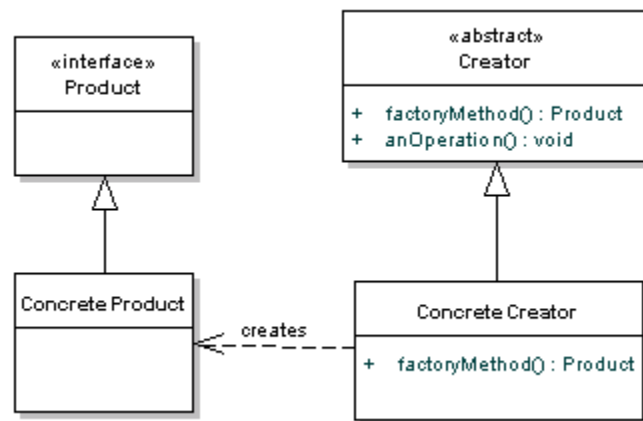


The Factory Pattern

The Factory Pattern อยู่ในกลุ่ม Creational Patterns เป็น Pattern ที่นิยามเมธอดสำหรับสร้างอ็อบเจกต์ของคลาสชนิดหนึ่งๆ แต่ไม่ได้เจาะจงชนิด โดย Factory Pattern จะเป็นรูปแบบที่จำลองโรงงานสร้างสิ่งของขึ้นมา โดยที่เราสามารถสั่งสร้างของได้โดยไม่ต้องสนใจโลจิกการสร้างของชิ้นนั้นๆ ทำให้ง่ายต่อการสร้างobject

การใช้ Factory Patternทำให้สามารถสร้าง object ที่พร้อมสำหรับใช้งาน โดยไม่ต้องระบุ data type ที่แท้จริงของ object ที่ต้องการสร้าง แต่จะให้sub classเป็นส่วนตัดสินใจในการสร้าง object ที่เหมาะสมแทน



โครงสร้าง factory pattern

Creator เป็น Abstract Class ที่มีหน้าที่กำหนดคุณสมบัติของ *Factory* โดยภายในจะมี Abstract Method ที่มีไว้ให้sub class ใช้สร้าง Object เรียกว่า `factoryMethod()`

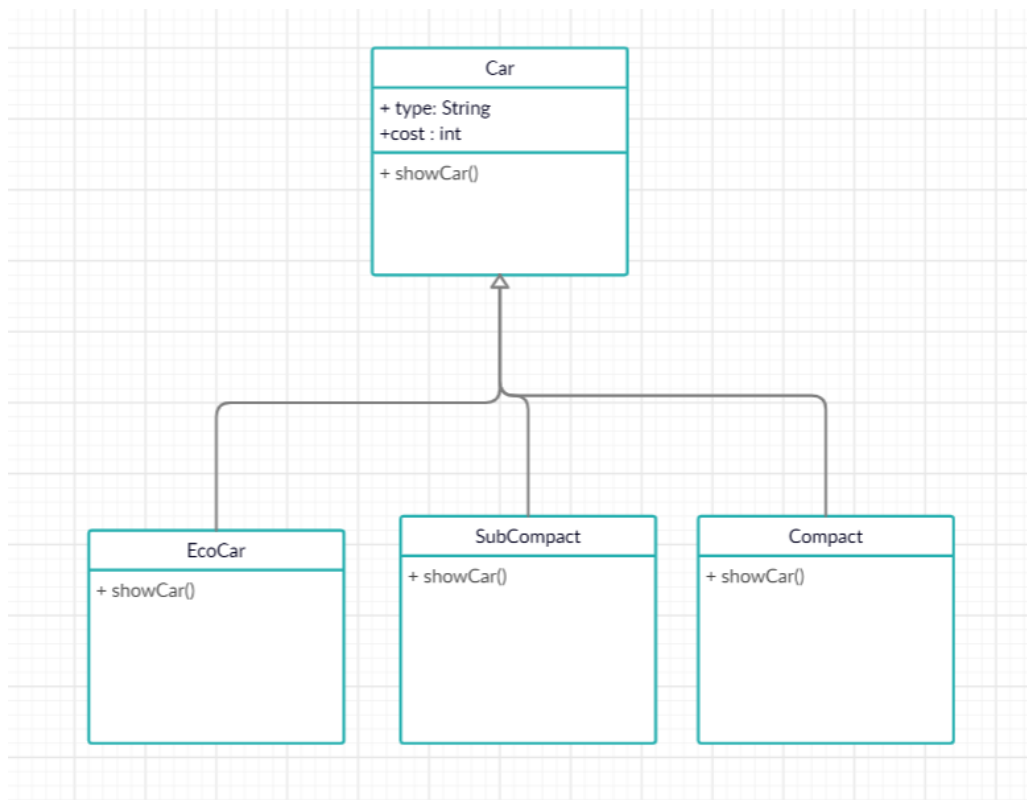
ConcreteCreator เป็นsub classของ **Creator** เป็น Class ที่มีการoverride abstract method โดยในส่วนนี้จะมีการกำหนดการสร้างObject

