**Banking System**

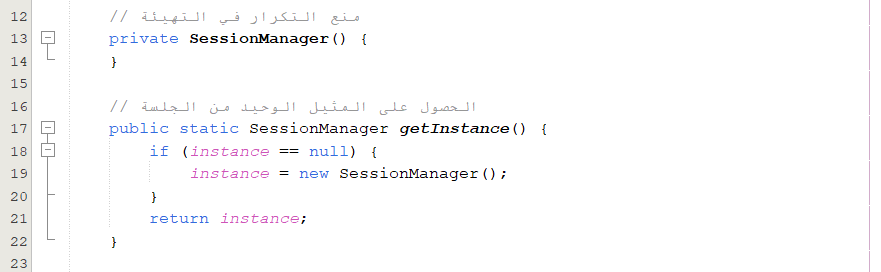
Description:

A desktop application for managing bank accounts, transactions, customer profiles, and loans

Design Patterns:

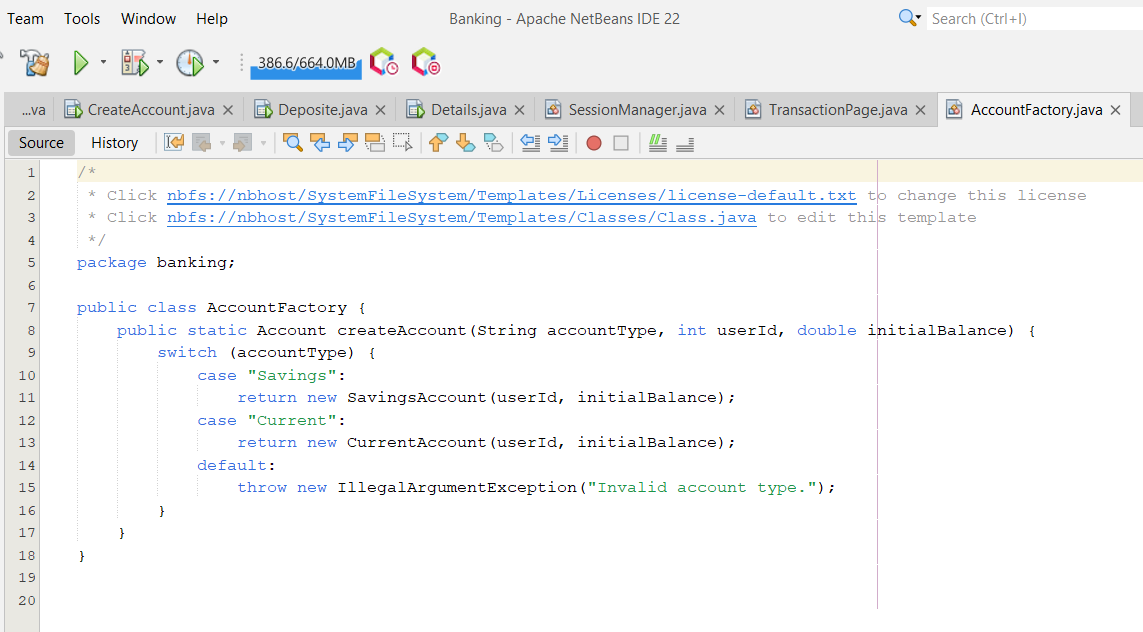
### 1-Singleton Pattern:

The Singleton pattern ensures that a class has only one instance, and it provides a global point of access to that instance:



2-Factory Pattern:

The factory method centralizes the object creation logic for different account types, which simplifies the code in other parts of the application. Clients (other classes) don't need to know about the specific classes (SavingsAccount, CurrentAccount), just the common Account interface:



3-Observer Patter

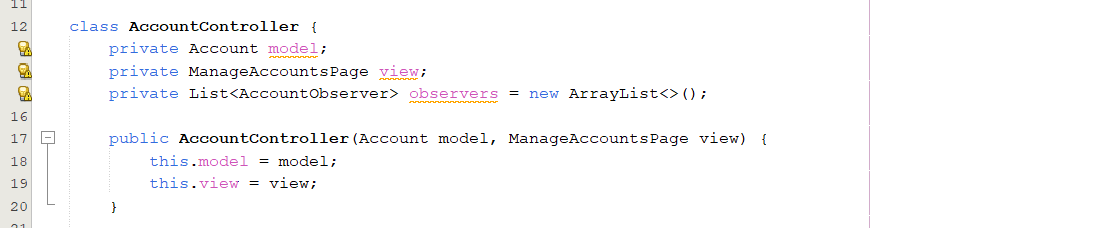
The Observer Pattern allows multiple components (observers) to automatically get updates whenever an object (subject) they are observing changes its state:

-When the model (Account) changes, balance is updated or an account is deleted the controller notifies all observers by calling the notifyObservers() method



