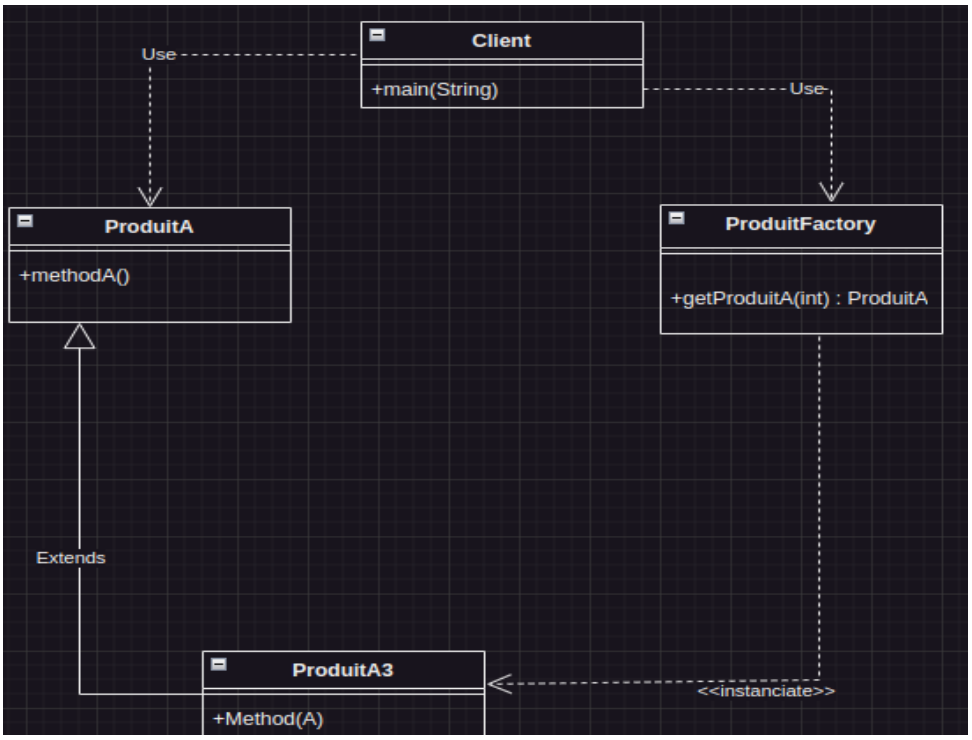
Jeudi 5 Octobre 2023

Exercice 1 :

patron de construction : «factory method»

- modèle générique




- Code : Client

```java
public class Client {

    public static void main(String[] args) {

        ProduitFactory produitFactory = new ProduitFactory();

        ProduitA produitA = null;

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
        produitA.methodeA();
    }
}
```

- ProduitFactory

```java
public class ProduitFactory {

    public static final int TYPE_PRODUITA3 = 3;

    public ProduitA geProduitA(int typeProduit){
        ProduitA produitA = null;

        switch (typeProduit) {
            case TYPE_PRODUITA3:
                produitA = new ProduitA3();
                break;
            default:
                throw new IllegalArgumentException("Type de produit inconnu");
        }
        return produitA;
    }
}
```

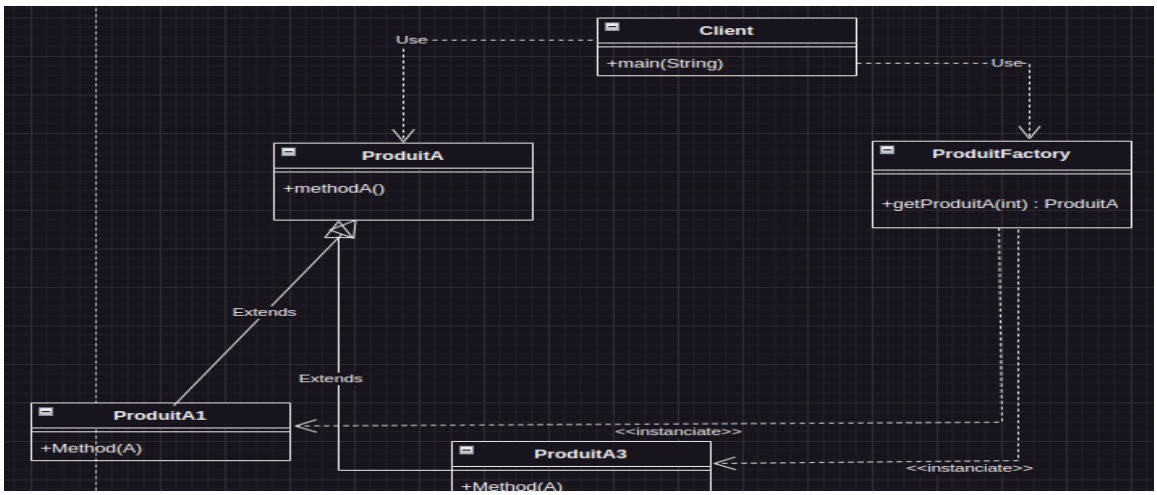
- ProduitA3

```java
public class ProduitA3 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A3");
        System.out.println("ProduitA3.methodeA");
    }
}
```

