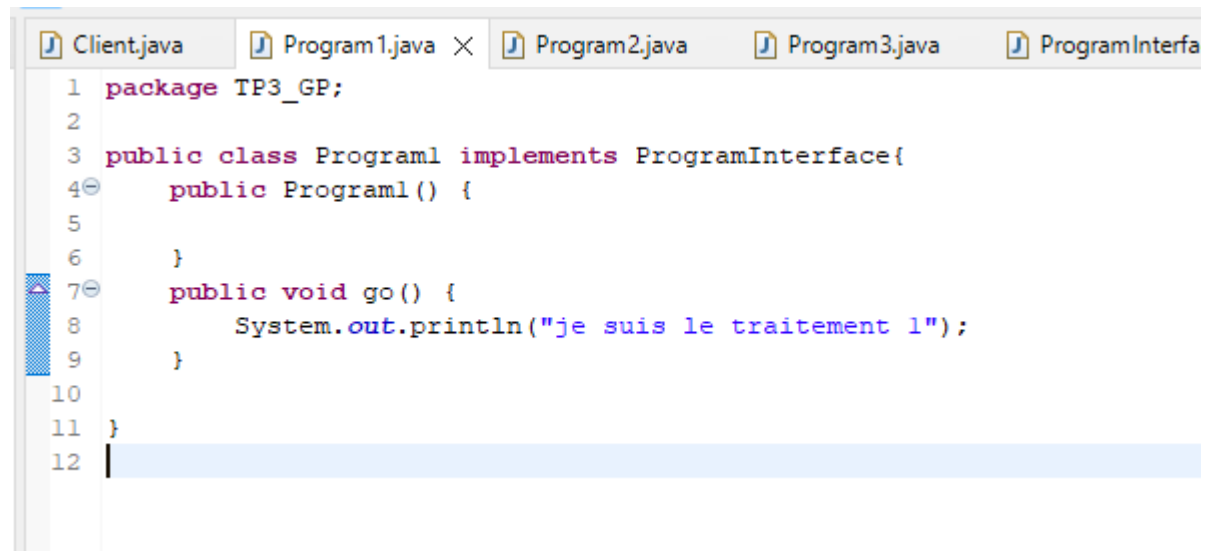
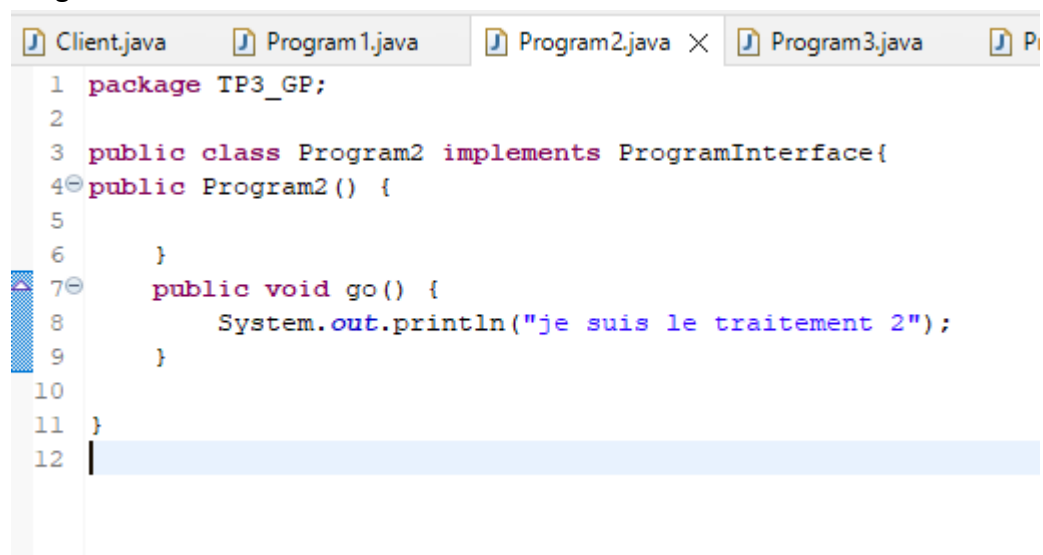


- Programme 1 :



```
Client.java Program1.java × Program2.java Program3.java ProgramInterfa
1 package TP3_GP;
2
3 public class Program1 implements ProgramInterface{
4     public Program1() {
5
6     }
7     public void go() {
8         System.out.println("je suis le traitement 1");
9     }
10
11 }
12
```

- Programme 2 :



```
Client.java Program1.java Program2.java × Program3.java P
1 package TP3_GP;
2
3 public class Program2 implements ProgramInterface{
4     public Program2() {
5
6     }
7     public void go() {
8         System.out.println("je suis le traitement 2");
9     }
10
11 }
12
```

