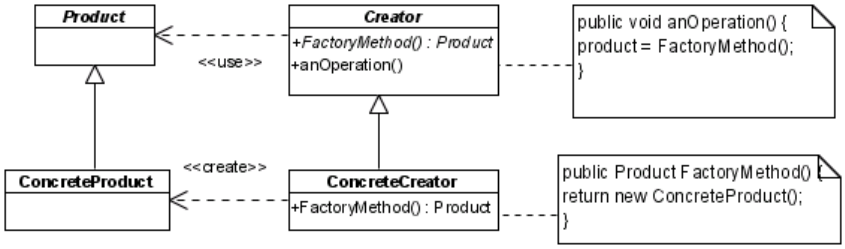
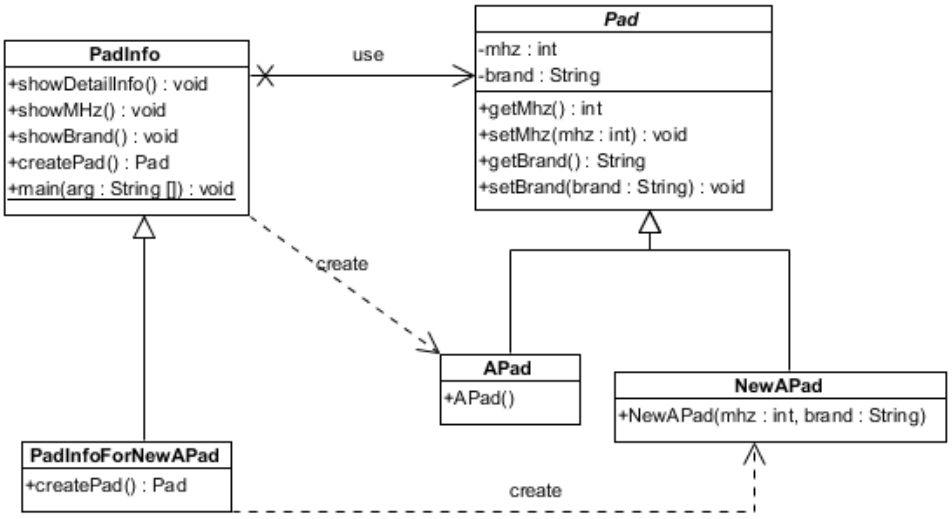
Supplementary Exercise on Factory Pattern

(a)



- Product defines the interface for the type of objects the factory method creates
- ConcreteProduct implements the Product interface
- Creator declares the factory method, which returns an object of type Product
- ConcreteCreator overrides the factory method to return an instance of a ConcreteProduct.

(b)



(c)

```
public class PadInfo {
  public void showDetailInfo() {
          Pad p = createPad();
          System.out.println("The "+ p.getBrand()+ " pad runs at " +
                                                        p.getMhz() + " MHz");
  }

  public Pad createPad() {
       return new APad();
  }

  public static void main(String[] args) {
       PadInfo p = new PadInfo();
       p.showDetailInfo();
  }

}
```

```
public class PadInfoForNewAPad extends PadInfo {
  public Pad createPad(){
        return new NewAPad(600, "Samchung" );
   }

  public static void main(String[] args) {
       PadInfo p = new PadInfoForNewAPad();
       p.showDetailInfo();
   }

}
```

(d)

Any one of the differences:

Factory Method	**Abstract Factory**
The same class creates and uses the product.	the client uses the factory class to creates the product.
Change the product created by overriding the factory method in the subclass of Creator	Change the product created by changing the associated factory class.

The Abstract Factory design pattern is more suitable for creating <u>a set of dependent/compatible products</u>