Façade Design Pattern

Sunday, December 13, 2020 11:15 PM

In our project phase 1, in the controllers package, it contains Conference.java, which uses the Façade Design Pattern.

Because the Façade Design Pattern solves the following problem:

① This class is responsible to multiple "actors".

▷ Since Conference.java is the interface to manipulate the whole system, it keeps the program running, it loads and control the flow of the whole program, so we cannot avoid having multiple "actors" in it.

② We want to encapsulate the code so that it interacts with individual actors.

▷ If we didn't use this design pattern, we would have violated the Single-responsibility-principle of SOLID.

③ We want a simplified interface to a more complex subsystem

▷ We need to put them in separate variables and put those as variables in the Façade, so the implementing details is not contained in the Conference.java.

After using the façade design pattern, although Conference.java contains all the flow of control of running the whole program, the body of the class is not heavy, because we've created individual classes that each interact with only one actor, while the Façade class still has roughly the same responsibilities as before. By delegating each responsibility to the individual classes, (i.e., the Façade object contains references to each individual class) We preserve the open-closed principle of SOLID.
The Façade contains all the variables to do all the work, but the rest of the program cannot access them directly, this encapsulates whatever is in the Façade.

```
public class Conference {
    private UserManager userManager;
    private AttendeeManager attendeeManager;
    private Organizer ...;
    private EventsController eventsController;
    private SaveConversation saveConversation;
    private FileReadWriter fileReadWriter = new FileReadWriter();

    public void run() {
        // initialize all managers
        ;
    }
    public void start() {
        // connect to Gateway, set up database
    }
        private void iteration() {
            ;
        }
        ;
}
```

These variables will do all the work for this Façade.

The different responsibilities of the Façade class are each encapsulated inside a class variable of the Façade

By Jiayi Su, please feel free to contact me if you spot an error, or you think there is something to improve.

I am happy to improve it and make it better:
My email address is: sjy.su@mail.utoronto.ca