Model-View-Controller Pattern

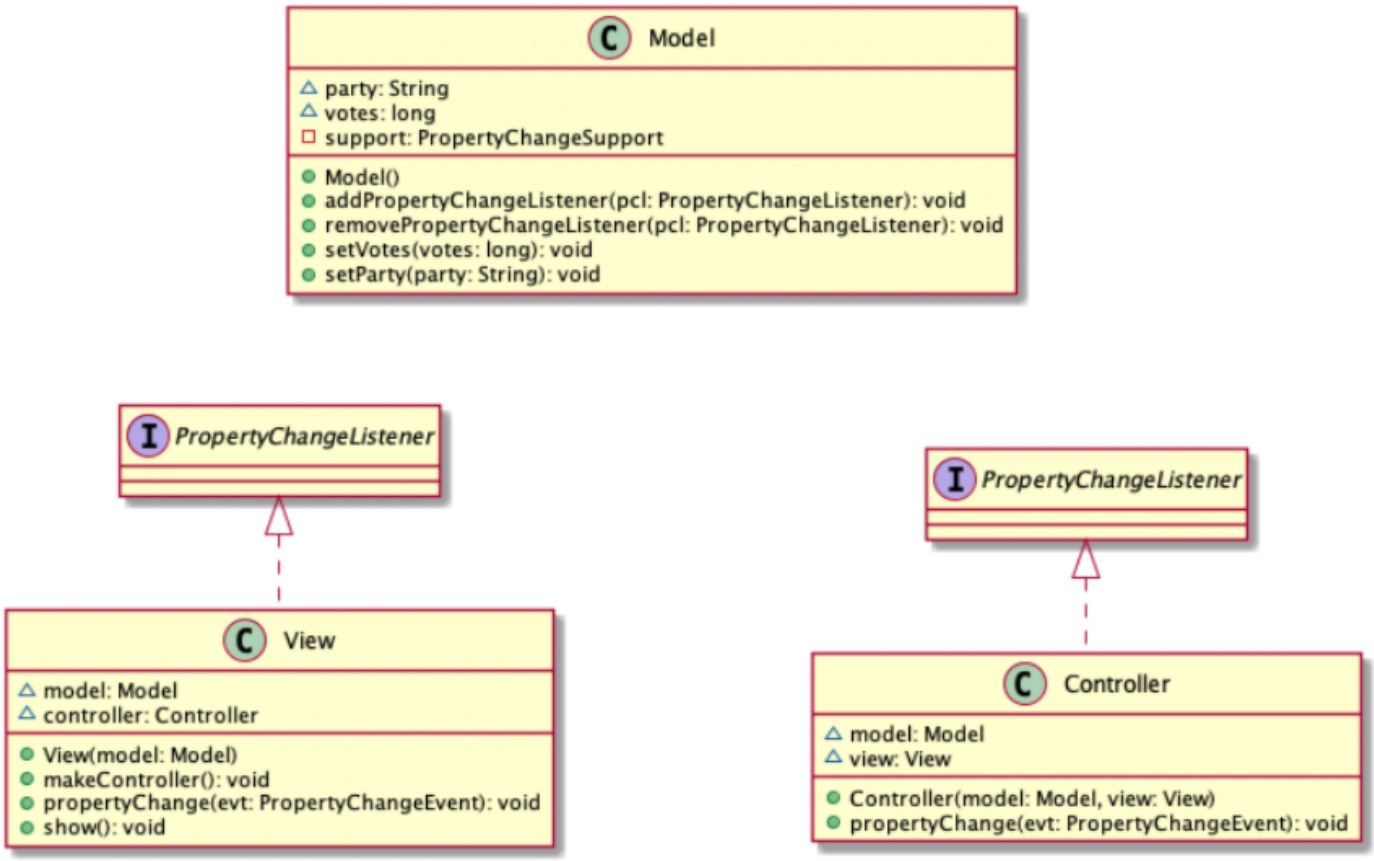
1. Model-View-Controller 패턴 [1]

애플리케이션의 관심사항을 다음과 같이 나누기 위해서 사용한다.

- Model: 모델은 데이터를 유지하는 객체를 의미하며, 데이터가 변경에 대한 업데이터 처리 로직을 포함
- View: 뷰는 모델이 포함한 데이터에 대한 가시화를 담당한다.
- Controller: 컨트롤러는 모델과 뷰에 동시에 작동한다. 모델의 데이터 흐름을 처리하며, 데이터가 변경될 때마다 뷰를 갱신한다. 뷰와 모델의 분리를 유지한다.

2. Model-View-Controller 패턴 구조 [1]

Model은 Party와 Votes라는 두 개의 값을 유지한다. 이 두 개의 값을 변경하게 되면 View와 Controller가 이를 감지한다. View에서는 업데이트한 값을 출력한다. 현재 Controller에서는 감지한 결과만 출력한다.



3. Model-View-Controller 패턴 사용

MVC 패턴을 사용하는 메인 함수는 다음과 같다.

```
public class Test {
        public static void main(String[] args) {
                Model model = new Model();
                View view = new View(model);
                model.setVotes(3);
                model.setParty("black");
        }
}
```

4. Model-View-Controller 패턴 활용

(1) 현재 모델의 party는 "black"으로 votes는 "3"으로 설정되어 있다. MVC 구조를 유지하면서 사용자 입력으로 값을 받도록 하자 (사용자 입력은 Controller에서 받도록 하자).

(2) 사용자 입력으로 값이 바뀔 때마다, 상세 정보를 재출력하도록 하자.

(3) 어떻게 MVC 구조로 상기와 같은 작업이 가능한지 코드를 분석하여 설명해 보자.

(4) 실제로는 콘솔보다는 GUI 쪽에서 MVC 구조를 사용한다. 해당 구조가 어떻게 GUI로 확장할 수 있는지 예를 들어 설명해 보자.

5. References

[1] MVC Pattern - https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

[2] The Observer Pattern in Java, https://www.baeldung.com/java-observer-pattern