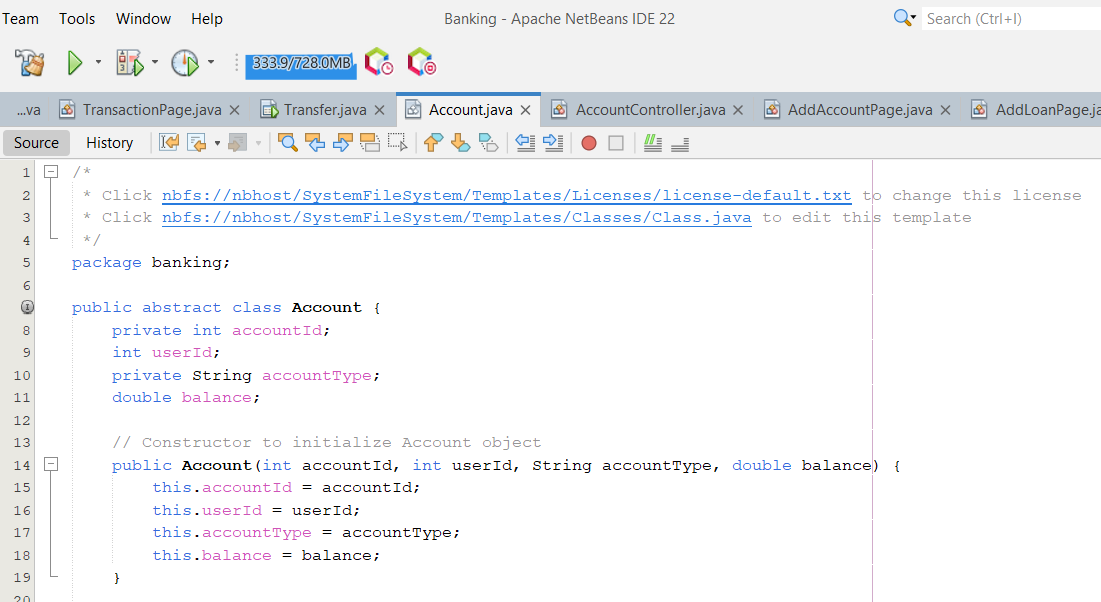
4- Model-View-Controller (MVC) Pattern:

The MVC design pattern separates an application into three interconnected components: Model (data), View (UI), and Controller (business logic):



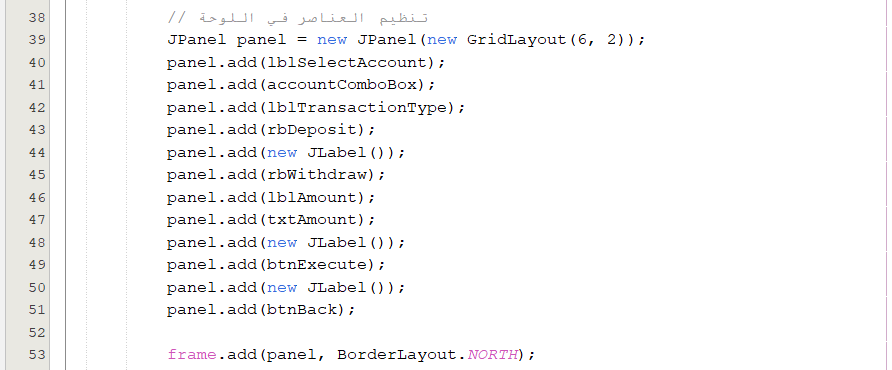
5-Template Method pattern

The Template Method pattern defines the skeleton of an algorithm in a base class (here, the Account class), allowing subclasses to override certain steps of the algorithm without changing the algorithm's structure.



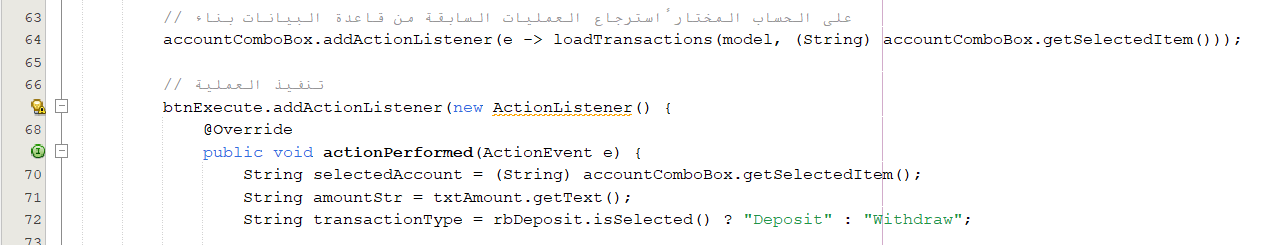
6-Builder Pattern:

The GUI is constructed incrementally using components like JPanel, JComboBox, JTable, and JScrollPane.



7-Command Pattern:

The event-handling logic encapsulates operations (e.g., executing a transaction or updating the table) into commands triggered by user actions.



Names:

1- Maria Kamal Bebawy 21-00974

2- Mary Gamal Anwer 21-01041

3- Mariam Benyamin Tawfik 21-01007

4- Jellan Romany George 21-00820

5- Gehad Fathy 21-00722

6- Mohamed Mamdouh Abdelbary 21-01182