- Programme 3 :

```

1 package TP3_GP;
2
3 public class Program3 implements ProgramInterface{
4 public Program3() {
5
6 }
7 public void go() {
8     System.out.println("je suis le traitement 3");
9 }
10
11 }
12

```

- Client :

The screenshot shows the Eclipse IDE with the 'Client.java' file open. The code defines a 'Client' class with three static methods: 'main', 'main2', and 'main3'. Each method creates an instance of 'Program1', 'Program2', or 'Program3' respectively and calls their 'go()' method. The console at the bottom shows the output of the program, which is 'Je suis le main1' followed by 'je suis le traitement 1'.

```

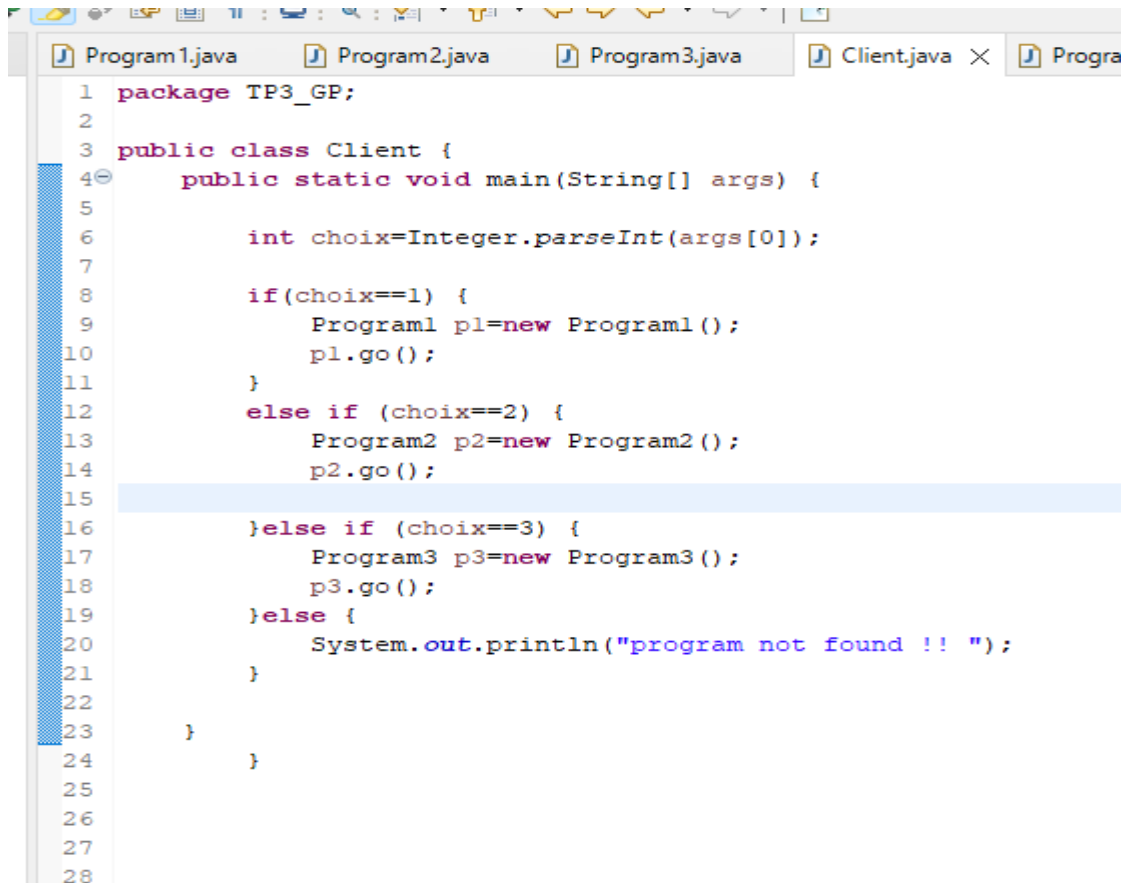
1 package TP3_GP;
2
3 public class Client {
4     public static void main(String[] args) {
5         Program1 p = new Program1();
6         System.out.println("Je suis le main1");
7         p.go();
8     }
9     public static void main2() {
10        Program2 p2 = new Program2();
11        System.out.println("Je suis le main2");
12        p2.go();
13    }
14    public static void main3() {
15        Program3 p3 = new Program3();
16        System.out.println("Je suis le main3");
17        p3.go();
18    }
19 }
20
21
22
23
24
25

```

Problems @ Javadoc Declaration Console × Terminal

<terminated> Client [Java Application] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot
Je suis le main1
je suis le traitement 1

- Classe client après la modification :



