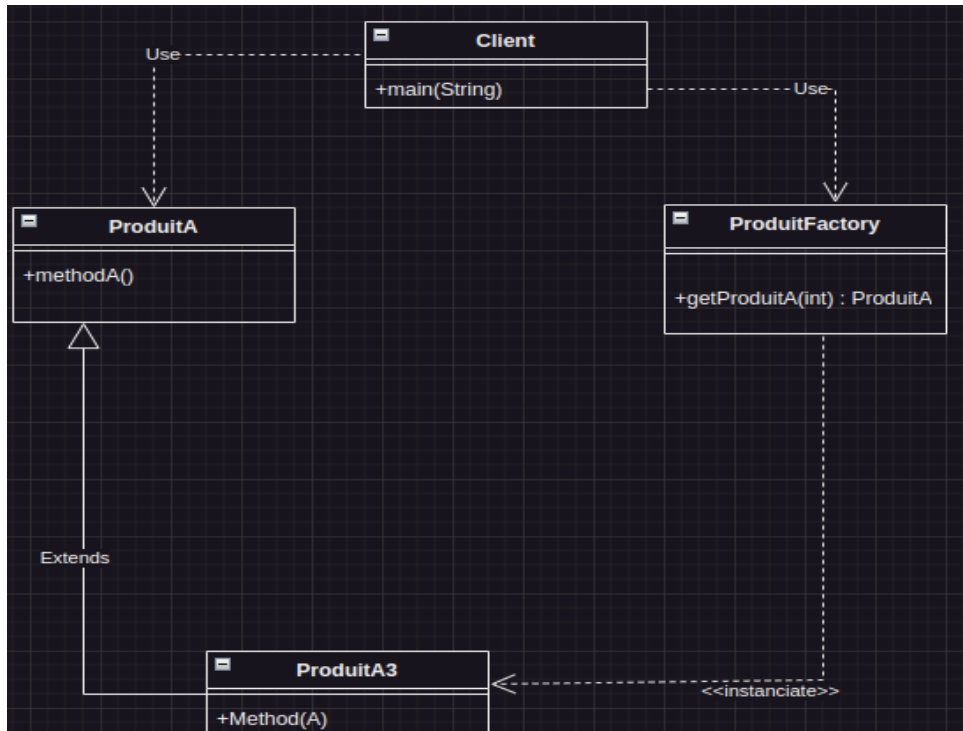


Jeudi 5 Octobre 2023

Exercice 1 :

- patron de construction : «factory method»
modèle générique



- Code : Client

```
1 public class Client {
2
3     public static void main(String[] args) {
4
5         ProduitFactory produitFactory = new ProduitFactory();
6
7         ProduitA produitA = null;
8
9         produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
10        produitA.methodeA();
11    }
12 }
```

➤ ProduitFactory

```
1 public class ProduitFactory {
2
3
4     public static final int TYPE_PRODUIT_A3 = 3;
5
6     public ProduitA geProduitA(int typeProduit){
7         ProduitA produitA = null;
8
9         switch (typeProduit) {
10             case TYPE_PRODUIT_A3:
11                 produitA = new ProduitA3();
12                 break;
13             default:
14                 throw new IllegalArgumentException("Type de produit inconnu");
15         }
16         return produitA;
17     }
18 }
```