- ProduitA

```java
public abstract class ProduitA {
    public abstract void methodeA();
}
```

Avec 2 produits :




- ProduitA1 :

```java
public class ProduitA1 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A1
        System.out.println("ProduitA1.methodeA");
    }
}
```

- ProduitA3 :

```java
public class ProduitA3 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A3");
        System.out.println("ProduitA3.methodeA");
    }
}
```

- client :

```java
public class Client {

    public static void main(String[] args) {

        ProduitFactory produitFactory = new ProduitFactory();

        ProduitA produitA = null;

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
        produitA.methodeA();

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
        produitA.methodeA();
    }
}
```

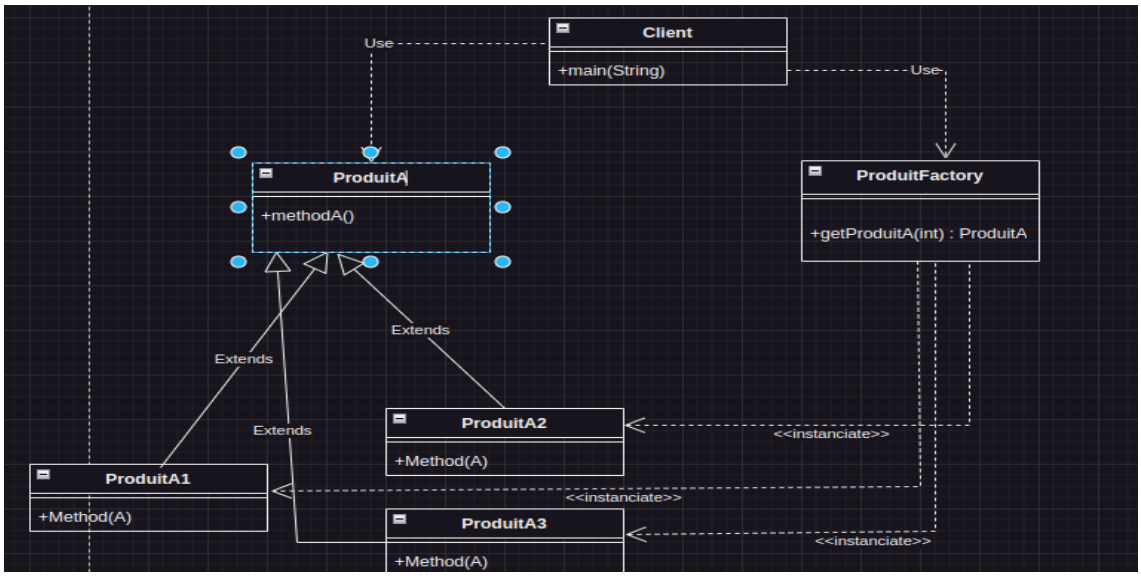
- ProduitFactory

```java
public class ProduitFactory {

    public static final int TYPE_PRODUITA1 = 1;
    public static final int TYPE_PRODUITA3 = 3;

    public ProduitA geProduitA(int typeProduit){
        ProduitA produitA = null;

        switch (typeProduit) {
            case TYPE_PRODUITA1:
                produitA = new ProduitA1();
                break;

            case TYPE_PRODUITA3:
                produitA = new ProduitA3();
                break;
            default:
                throw new IllegalArgumentException("Type de produit inconnu");
        }
        return produitA;
    }
}
```

