
* ServicioRetrato
* ServicioCaricatura

Para esto cree una factory abstracta la cual llame “ServicioFactory “ y de esta interface se derivan las clases fabrica :

* OleoProyecto
* RetratosProyecto
* CaricaturaProyecto

Estas son las encargadas de instanciar correspondientemente los objetos de las clases producto.

Así que la clase ServicioOleo me va crear objetos de OleoProyecto, la clase ServicioRetrato me va crear objetos RetratosProyecto y la clase ServicioCaricatura me va a crear objetos CaricaturaProyecto.

Además esta interfaz (ServicioFactory) lo que hace es definir un método abstracto que se llama:

* crearServicio()

el cual me devuelve un ServicionEnArte.

ServicioEnArte es una interfaz que viene siendo el producto abstracto que contiene tres funciones:

* asignarTrabajo()
* indicarFechaEntrega()
* informarSobrePago()

(Estos métodos están implementados en cada una de las clase producto).

La función asignarTrabajo() es un método para asignarle un trabajo a un colega del arte que previamente tendrá una relación conmigo en modalidad de prestación de servicios.

La función indicarFechaEntrega() es un método para estimar una fecha de entrega y que el cliente pueda conocer el detalle de cuando tendrá la obra en sus manos.

La función informarSobrePago() es un método para darle a conocer al cliente el valor a pagar de acuerdo con la cantidad de obras realizadas.

A continuación el código desarrollado en Netbeans con JAVA:

CaricaturaProyecto.java

package AbstractFactory;

public class CaricaturaProyecto implements ServicioFactory {

@Override

public ServicioEnArte crearServicio(){

return new ServicioCaricatura();

}

}

OleoProyecto.java

package AbstractFactory;

public class OleoProyecto implements ServicioFactory{

@Override

public ServicioEnArte crearServicio(){

return new ServicioOleo();

}

}

RetratosProyecto.java

package AbstractFactory;

public class RetratosProyecto implements ServicioFactory {

@Override

public ServicioEnArte crearServicio(){

return new ServicioRetrato();

}

}

ServicioCaricatura.java

package AbstractFactory;

public class ServicioCaricatura implements ServicioEnArte{

@Override

public void asignarTrabajo(){

System.out.println("Nuestro Caricaturista esta informado del proyecto.");

}

@Override

public void indicarFechaEntrega(){

System.out.println("Su Caricatura estar terminada 20/3/2020");

}

@Override

public void informarSobrePago(){

System.out.println("El monto a pagar será proporcional a la cantidad de personas en caricaturizar");

}

}

ServicioEnArte.java

package AbstractFactory;

public interface ServicioEnArte {

public void asignarTrabajo();

public void indicarFechaEntrega();

public void informarSobrePago();

}

ServicioFactory.java

package AbstractFactory;

public interface ServicioFactory {

public ServicioEnArte crearServicio();

}

ServicioOleo.java

package AbstractFactory;

public class ServicioOleo implements ServicioEnArte{

@Override

public void asignarTrabajo(){

System.out.println("El trabajo ha sido asignado a artistas disponibles");

}

@Override

public void indicarFechaEntrega(){

System.out.println("El artista ha determinado terminar la obra como maximo el día 15/03/2020");

}

@Override

public void informarSobrePago(){

System.out.println("Debe realizar el pago en el momento de entrega y a satisfacción.");

}

}

ServicioRetrato.java

package AbstractFactory;

public class ServicioRetrato implements ServicioEnArte {

public void asignarTrabajo(){

System.out.println("El Artista de retratos se le ha asignado el proyecto.");

}

public void indicarFechaEntrega(){

System.out.println("El retrato sera terminado a mas tardar el 12/03/2020.");

}

public void informarSobrePago(){

System.out.println("El monto a pagar no incluye el envio.");

}

}