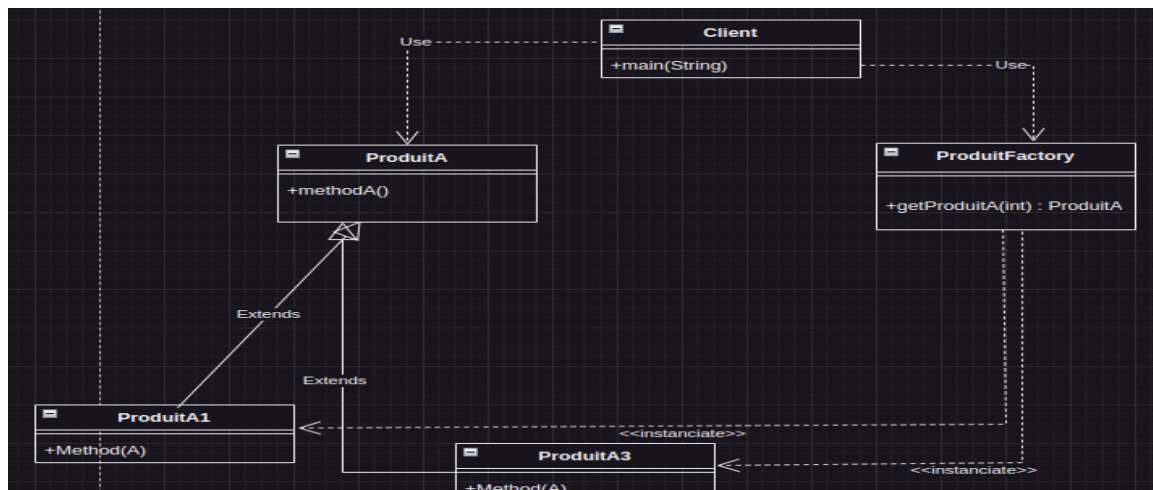
➤ ProduitA3

```
1 public class ProduitA3 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A3");
5         System.out.println("ProduitA3.methodeA");
6     }
7 }
```

➤ ProduitA

```
1 public abstract class ProduitA {
2     public abstract void methodeA();
3 }
```

Avec 2 produits :



➤ ProduitA1 :

```

1 public class ProduitA1 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A1");
5         System.out.println("ProduitA1.methodeA");
6     }
7 }
  
```

➤ ProduitA3 :

```

1 public class ProduitA3 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A3");
5         System.out.println("ProduitA3.methodeA");
6     }
7 }
  
```

➤ client :

```

1 public class Client {
2
3     public static void main(String[] args) {
4
5         ProduitFactory produitFactory = new ProduitFactory();
6
7         ProduitA produitA = null;
8
9         produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
10        produitA.methodeA();
11
12        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
13        produitA.methodeA();
14    }
15 }
  
```