Product เป็นClassที่เป็นตัวกำหนดคุณสมบัติของ Object ที่จะถูกสร้างขึ้นจาก ConcreteCreator

ConcreteProduct เป็นsub classของProduct ที่มีการทำงานของ Method ที่ Override มาจาก Product

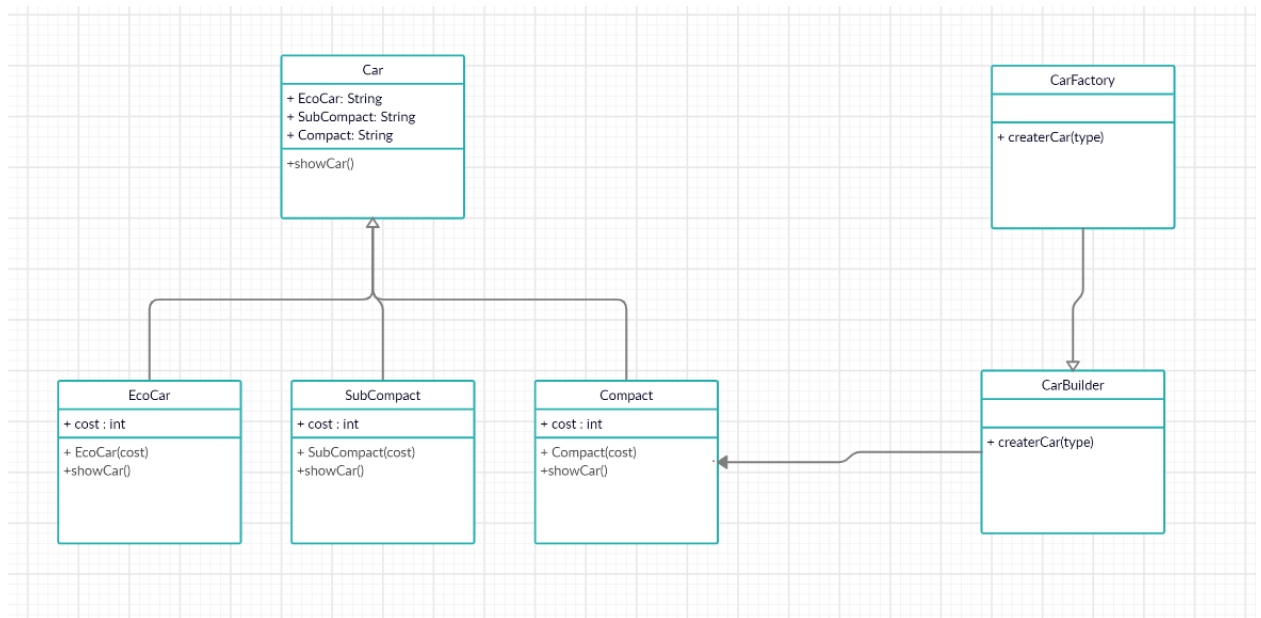
กรณีศึกษา

“โรงงานผลิตรถยนต์” เราต้องการจะสร้างรถยนต์ทั้งหมด 3 แบบ คือ 1.Eco Car 2.Sub Compact 3.Compact ที่มีการกำหนด โดยถ้าดูแบบภาพรวมก็จะเห็นว่าทุกๆแบบนั้นล้วนเป็นรถยนต์เหมือนกัน จะสามารถออกแบบได้ตามภาพด้านล่าง



จากภาพด้านบนเมื่อเรานำไปเขียนcodeจะเห็นว่า ถ้าในอนาคตมีการเปลี่ยนเงื่อนไขในรถต่างๆแบบใหม่ อาจจะทำให้เราต้องทำการแก้ไขหรือเพิ่มcodeใหม่ทั้งหมด ซึ่งจะทำให้codeมีขนาดที่ใหญ่ขึ้นเรื่อยๆ โดยเราสามารถทำการป้องกันปัญหานี้ได้ด้วยการใช้Factory Pattern

