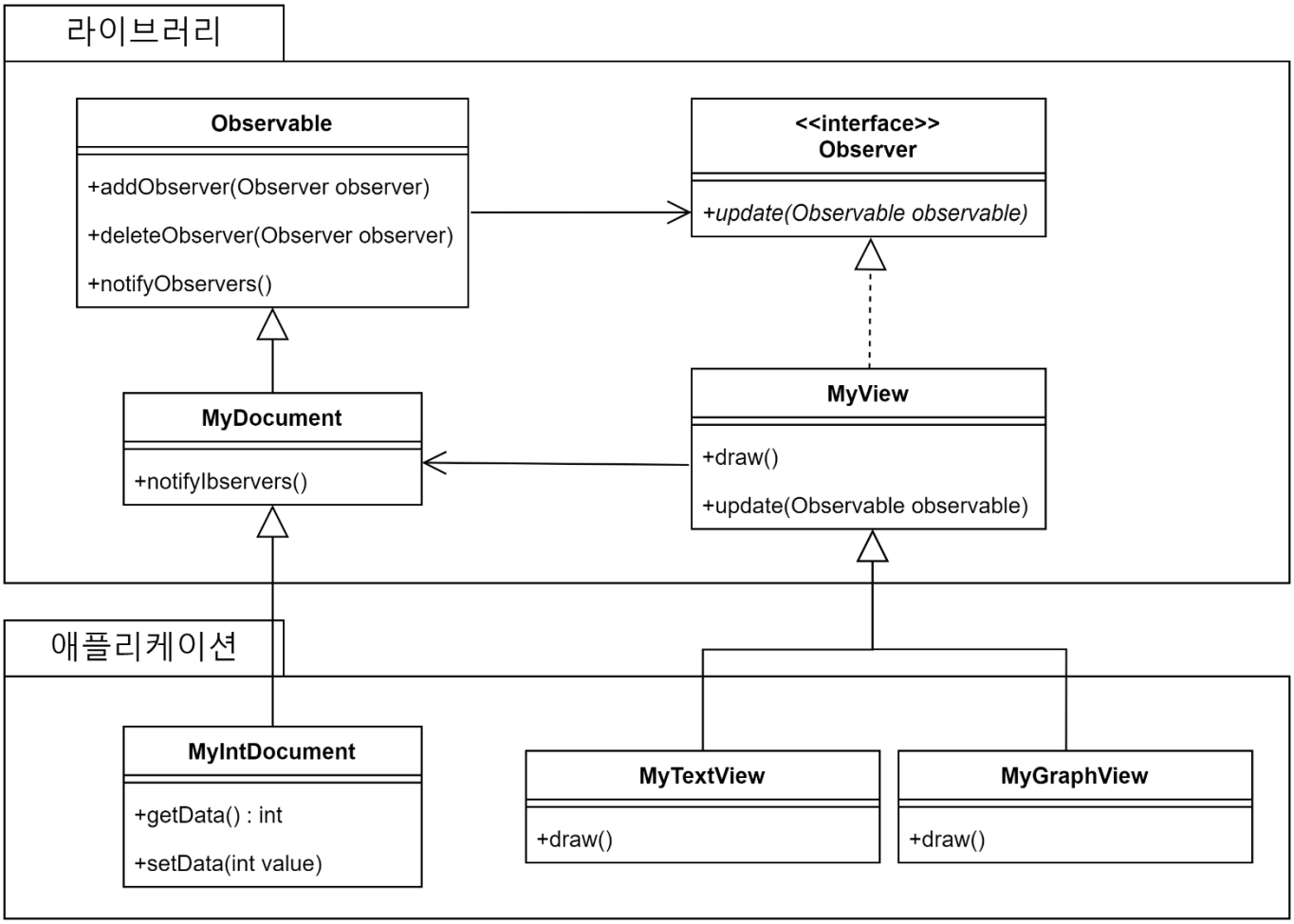
1) 개요


라이브러리
Observable
+addObserver(Observer observer)
+deleteObserver(Observer observer)
+notifyObservers()
<<interface>>
Observer
+update(Observable observable)
MyDocument
+notifyIbservers()
MyView
+draw()
+update(Observable observable)
애플리케이션
MyIntDocument
+getData() : int
+setData(int value)
MyTextView
+draw()
MyGraphView
+draw()

2) Observer.java

```java
package observer.e3;

interface Observer {
    void update(Observable observable);
}
```

3) Observable.java

```java
package observer.e3;

import java.util.ArrayList;
import java.util.List;

public class Observable {
    List<Observer> observers = new ArrayList<Observer>();

    void addObserver(Observer observer) {
        observers.add(observer);
    }

    void deleteObserver(Observer observer) {
        observers.remove(observer);
    }

    void notifyObservers() {
        for (int i = 0; i < observers.size(); ++i)
            observers.get(i).update(this);
    }
}
```

4) MyDocument.java

```
1  package observer.e3;
2
3  public class MyDocument extends Observable {
4
5      public void updateAllViews() {
6          notifyObservers();
7      }
8  }
```

5) MyIntDocument.java

```
1  package observer.e3;
2
3  public class MyIntDocument extends MyDocument {
4      private int data;
5
6      public int getData() {
7          return data;
8      }
9
10     public void setData(int i) {
11         data = i;
12         updateAllViews();
13     }
14 }
```

6) MyView.java

```
1  package observer.e3;
2
3  abstract class MyView implements Observer {
4      protected MyDocument document;
5
6      public MyView(MyDocument doc) {
7          document = doc;
8          doc.addObserver(this);
9      }
10
11     public abstract void draw();
12
13     @Override
14     public void update(Observable observable) {
15         draw();
16     }
17 }
```

7) MyTextView.java

```
1  package observer.e3;
2
3  public class MyTextView extends MyView {
4
5      public MyTextView(MyIntDocument doc) {
6          super(doc);
7      }
8
9      @Override
10     public void draw() {
11         int data = ((MyIntDocument)document).getData();
12         System.out.printf("View1: %dWn", data);
13     }
14 };
```

8) MyGraphView.java

```
1  package observer.e3;
2
3  public class MyGraphView extends MyView {
4
5      public MyGraphView(MyIntDocument doc) {
6          super(doc);
7      }
8
9      @Override
10     public void draw() {
11         int data = ((MyIntDocument)document).getData();
12         System.out.printf("View2: ");
13         for (int i = 0; i < data; ++i)
14             System.out.print('*');
15         System.out.print('Wn');
16     }
17 }
```

9) Example3.java

```
1  package observer.e3;
2
3  import java.util.Scanner;
4
5  public class Example3 {
6
7      public static void main(String[] args) {
8          try (Scanner scanner = new Scanner(System.in)) {
9              MyIntDocument doc = new MyIntDocument();
10             new MyTextView(doc);
11             new MyGraphView(doc);
12
13             for (;;) {
14                 int n;
15                 System.out.print("Wn set document data (0 = quit) : ");
16                 n = scanner.nextInt();
17                 if (n <= 0)
18                     break;
19                 doc.setData(n);
20             }
21         }
22     }
23
24 }
```

클래스 라이브러리
    Observer, Observable, MyDocument, MyView

애플리케이션
    MyIntDocument, MyTextView, MyGraphView, Example3

클래스 라이브러리 소스코드는 유지보수 대상이 아니고, 재사용 대상임.
애플리케이션 소스 코드들 사이의 참조 의존성을 평가하시오.

공통 코드를 부모 클래스로 추출하여 공유하도록 구현한 이 document, view 구조는
어떤 패턴인가?