➤ ProduitFactory

```

1 public class ProduitFactory {
2
3     public static final int TYPE_PRODUITA1 = 1;
4     public static final int TYPE_PRODUITA3 = 3;
5
6     public ProduitA geProduitA(int typeProduit){
7         ProduitA produitA = null;
8
9         switch (typeProduit) {
10             case TYPE_PRODUITA1:
11                 produitA = new ProduitA1();
12                 break;
13
14             case TYPE_PRODUITA3:
15                 produitA = new ProduitA3();
16                 break;
17             default:
18                 throw new IllegalArgumentException("Type de produit inconnu");
19         }
20         return produitA;
21     }
22 }

```

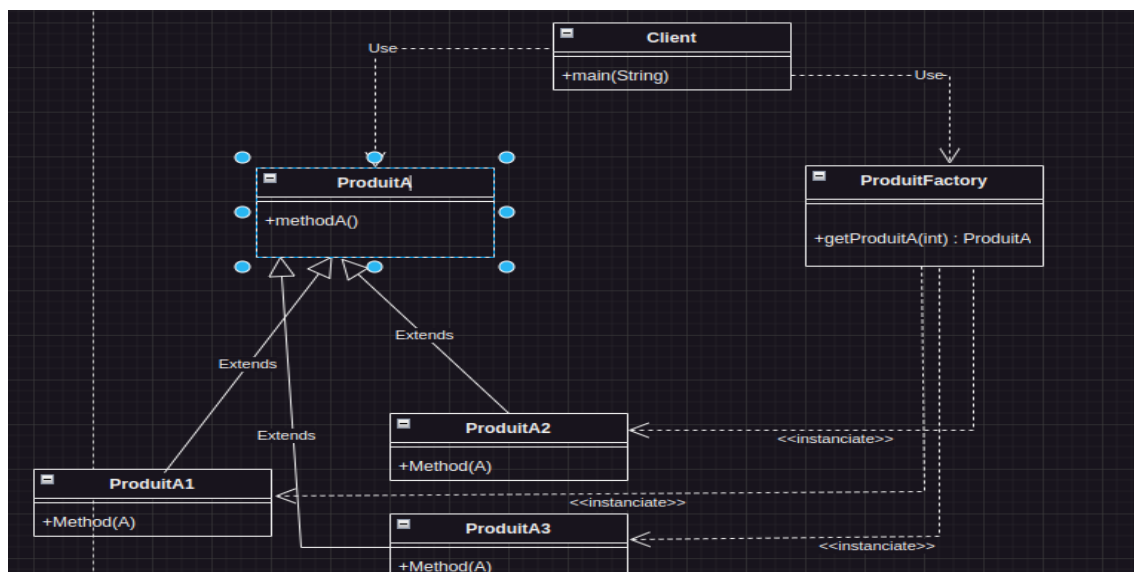
➤ Client

```

1 public class Client {
2
3     public static void main(String[] args) {
4
5         ProduitFactory produitFactory = new ProduitFactory();
6
7         ProduitA produitA = null;
8
9         produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
10        produitA.methodeA();
11
12        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
13        produitA.methodeA();
14    }
15 }

