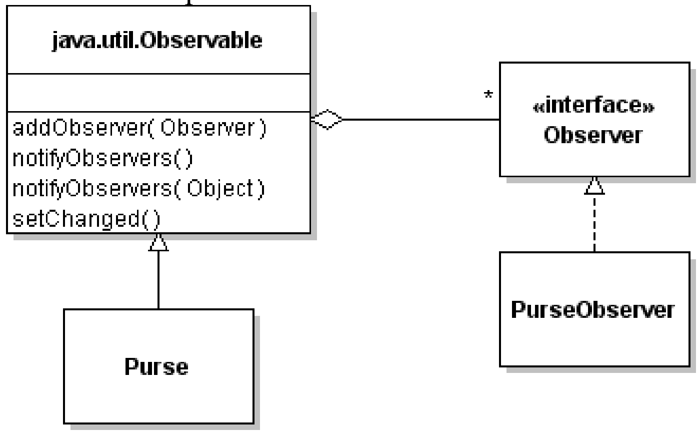
Objectives	Practice applying the Observer Pattern.
What to Submit	Commit your work as part of the Purse project. Add a new TAG named LAB6 for this iteration of the Purse.

Problem 1: Create an Observable Purse

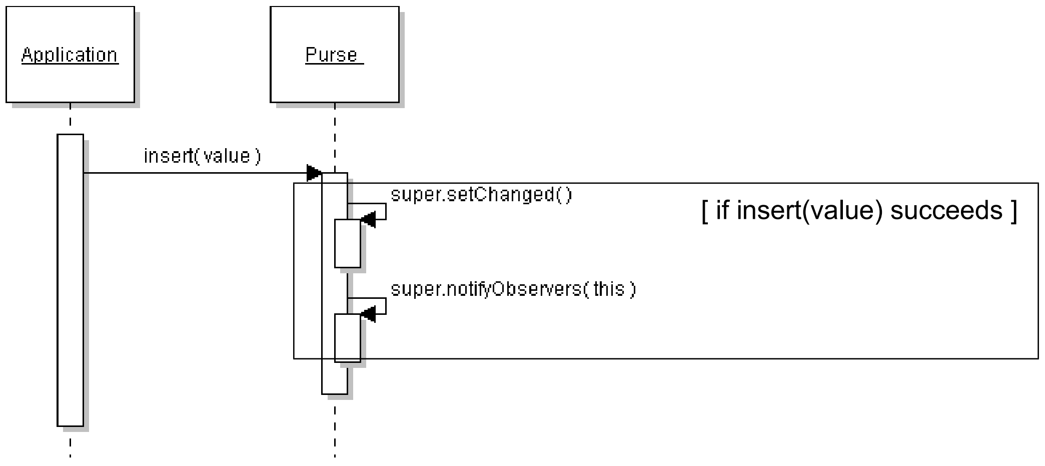
Implement the Observer Pattern for the coin purse, so that other objects can be notified when the state of the Purse changes. Apply the Observer pattern to the Purse like this:



The Purse has to notify observers when the state of the Purse changes.

What methods can change the *state* of the Purse? ________________________________

When these methods are invoked the Purse should notify the observers if something changes (but not if there is no change). The sequence of actions looks like this:



1. Modify the Purse class so it extends Observable as shown above.

2. Modify each method that change the contents of the Purse so they will notify observers when the purse changes. For example:

```
public boolean insert( Valuable val ) {
     do some work
     if success then:
     // notify the observers
         super.setChanged();
         super.notifyObservers( this ); // paramater is optional
}
```

3. Be careful that you **do not** notify observers when other methods are called (like getBalance). This can create an infinite loop: Purse notifies observer and observer calls a Purse method that causes the Purse to notify observers *again*, and observer calls the Purse method *again*, etc.

4. Run your application and verify that it still works the same as before.

Problem 2: Create Graphical Observers

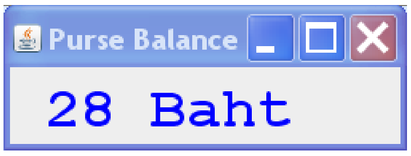
Create 2 graphical observers that shows the status of the purse in a window.

The Observers must *implement* the Observer interface. The update( ) method should update the user interface using information from the Purse when it is invoked.
Here is a console example (your code should use GUI components):

```
public class PurseObserver implements Observer {
    public PurseObserver( ) {
        // no reference to purse necessary
    }
    /** update receives notification from the purse */
    public void update(Observable subject, Object info) {
        if (subject instanceof Purse) {
            Purse purse = (Purse)subject;
            int balance = purse.getBalance();
            System.out.println("Balance is: " + balance);
        }
        if (info != null) System.out.println( info );
    }
}
```

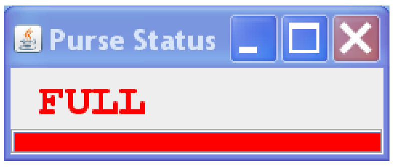
Another way to write an Observer is to give it a *reference to Purse* in the constructor, rather than use the parameters to update() to get the Purse. For this problems, its not necessary. But remember it.

2.1 *Purse Balance Observer:* Write another observer to display the balance in the Purse.

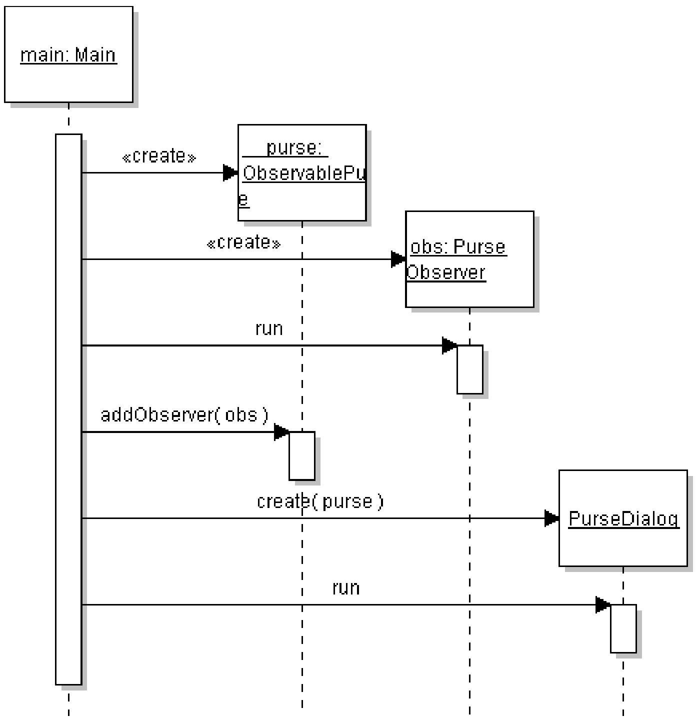


2.2 *Purse Status Observer*: Write a graphical observer that displays "FULL", "EMPTY", or number of coins in the purse (when not full or empty).

Use a JProgressBar to show the % full, and a text box to display "FULL", "EMPTY", or number of items.



2.3 Modify the Main class to create the observers and add Observers to the Purse. Main should create a Purse and Observers, and add observers to the Purse.
Also create a user dialog and run it.


main: Main
«create»
purse: ObservablePurse
«create»
obs: Purse Observer
run
addObserver(obs)
create(purse)
PurseDialog
run