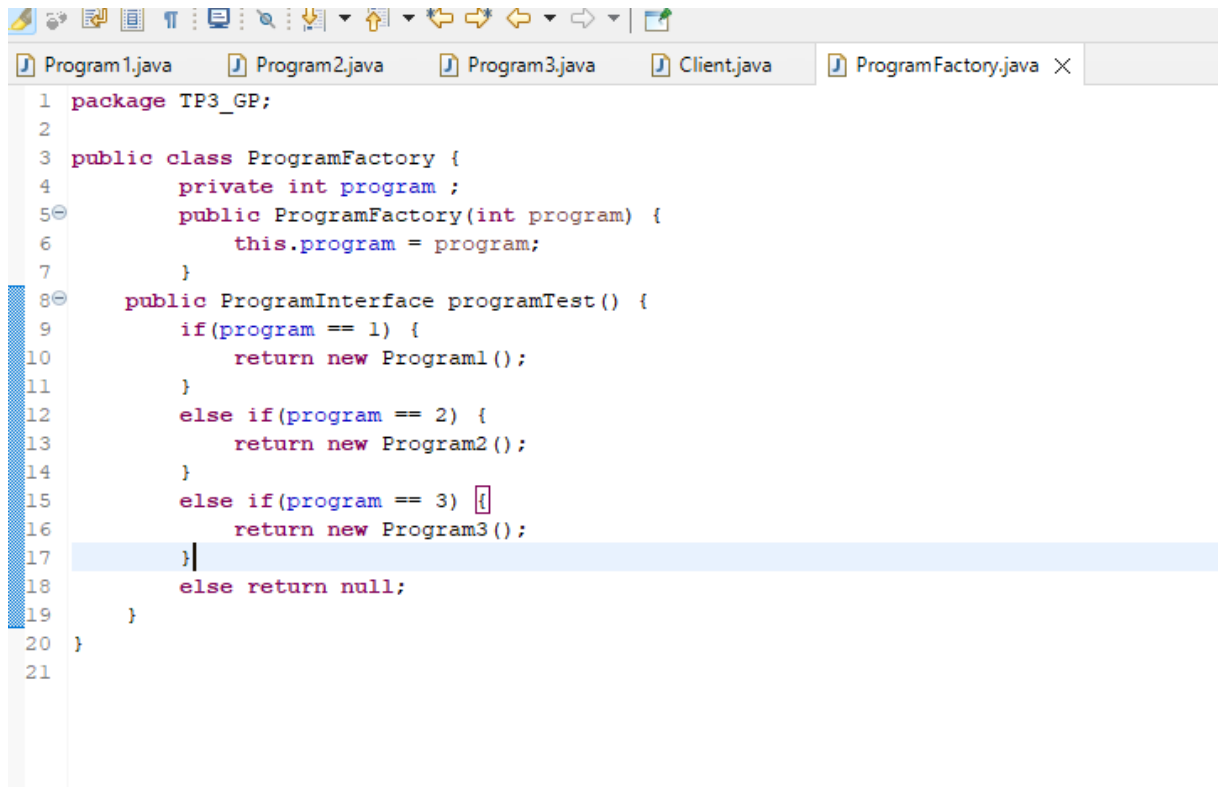
```
1 package TP3_GP;
2
3 public class Client {
4     public static void main(String[] args) {
5
6         int choix=Integer.parseInt(args[0]);
7
8         if(choix==1) {
9             Program1 p1=new Program1();
10            p1.go();
11        }
12        else if (choix==2) {
13            Program2 p2=new Program2();
14            p2.go();
15
16        }else if (choix==3) {
17            Program3 p3=new Program3();
18            p3.go();
19        }else {
20            System.out.println("program not found !! ");
21        }
22    }
23 }
24
25
26
27
28
```

Remarque :

On remarque que dans cette solution, le code client dépend directement des classes, concrètes Program1, Program2 et Program3

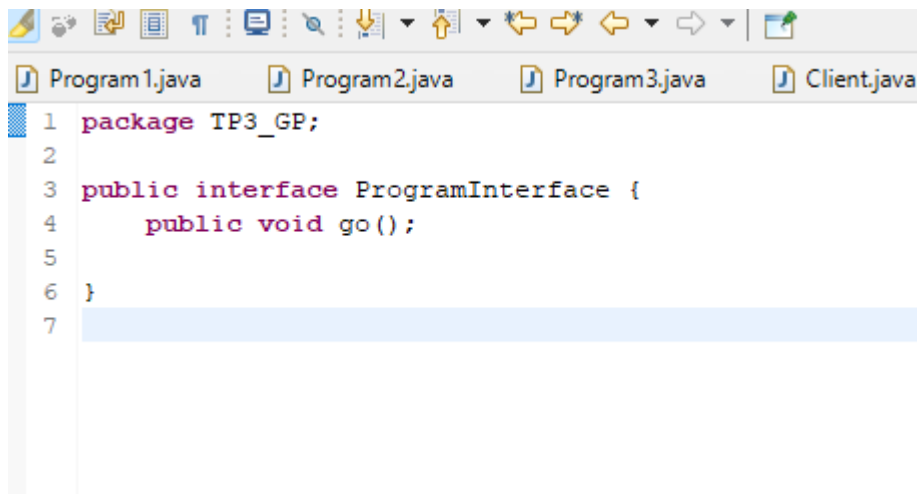
Cela crée une forte dépendance entre les composants et rend le système peu flexible et chaque ajout d'un nouveau programme nécessite la modification dans le code client ce qui augmente la complexité et rend la maintenance plus difficile.