```

avec 3 produits :



➤ produitA1 :

```

1 public class ProduitA1 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A1");
5         System.out.println("ProduitA1.methodeA");
6     }
7 }

```

➤ ProduitA2 :

```

1 public class ProduitA2 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A2");
5         System.out.println("ProduitA2.methodeA");
6     }
7 }

```

➤ ProduitA3 :

```

1 public class ProduitA3 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A3");
5         System.out.println("ProduitA3.methodeA");
6     }
7 }

```

➤ Client :

```

1 public class Client {
2
3     public static void main(String[] args) {
4
5         ProduitFactory produitFactory = new ProduitFactory();
6
7         ProduitA produitA = null;
8
9         produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
10        produitA.methodeA();
11        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA2);
12        produitA.methodeA();
13        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
14        produitA.methodeA();
15    }
16 }

```

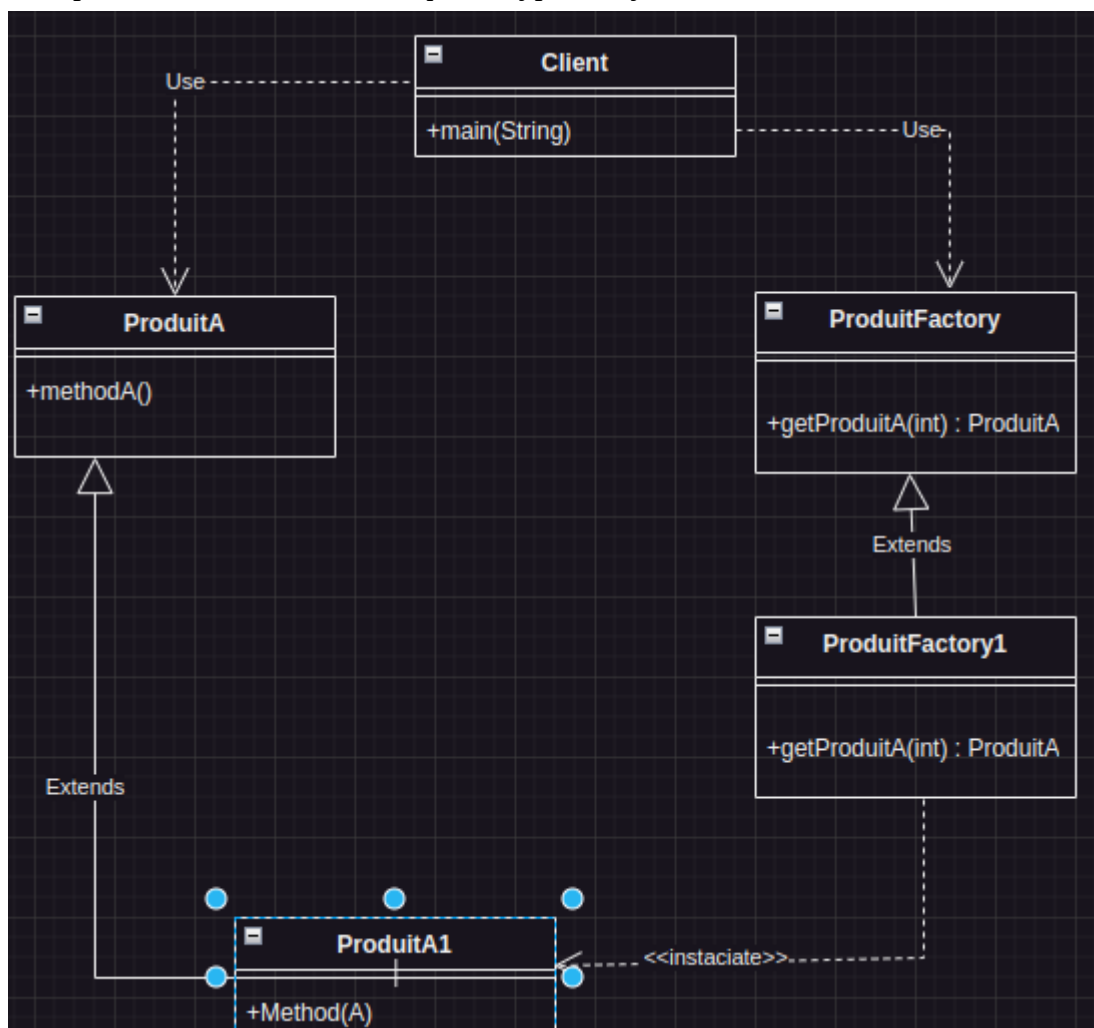
➤ ProduitFactory :


```

1 public class ProduitFactory {
2
3     public static final int TYPE_PRODUITA1 = 1;
4     public static final int TYPE_PRODUITA2 = 2;
5     public static final int TYPE_PRODUITA3 = 3;
6
7     public ProduitA getProduitA(int typeProduit){
8         ProduitA produitA = null;
9
10        switch (typeProduit) {
11            case TYPE_PRODUITA1:
12                produitA = new ProduitA1();
13                break;
14            case TYPE_PRODUITA2:
15                produitA = new ProduitA2();
16                break;
17            case TYPE_PRODUITA3:
18                produitA = new ProduitA3();
19                break;
20            default:
21                throw new IllegalArgumentException("Type de produit inconnu");
22        }
23        return produitA;
24    }
25 }