- Client

```java
public class Client {

    public static void main(String[] args) {

        ProduitFactory produitFactory = new ProduitFactory();

        ProduitA produitA = null;

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
        produitA.methodeA();

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
        produitA.methodeA();
    }
}
```

avec 3 produits :




- produitA1 :

```java
public class ProduitA1 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A1
        System.out.println("ProduitA1.methodeA");
    }
}
```

- ProduitA2 :

```java
public class ProduitA2 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A2");
        System.out.println("ProduitA2.methodeA");
    }
}
```

- ProduitA3 :

```java
public class ProduitA3 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A3");
        System.out.println("ProduitA3.methodeA");
    }
}
```

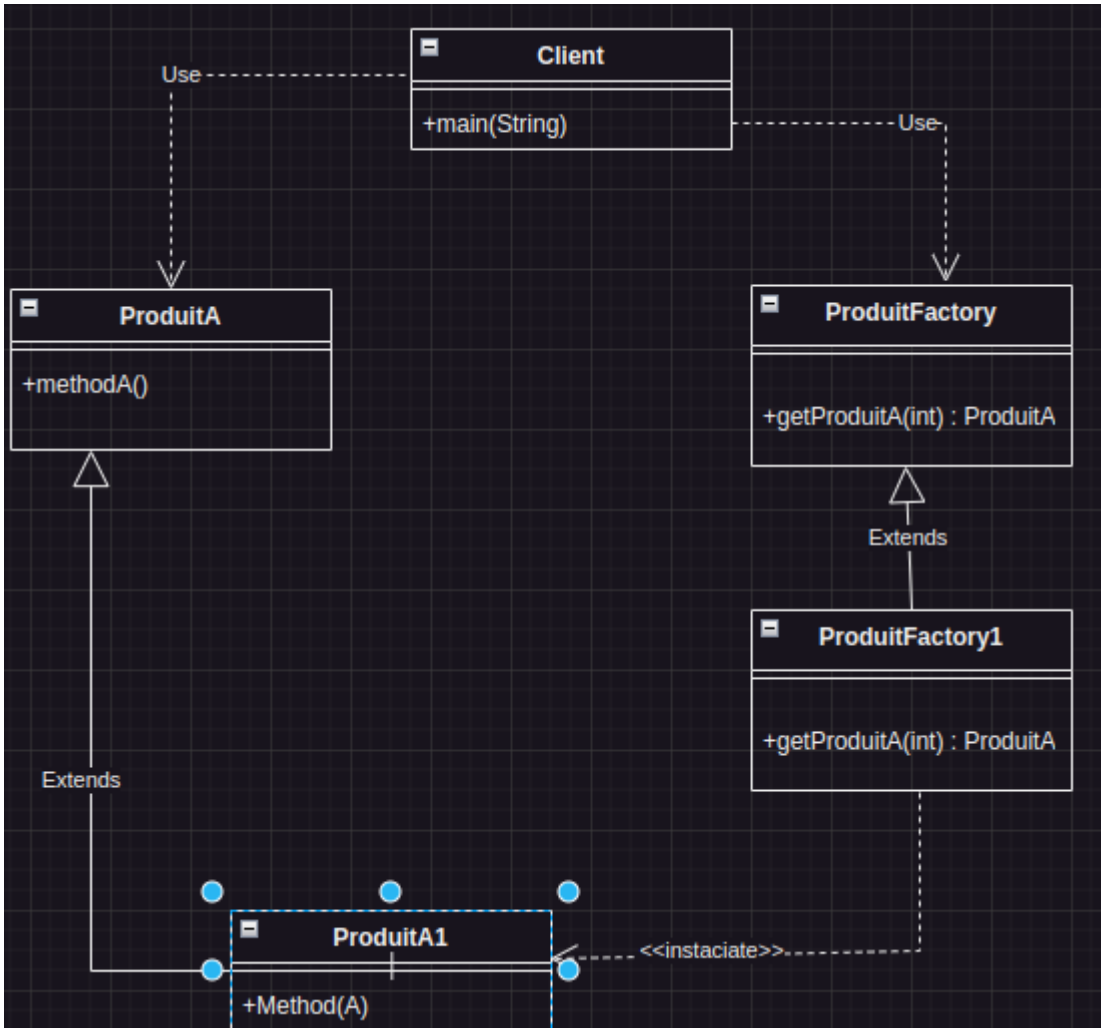
- Client :

```java
public class Client {

    public static void main(String[] args) {

        ProduitFactory produitFactory = new ProduitFactory();

        ProduitA produitA = null;

        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA1);
        produitA.methodeA();
        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA2);
        produitA.methodeA();
        produitA = produitFactory.geProduitA(produitFactory.TYPE_PRODUITA3);
        produitA.methodeA();
    }
}
```

