README:

The group 34

s3571517 Yaoxuan Liu
s3596621 Jinming Liu

We mainly used Facade and Builder for this project. Using the builder pattern, it allows the client to avoid having to know the details of the product's internal composition. Also, the specific builder classes are independent of each other, which means it can gradually refine the construction process without any impact on other modules. On the other hand, it is easy to expand. As shown in our code, the transaction types can be built as “Deposit”, ”Withdrawal”, ”Transfer” and “Service charge”. Therefore, we choose the Builder patter.

In addition, we used the facade pattern, which is intended to hide the components of subsystem to simplify the interface, reduce the number of objects handed by customers and improve the subsystem to be easier. Besides, the facade pattern realizes the loose coupling relationship between the subsystem and the customer, while the functional components inside the subsystem are tightly coupled. Loose coupling allows the component changes of a subsystem not to affect its customers.
After the role of facade shows up, the user only needs to interact with it. The complex logical relationship between the user and the subsystem is implemented by facade. Furthermore, changes to one subsystem have no effect on the others, and internal changes to the subsystem do not affect the facade objects.
On account of the program there are many subsystems like ATM, transfer, check statements. We need to keep each of the subsystems operating normally and the changes to each one would only have influence to itself. This is why we choose facade pattern.