```

Exercice 2 :
Autant de spécialiste de la fabrication que de type d'objet



➤ client :

```

public class Client {
    Run | Debug
    public static void main(String[] args) {
        ProduitFactory produitFactory1 = new ProduitFactory1();

        ProduitA produitA = null;

        System.out.println("Utilisation de la première fabrique");
        produitA = produitFactory1.getProduitA();
        produitA.methodeA();
    }
}

```

➤ ProduitFactory :

```

public abstract class ProduitFactory {
    public ProduitA getProduitA(){
        return createProduitA();
    }
    protected abstract ProduitA createProduitA();
}

```

➤ ProduitFactory1 :

```

public class ProduitFactory1 extends ProduitFactory{
    protected ProduitA createProduitA(){
        return new ProduitA1();
    }
}

```

➤ ProduitA1 :

```

1 public class ProduitA1 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A1");
5         System.out.println("ProduitA1.methodeA");
6     }
7 }

```

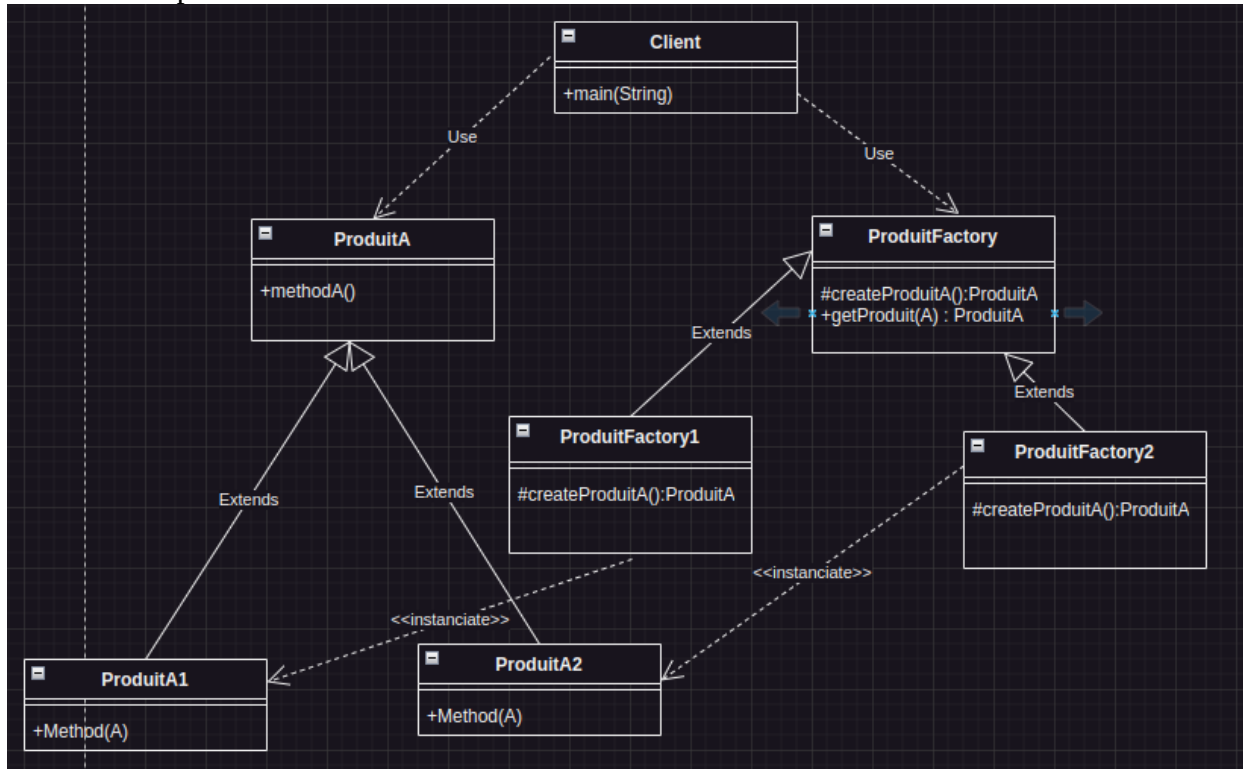
➤ ProduitA :

```

1 public abstract class ProduitA {
2     public abstract void methodeA();
3 }

```

➤ Avec 2 produits



➤ ProduitA2:

```

1 public class ProduitA2 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A2");
5         System.out.println("ProduitA2.methodeA");
6     }
7 }

```

➤ ProduitFactoryA2 :

```

public class ProduitFactory2 extends ProduitFactory{

    protected ProduitA createProduitA(){
        return new ProduitA2();
    }
}