- ProduitFactory :

```java
public class ProduitFactory {

    public static final int TYPE_PRODUITA1 = 1;
    public static final int TYPE_PRODUITA2 = 2;
    public static final int TYPE_PRODUITA3 = 3;

    public ProduitA geProduitA(int typeProduit){
        ProduitA produitA = null;

        switch (typeProduit) {
            case TYPE_PRODUITA1:
                produitA = new ProduitA1();
                break;
            case TYPE_PRODUITA2:
                produitA = new ProduitA2();
                break;
            case TYPE_PRODUITA3:
                produitA = new ProduitA3();
                break;
            default:
                throw new IllegalArgumentException("Type de produit inconnu");
        }
        return produitA;
    }
}
```

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




- client :

```java
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 :

```java
public abstract class ProduitFactory {

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

- ProduitFactory1 :

```java
public class ProduitFactory1 extends ProduitFactory{

    protected ProduitA createProduitA(){
        return new ProduitA1();
    }
}
```

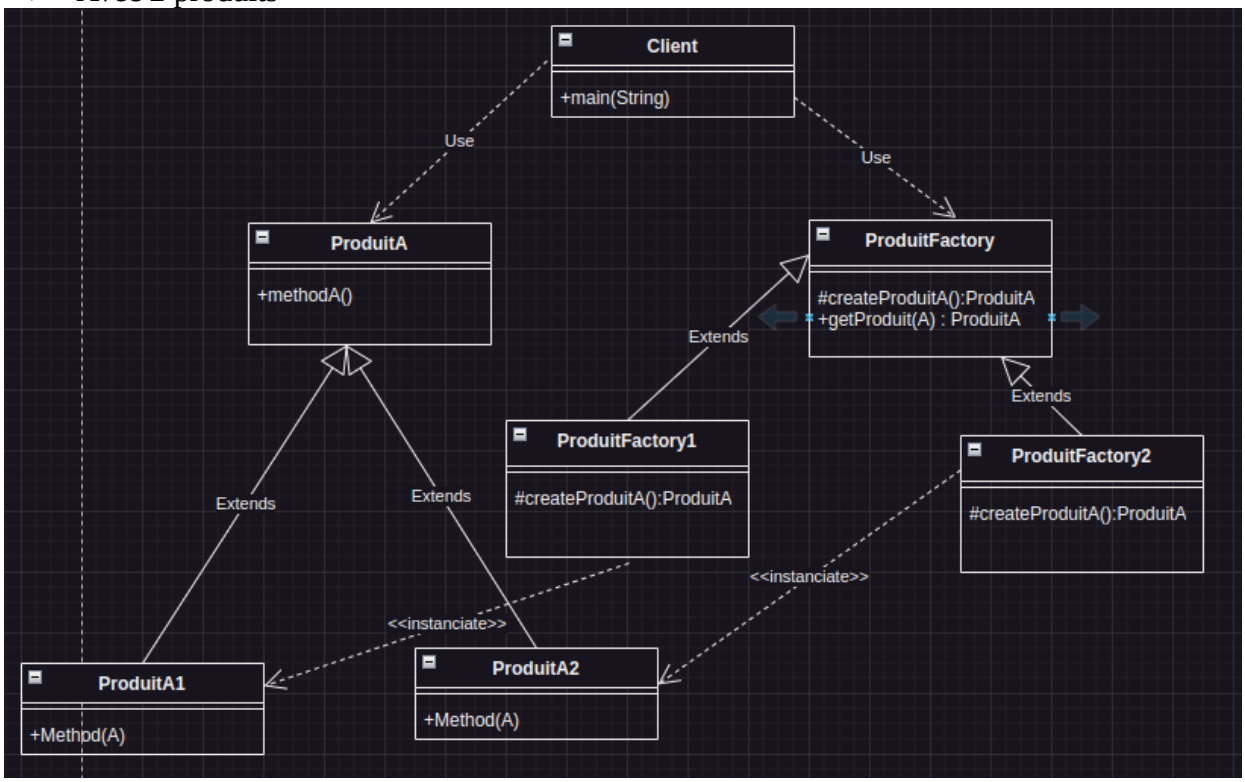
- ProduitA1 :

```java
public class ProduitA1 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A1
        System.out.println("ProduitA1.methodeA");
    }
}
```

