Ejemplo de patrón factory method

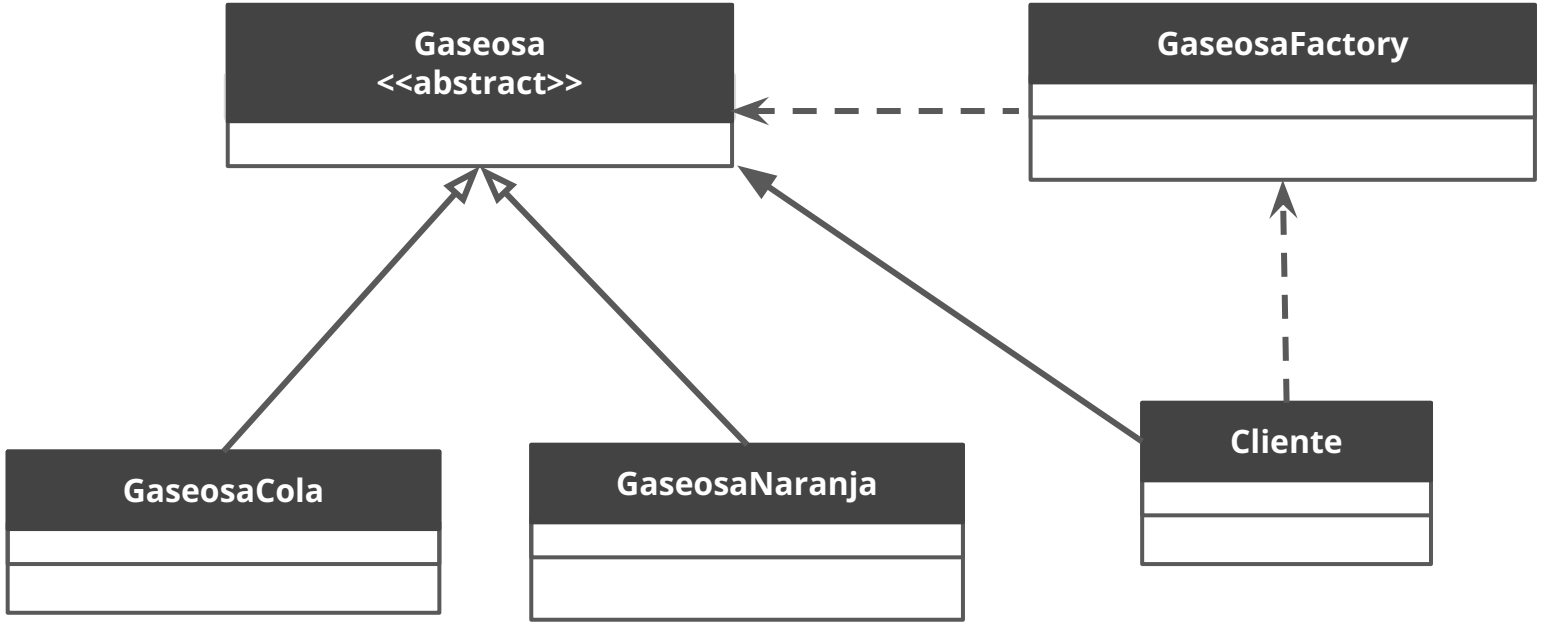
DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Patrón factory method: Ejemplo

En este ejemplo iremos a la implementación del factory method, utilizando un fabricador de gaseosas que construye diferentes tipos dependientes de la implementación de sus subclases.

Factory method



Clase abstracta del producto

```
public abstract class Gaseosa {
   private  String nombre;

   public String getNombre(){
       return nombre;
   }

   public void abrir(){
       System.out.println("Abriste una refrescante gaseosa de " + getNombre());
   }
}
```

Clases concretas de los productos

Cada producto concreto hereda de la clase abstracta principal.

```java
public class GaseosaCola extends Gaseosa {
    String nombre = "Coca Cool";

    @Override
    public String getNombre() {
        return nombre;
    }
    @Override
    public void abrir() {
        super.abrir();
    }
}
```

```java
public class GaseosaNaranja extends Gaseosa{
    String nombre = "Naranja dulce";

    @Override
    public String getNombre() {
        return nombre;
    }
    @Override
    public void abrir() {
        super.abrir();
    }
}
```

Clase Fabricadora (GaseosaFactory)

```
public class GaseosaFactory {
   public static Gaseosa construir(String tipo){

       switch (tipo){
           case "Coca":
               return new GaseosaCola();
           case "Naranja":
               return new GaseosaNaranja();
           default:
               System.out.println("Ups, no encontramos este objeto para construir");
               return null;
       }
   }
}
```

{ejecución}

Se crean tres instancias usando el método estático **construir** de la clase fabricadora *GaseosaFactory,* dos de los productos son existentes, sin embargo el tercero, ***"uva"***, nunca fue creado.

```
    public static void main(String[] args) {

    Gaseosa gs1 = GaseosaFactory.construir("Coca");
    gs1.abrir();

    Gaseosa gs2 = GaseosaFactory.construir("Naranja");
    gs2.abrir();

    Gaseosa gs3 = GaseosaFactory.construir("Uva");
    gs3.abrir();
}
```

{ejecución}

Al compilar vemos cómo con solo pasarle el parámetro correspondiente al fabricador, usa la subclase correspondiente sin que el cliente necesite especificarla. También tenemos una excepción que podemos controlar con un try catch.

```
Abriste una refrescante gaseosa de Coca Cool
Abriste una refrescante gaseosa de Naranja dulce
Ups, no encontramos este objeto para construir

Exception in thread "main" java.lang.NullPointerException: Cannot invoke
"Gaseosa.abrir()" because "gs3" is null
    at Cliente.main(Cliente.java:11)

}
```

{ejecución}

```
public static void main(String[] args) {
    try {
       Gaseosa gs1 = GaseosaFactory.construir("Coca");
       gs1.abrir();

       Gaseosa gs2 = GaseosaFactory.construir("Naranja");
       gs2.abrir();

       Gaseosa gs3 = GaseosaFactory.construir("Uva");
       gs3.abrir();

    }catch (Exception e){
       System.out.println("¡Exception encontrada!: " + e);
    }
}
```

{ejecución}

```
Abriste una refrescante gaseosa de Coca Cool

Abriste una refrescante gaseosa de Naranja dulce

Ups, no encontramos este objeto para construir

¡Exception encontrada!: java.lang.NullPointerException:
Cannot invoke "Gaseosa.abrir()" because "gs3" is null
```


DigitalHouse>
Coding School