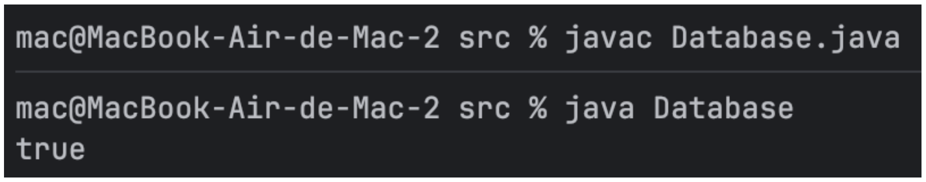
Exercise 1:

Implementation

Listing 1: Example Java code

```java
public class Database {
    // Static variable to hold the single instance
    private static Database name;

    // Private constructor
    private Database() {
    }

    // Public static method to provide access
    public static Database getInstance() {
        if (name == null) {
            name = new Database();
        }
        return name;
    }



    public static void main(String[] args) {
        Database obj1 = Database.getInstance();
        Database obj2 = Database.getInstance();
        System.out.println(obj1 == obj2); // true, same instance

    }
}
```

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:

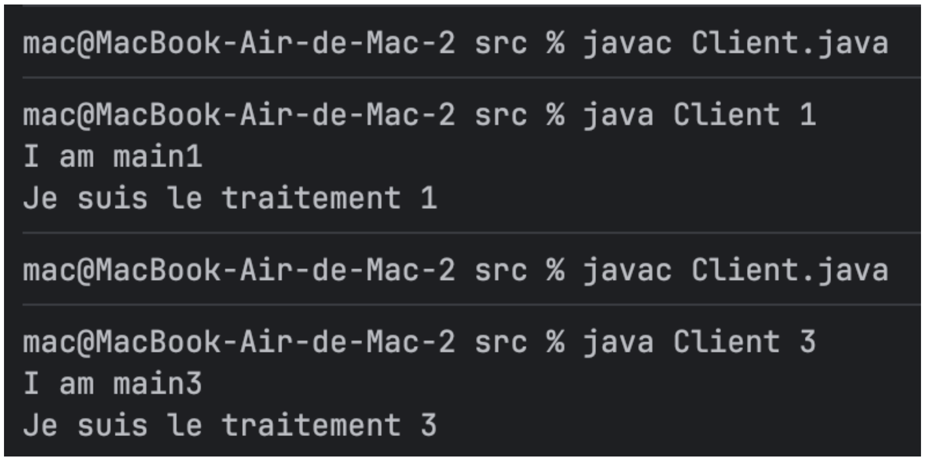
Part 1 : Naive solution

Listing 2: Example Java code

```java
public class Client {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please pass 1, 2, or 3 as a command-line argument.");
            return;
        }

        int x = Integer.parseInt(args[0]);
        if(x==1){
            Program1 p = new Program1();
            System.out.println("I am main1");
            p.go();
        }
        if(x==2){

            Program2 p = new Program2();
            System.out.println("I am main2");
            p.go();
        }
        if(x==3){
            Program3 p = new Program3();
            System.out.println("I am main3");
            p.go();
        }

    }




}

class Program1 {

    public Program1() {

    }

    public void go() {
        System.out.println("Je suis le traitement 1");
    }


}

class Program2 {

    public Program2() {

    }

    public void go() {
        System.out.println("Je suis le traitement 2");
    }


}
```

```java
class Program3 {

    public Program3() {

    }

    public void go() {
        System.out.println("Je suis le traitement 3");
    }


}
```

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
```

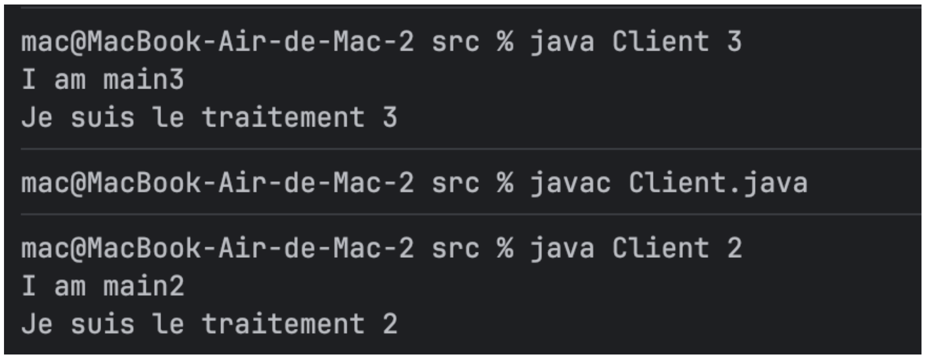
Figure 2: screenshot of terminal

Part 2 : Apply design patterns

Listing 3: Example Java code