```

➤ Client :


```

public class Client {

    Run | Debug
    public static void main(String[] args) {
        ProduitFactory produitFactory1 = new ProduitFactory1();
        ProduitFactory produitFactory2 = new ProduitFactory2();

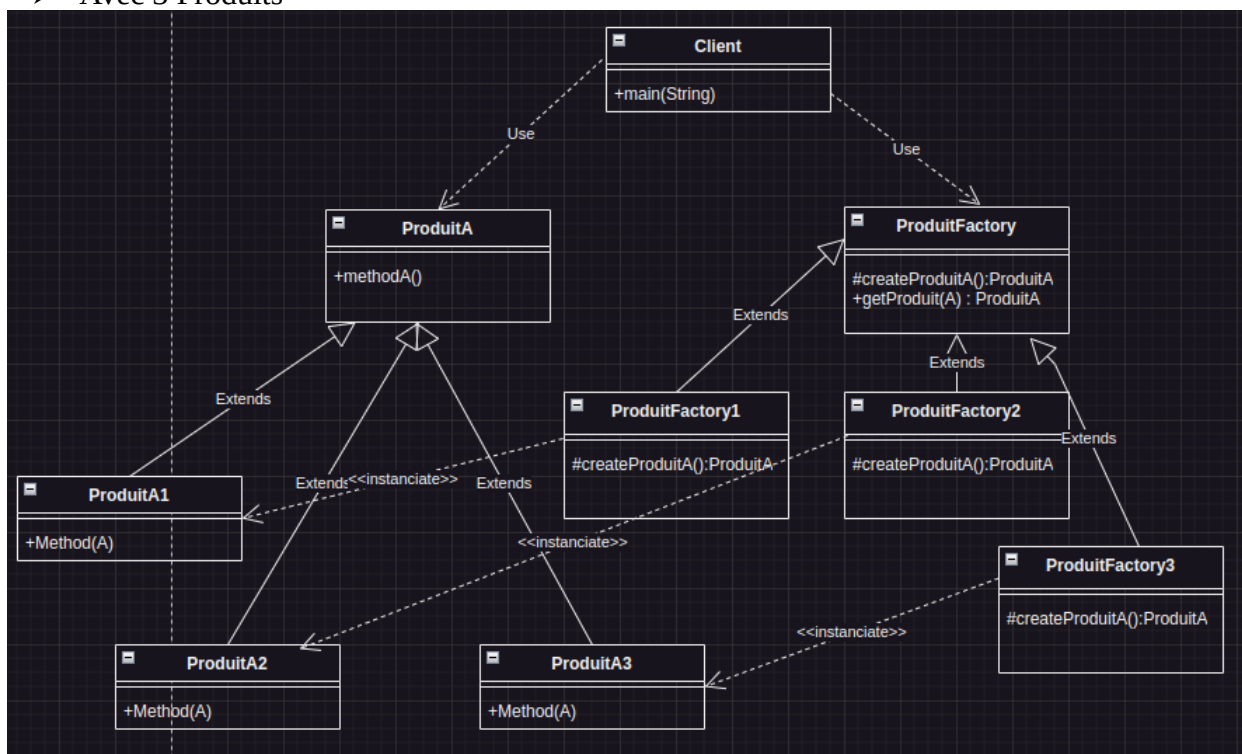
        ProduitA produitA = null;

        System.out.println("Utilisation de la première fabrique");
        produitA = produitFactory1.getProduitA();
        produitA.methodeA();

        System.out.println("Utilisation de la seconde fabrique");
        produitA = produitFactory2.getProduitA();
        produitA.methodeA();
    }
}

```

➤ Avec 3 Produits



➤ ProduitA3 :

```

1 public class ProduitA3 extends ProduitA{
2
3     public void methodeA(){
4         System.out.println("Je suis un produit de type A3");
5         System.out.println("ProduitA3.methodeA");
6     }
7 }

```

➤ ProduitFactory3 :

```

public class ProduitFactory3 extends ProduitFactory{
    protected ProduitA createProduitA(){
        return new ProduitA3();
    }
}

```

➤ ProduitFactory :

```

1 public class ProduitFactory {
2
3     public static final int TYPE_PRODUITA1 = 1;
4     public static final int TYPE_PRODUITA2 = 2;
5     public static final int TYPE_PRODUITA3 = 3;
6
7     public ProduitA geProduitA(int typeProduit){
8         ProduitA produitA = null;
9
10        switch (typeProduit) {
11            case TYPE_PRODUITA1:
12                produitA = new ProduitA1();
13                break;
14            case TYPE_PRODUITA2:
15                produitA = new ProduitA2();
16                break;
17            case TYPE_PRODUITA3:
18                produitA = new ProduitA3();
19                break;
20            default:
21                throw new IllegalArgumentException("Type de produit inconnu");
22        }
23        return produitA;
24    }
25 }

```

➤ Client :

```
public class Client {  
    Run | Debug  
    public static void main(String[] args) {  
        ProduitFactory produitFactory1 = new ProduitFactory1();  
        ProduitFactory produitFactory2 = new ProduitFactory2();  
        ProduitFactory produitFactory3 = new ProduitFactory3();  
  
        ProduitA produitA = null;  
  
        System.out.println("Utilisation de la première fabrique");  
        produitA = produitFactory1.getProduitA();  
        produitA.methodeA();  
  
        System.out.println("Utilisation de la seconde fabrique");  
        produitA = produitFactory2.getProduitA();  
        produitA.methodeA();  
  
        System.out.println("Utilisation de la troisième fabrique");  
        produitA = produitFactory3.getProduitA();  
        produitA.methodeA();  
    }  
}
```