- Class ProgramFactory :



```
1 package TP3_GP;
2
3 public class ProgramFactory {
4     private int program ;
5     public ProgramFactory(int program) {
6         this.program = program;
7     }
8     public ProgramInterface programTest() {
9         if(program == 1) {
10             return new Program1();
11         }
12         else if(program == 2) {
13             return new Program2();
14         }
15         else if(program == 3) {
16             return new Program3();
17         }
18         else return null;
19     }
20 }
21
```

- Program Interface :



```
1 package TP3_GP;
2
3 public interface ProgramInterface {
4     public void go();
5 }
6
7
```

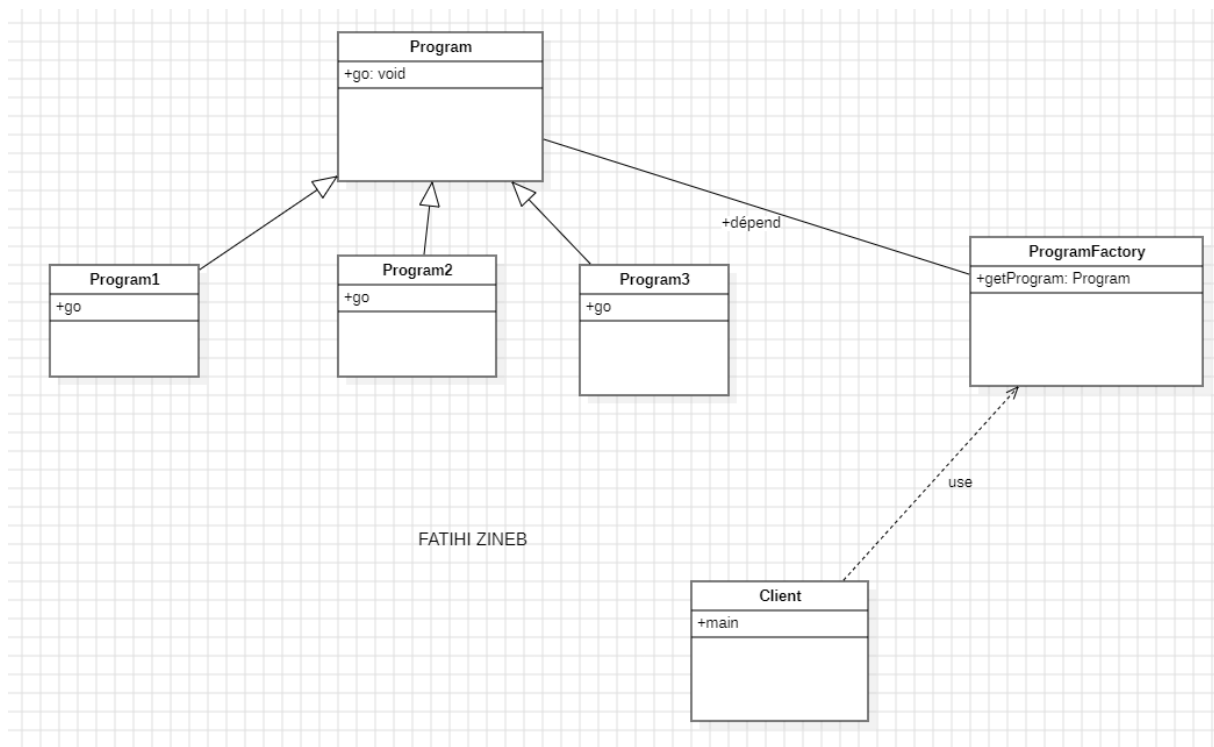
- Class main après Factory :

```

1 package TP3_GP;
2
3 public class Client {
4     public static void main(String[] args) {
5
6         int number = Integer.parseInt(args[0]);
7         ProgramFactory p = new ProgramFactory(number);
8         ProgramInterface program = p.programTest();
9         program.go();
10
11     }
12 }
13
14
15
16
17
18
19
20
21

```

- Diagramme de classe



- Ajout de programme 4 :
L'ajout de programme 4 a été simple grâce à la Factory. Il a suffi d'ajouter la nouvelle classe et de l'enregistrer dans la Factory sans modification dans la classe client, cela montre que le code est plus extensible, facile à maintenir et respecte le principe open/close