- ProduitA :

```java
public abstract class ProduitA {
    public abstract void methodeA();
}
```

- Avec 2 produits




- ProduitA2:

```java
public class ProduitA2 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A2");
        System.out.println("ProduitA2.methodeA");
    }
}
```

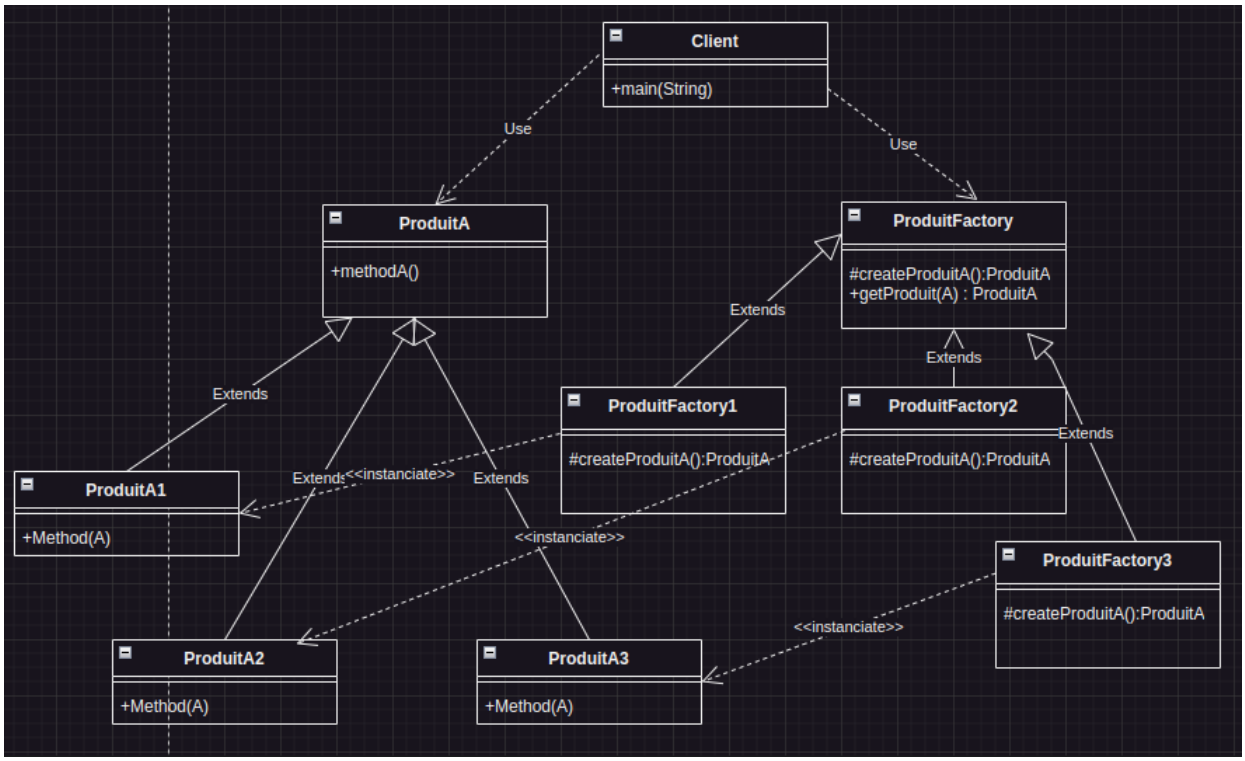
- ProduitFactoryA2 :

```java
public class ProduitFactory2 extends ProduitFactory{

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

- Client :

```java
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 :

```java
public class ProduitA3 extends ProduitA{

    public void methodeA(){
        System.out.println("Je suis un produit de type A3");
        System.out.println("ProduitA3.methodeA");
    }
}
```

