

Exercise 1:

Implementation

Listing 1: Example Java code

```
1  public class Database {
2      // Static variable to hold the single instance
3      private static Database name;
4
5      // Private constructor
6      private Database() {
7      }
8
9      // Public static method to provide access
10     public static Database getInstance() {
11         if (name == null) {
12             name = new Database();
13         }
14         return name;
15     }
16
17
18
19     public static void main(String[] args) {
20         Database obj1 = Database.getInstance();
21         Database obj2 = Database.getInstance();
22         System.out.println(obj1 == obj2); // true, same instance
23     }
24 }
25 }
```

Output

```
mac@MacBook-Air-de-Mac-2 src % javac Database.java
mac@MacBook-Air-de-Mac-2 src % java Database
true
```

Figure 1: screenshot of terminal

Exercise 2:

Part 1 : Naive solution

Listing 2: Example Java code

```

1 public class Client {
2     public static void main(String[] args) {
3         if (args.length == 0) {
4             System.out.println("Please pass 1, 2, or 3 as a command-line argument.");
5             return;
6         }
7
8         int x = Integer.parseInt(args[0]);
9         if(x==1){
10             Program1 p = new Program1();
11             System.out.println("I am main1");
12             p.go();
13         }
14         if(x==2){
15
16             Program2 p = new Program2();
17             System.out.println("I am main2");
18             p.go();
19         }
20         if(x==3){
21             Program3 p = new Program3();
22             System.out.println("I am main3");
23             p.go();
24         }
25
26     }
27
28
29
30
31 }
32
33 class Program1 {
34
35     public Program1() {
36
37     }
38
39     public void go() {
40         System.out.println("Je suis le traitement 1");
41     }
42
43
44 }
45
46 class Program2 {
47
48     public Program2() {
49
50     }
51
52     public void go() {
53         System.out.println("Je suis le traitement 2");
54     }
55
56
57 }

```

```

58 class Program3 {
59
60     public Program3() {
61
62     }
63
64     public void go() {
65         System.out.println("Je suis le traitement 3");
66     }
67
68 }

```

Output

```

mac@MacBook-Air-de-Mac-2 src % javac Client.java
mac@MacBook-Air-de-Mac-2 src % java Client 1
I am main1
Je suis le traitement 1
mac@MacBook-Air-de-Mac-2 src % javac Client.java
mac@MacBook-Air-de-Mac-2 src % java Client 3
I am main3
Je suis le traitement 3

```

Figure 2: screenshot of terminal

Part 2 : Apply design patterns

Listing 3: Example Java code

```

1 public class Client {
2     public static void main(String[] args) {
3         if (args.length == 0) {
4             System.out.println("Please enter program number (1, 2, 3, ...)");
5             return;
6         }
7
8         int x = Integer.parseInt(args[0]);
9         Program p = ProgramFactory.createProgram(x);
10
11        if (p != null) {
12            System.out.println("I am main" + x);

```

```

13         p.go();
14     } else {
15         System.out.println("Invalid program number.");
16     }
17 }
18
19
20
21
22 }
23 }
24
25 class ProgramFactory {
26     public static Program createProgram(int x) {
27         if (x == 1) {
28             Program p = new Program1();
29             return p;
30         }
31         if (x == 2) {
32             Program p = new Program2();
33             return p;
34         }
35         if (x == 3) {
36             Program p = new Program3();
37             return p;
38         }
39         return null;
40     }
41 }
42
43 }
44 abstract class Program {
45
46
47     public abstract void go();
48
49
50 }
51
52 class Program1 extends Program {
53
54     public void go() {
55         System.out.println("Je suis le traitement 1");
56     }
57
58 }
59
60 class Program2 extends Program {
61
62     public void go() {
63         System.out.println("Je suis le traitement 2");
64     }
65
66 }
67
68 class Program3 extends Program {
69

```

```
70     public void go() {
71         System.out.println("Je suis le traitement 3");
72     }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
```

Output

```
mac@MacBook-Air-de-Mac-2 src % java Client 3
I am main3
Je suis le traitement 3

mac@MacBook-Air-de-Mac-2 src % javac Client.java

mac@MacBook-Air-de-Mac-2 src % java Client 2
I am main2
Je suis le traitement 2
```

Figure 3: screenshot of terminal

Use Case Diagram

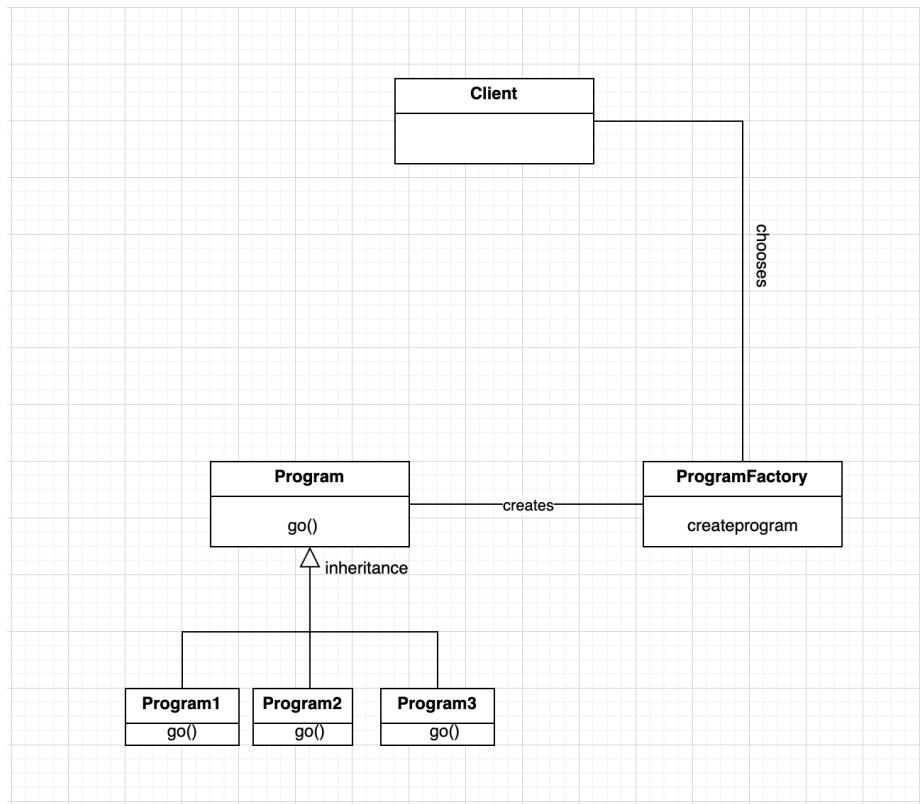


Figure 4: screenshot of terminal