```java
public class Client {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please enter program number (1, 2, 3, ...)");
            return;
        }

        int x = Integer.parseInt(args[0]);
        Program p = ProgramFactory.createProgram(x);

        if (p != null) {
            System.out.println("I am main" + x);
            p.go();
        } else {
            System.out.println("Invalid␣program␣number.");
        }
    }





}

class ProgramFactory {
    public static Program createProgram(int x) {
        if (x == 1) {
            Program p = new Program1();
            return p;
        }
        if (x == 2) {
            Program p = new Program2();
            return p;
        }
        if (x == 3) {
            Program p = new Program3();
            return p;
        }
        return null;

    }

}
abstract class Program {



    public abstract void go();


}
class Program1 extends Program {

    public void go() {
        System.out.println("Je␣suis␣le␣traitement␣1");
    }


}
class Program2 extends Program {

    public void go() {
        System.out.println("Je␣suis␣le␣traitement␣2");
    }


}
class Program3 extends Program {

    public void go() {
        System.out.println("Je suis le traitement 3");
    }


}




}
```

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

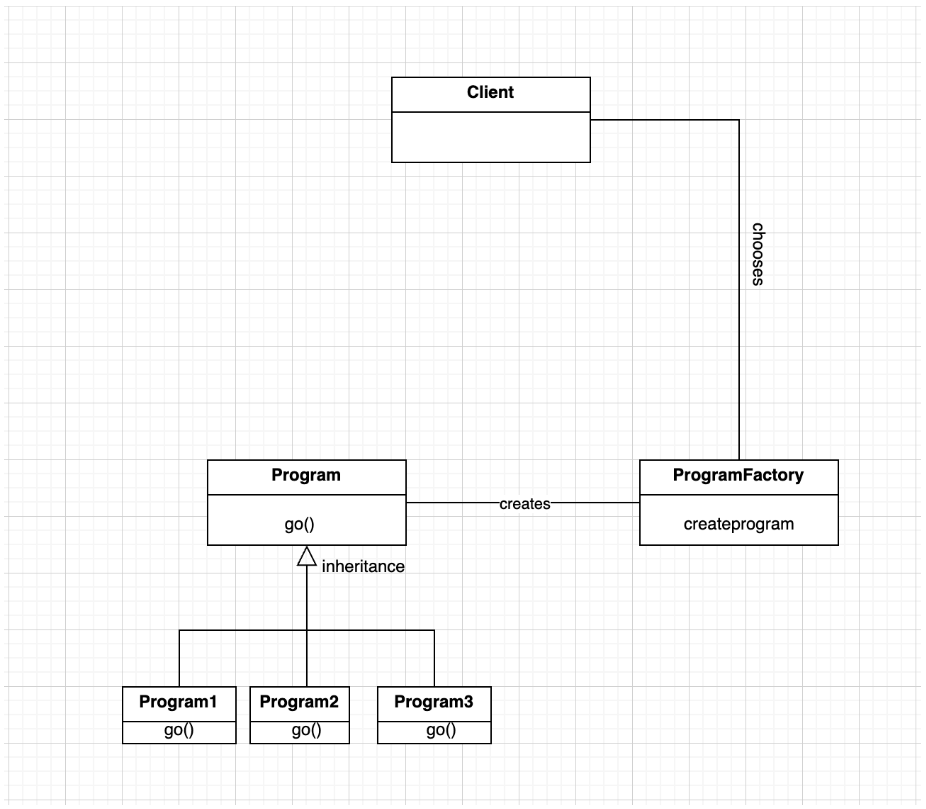


Figure 4: screenshot of terminal