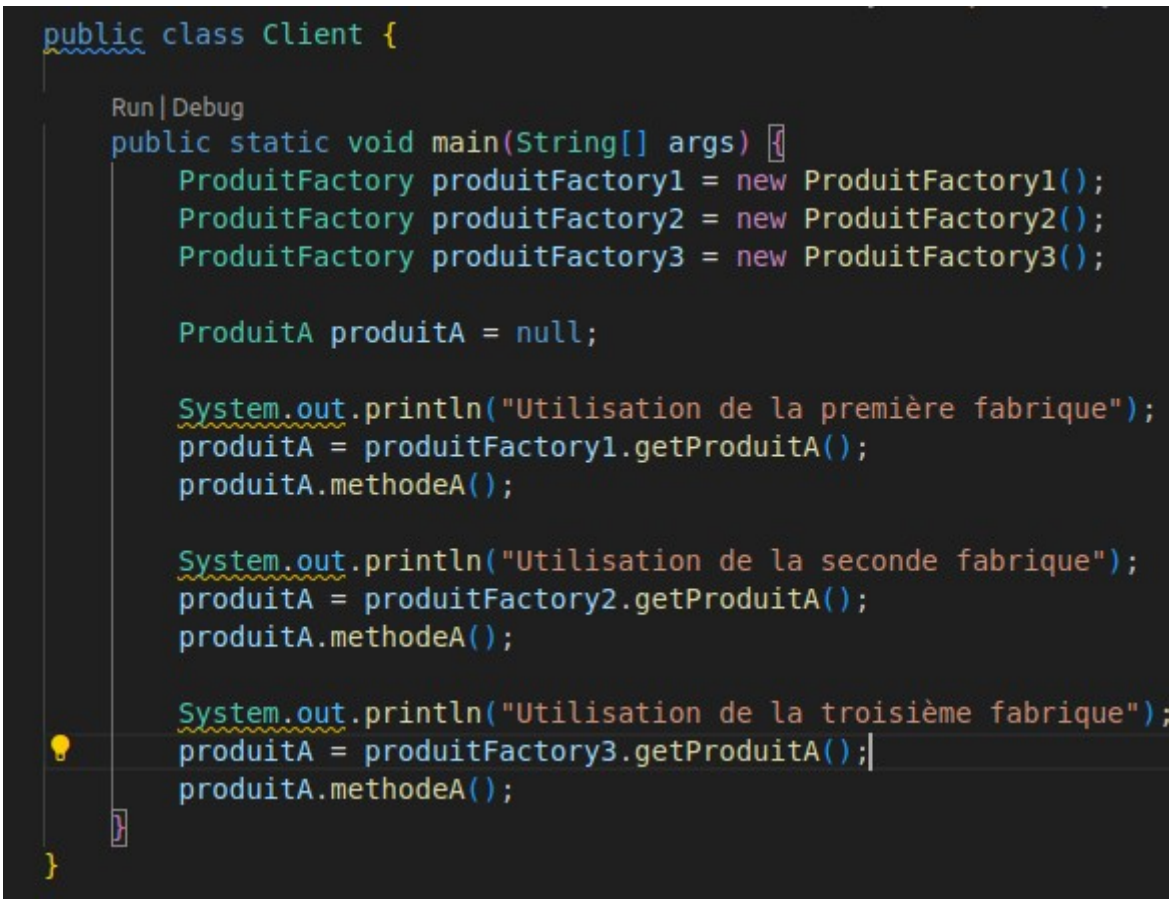
- ProduitFactory3 :

```java
public class ProduitFactory3 extends ProduitFactory{

    protected ProduitA createProduitA(){
        return new ProduitA3();
    }
}
```

- ProduitFactory :

```java
public class ProduitFactory {

    public static final int TYPE_PRODUITA1 = 1;
    public static final int TYPE_PRODUITA2 = 2;
    public static final int TYPE_PRODUITA3 = 3;

    public ProduitA geProduitA(int typeProduit){
        ProduitA produitA = null;

        switch (typeProduit) {
            case TYPE_PRODUITA1:
                produitA = new ProduitA1();
                break;
            case TYPE_PRODUITA2:
                produitA = new ProduitA2();
                break;
            case TYPE_PRODUITA3:
                produitA = new ProduitA3();
                break;
            default:
                throw new IllegalArgumentException("Type de produit inconnu");
        }
        return produitA;
    }
}
```

- Client :



```java
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();
    }
}
```