ซึ่งเมื่อลองแปลงClass Diagram “โรงงานผลิตรถยนต์” ก่อนหน้า ให้ออกมาเป็น Class Diagram ตามรูปโครงสร้างFactory Patternที่กล่าวไว้ก่อนหน้าแล้วจะได้ตามภาพด้านล่าง



โดยจากDiagramข้างต้นจะเห็นว่าเมื่อทำการปรับรูปแบบให้เป็นแบบ The Factory Pattern แล้ว จะสังเกตเห็นได้ว่าในอนาคตหากมีการเพิ่มหรือปรับเปลี่ยนรูปแบบการผลิตรถนั้น ก็สามารถเพิ่มได้โดยไม่จำเป็นต้องแก้ไขcodeทั้งหมด โดยจาก Class Diagram ข้างต้นจะมีรายละเอียด code ดังนี้

```
Car.java X
src > v1 > Car.java > Car > showCar()
1 package v1;
2 public interface Car{
3     public String EcoCar="EcoCar";
4     public String SubCompact="SubCompact";
5     public String Compact="Compact";
6     public abstract void showCar();
7 }
```

รูปที่ 1 คลาส Car

```
EcoCar.java X
src > v1 > EcoCar.java > EcoCar > showCar()
1 package v1;
2
3 public class EcoCar implements Car{
4     private int cost;
5     public EcoCar(int cost){
6         this.cost = cost;
7     }
8
9     @Override
10    public void showCar() {
11        System.out.println("Eco car cost "+this.cost);
12    }
13
14 }
```

รูปที่ 2 คลาส EcoCar

```
SubCompact.java X
src > v1 > SubCompact.java > SubCompact > showCar()
1 package v1;
2
3 public class SubCompact implements Car{
4     private int cost;
5     public SubCompact(int cost){
6         this.cost = cost;
7     }
8
9     @Override
10    public void showCar() {
11        System.out.println("SubCompact car cost "+this.cost);
12    }
13
14 }
```

รูปที่ 3 คลาส SubCompact

```
Compact.java X
src > v1 > Compact.java > Compact > showCar()
1 package v1;
2
3 public class Compact implements Car{
4     private int cost;
5     public Compact(int cost){
6         this.cost = cost;
7     }
8
9     @Override
10    public void showCar() {
11        System.out.println("Compact car cost "+this.cost);
12    }
13
14 }
```

รูปที่ 4 คลาส Compact

```

CarFactory.java X
src > v1 > CarFactory.java > CarFactory > createCar(String)
1  package v1;
2  public class CarFactory{
3      public static Car createCar(String type){
4          if(type.equals(Car.EcoCar)){
5              return new EcoCar(100000);
6          }
7
8          else if(type.equals(Car.Compact)){
9              return new Compact(200000);
10         }
11         else if(type.equals(Car.SubCompact)){
12             return new SubCompact(300000);
13         }
14         else{
15             return null;
16         }
17     }
18 }
19
20

```

รูปที่ 5 คลาส CarFactory

```

CarBuilder.java X
src > v1 > CarBuilder.java > CarBuilder > main(String[])
1  package v1;
2
3
4  public class CarBuilder {
5      Run | Debug
6      public static void main(String[] args) {
7          Car Eco1 = CarFactory.createCar(Car.EcoCar);
8
9          Car compact = CarFactory.createCar(Car.Compact);
10
11         Eco1.showCar();
12         compact.showCar();
13     }
14 }
15

```

รูปที่ 6 คลาส CarBuilder

แหล่งอ้างอิง

1. <https://th.wikipedia.org/wiki/%E0%B9%81%E0%B8%9A%E0%B8%9A%E0%B9%82%E0%B8%A3%E0%B8%87%E0%B8%87%E0%B8%B2%E0%B8%99%E0%B8%A2%E0%B9%88%E0%B8%AD>
2. <https://www.saladpuk.com/beginner-1/design-patterns/creational/factory-method-pattern#undefined-1>
3. <https://medium.com/thipwriteblog/design-patterns-series-1-factory-method-%E0%B8%8A%E0%B8%A7%E0%B8%99%E0%B8%A1%E0%B8%B2%E0%B8%94%E0%B8%B9%E0%B9%82%E0%B8%A3%E0%B8%87%E0%B8%87%E0%B8%B2%E0%B8%99%E0%B9%84%E0%B8%AD%E0%B8%A8%E0%B8%81%E0%B8%A3%E0%B8%B5%E0%B8%A1-5feb099298e2>