DW N° 2

Exercise 1

Java Code:

```java
public class Pilot {

    private static Pilot instance = null;
    private int card;

    private Pilot(int card) {
       this.card = card;
       System.out.println("Pilot Instance");
    }

    public static Pilot getInstance() {

            synchronized(Pilot.class) {

            if (instance == null) {
                instance = new Pilot(100);

            }
        }

        return instance;
    }


}
```

```java
public class Main {

     public static void main(String[] args) {

        Runnable task = () -> {
         System.out.println("Running on thread: " + Thread.currentThread().getName());
       Pilot p1 =  Pilot.getInstance();
        };

        Thread thread1 = new Thread(task);
      Thread thread2 = new Thread(task);
      Thread thread3 = new Thread(task);

      thread1.start();
      thread2.start();
      thread3.start();
     }
}
```

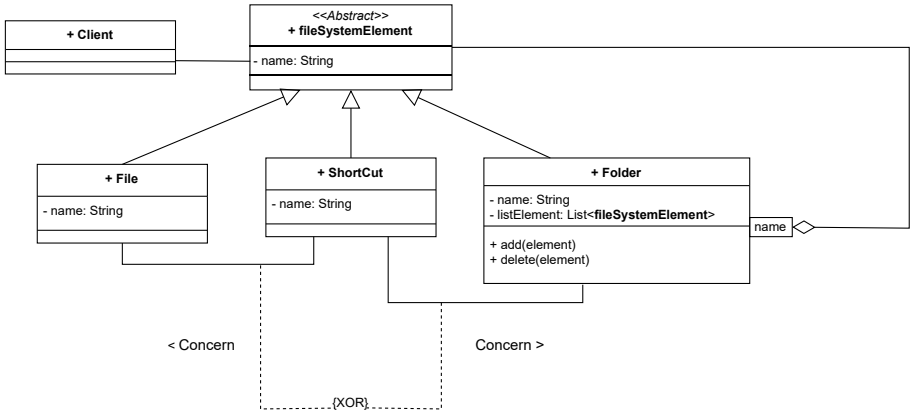
Output:

```
Running on thread: Thread-0
Running on thread: Thread-1
Pilot Instance
Running on thread: Thread-2
```

Note

Note that this isn't the most optimized solution , the detailed optimized solution is in the course.

Exercise 2



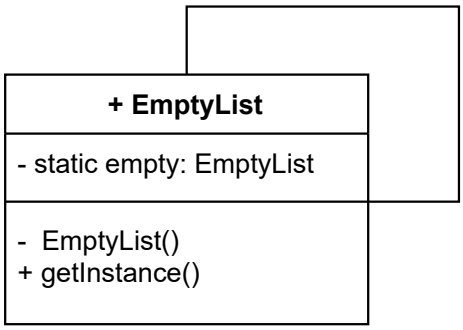
Explication

We used the **composite design pattern** because the file system is inherently nested and recursive: we have `shortcut` and `file` as the leaf elements, and `folder` as the complex nesting element.

A folder can contain either leaf elements or other folders, forming a hierarchical structure.

A `shortcut` can be associated with either a file or a folder, but never both at the same time this exclusivity is why we used the XOR (exclusive OR) operator.

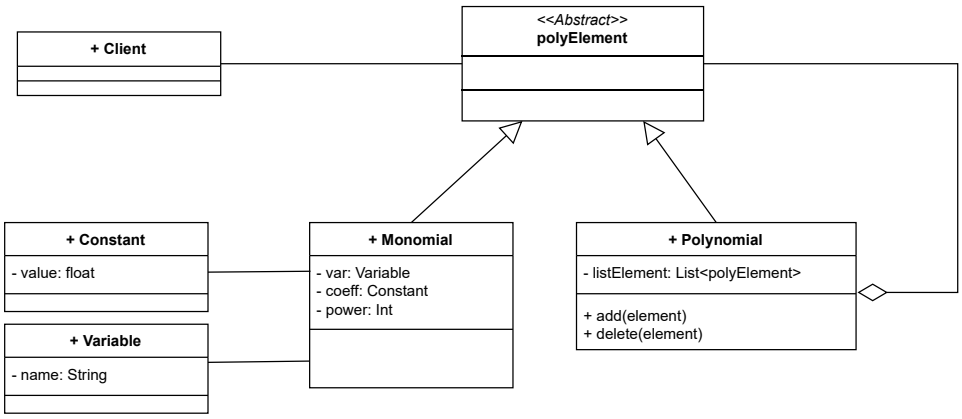
Exercise 3



Explication

We chose the `EmptyList` class as the **Singleton** class because it doesn't hold any data and simply serves as a utility to indicate whether the list is empty or not. Therefore, a single instance is sufficient.

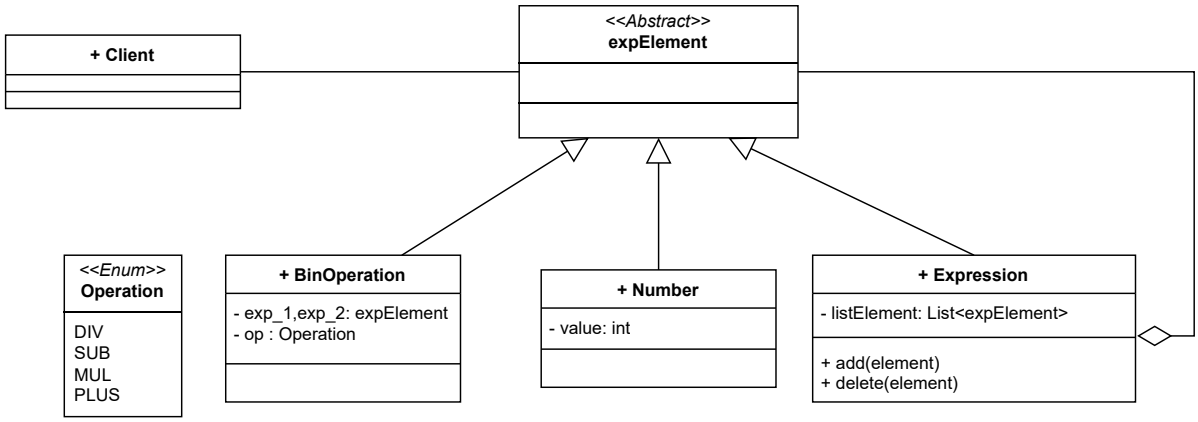
Exercise 4



Explication

We used the `Constant` and `Variable` classes to create the `Monomial` class, which holds the name of the variable, the power, and the coefficient. The `Monomial` is a leaf element, and the `Polynomial` is the complex element that holds a list of `Monomial`.

Exercise 5



Explication

The `Operation` is an `enum` class that represents the different arithmetic operations (`DIV`, `SUB`, `MUL`, `PLUS`).

The `Number` class represents an integer value and serves as a leaf element.

The `BinOperation` class is a composite element that holds two `expElement` instances (`exp_1` and `exp_2`), which can be either `Number`, another `BinOperation`. It also includes an `Operation` to define the arithmetic operator.

Finally, the `Expression` class is a composite element that holds a list of `expElement` instances via the `listElement`.