

CMPE 202

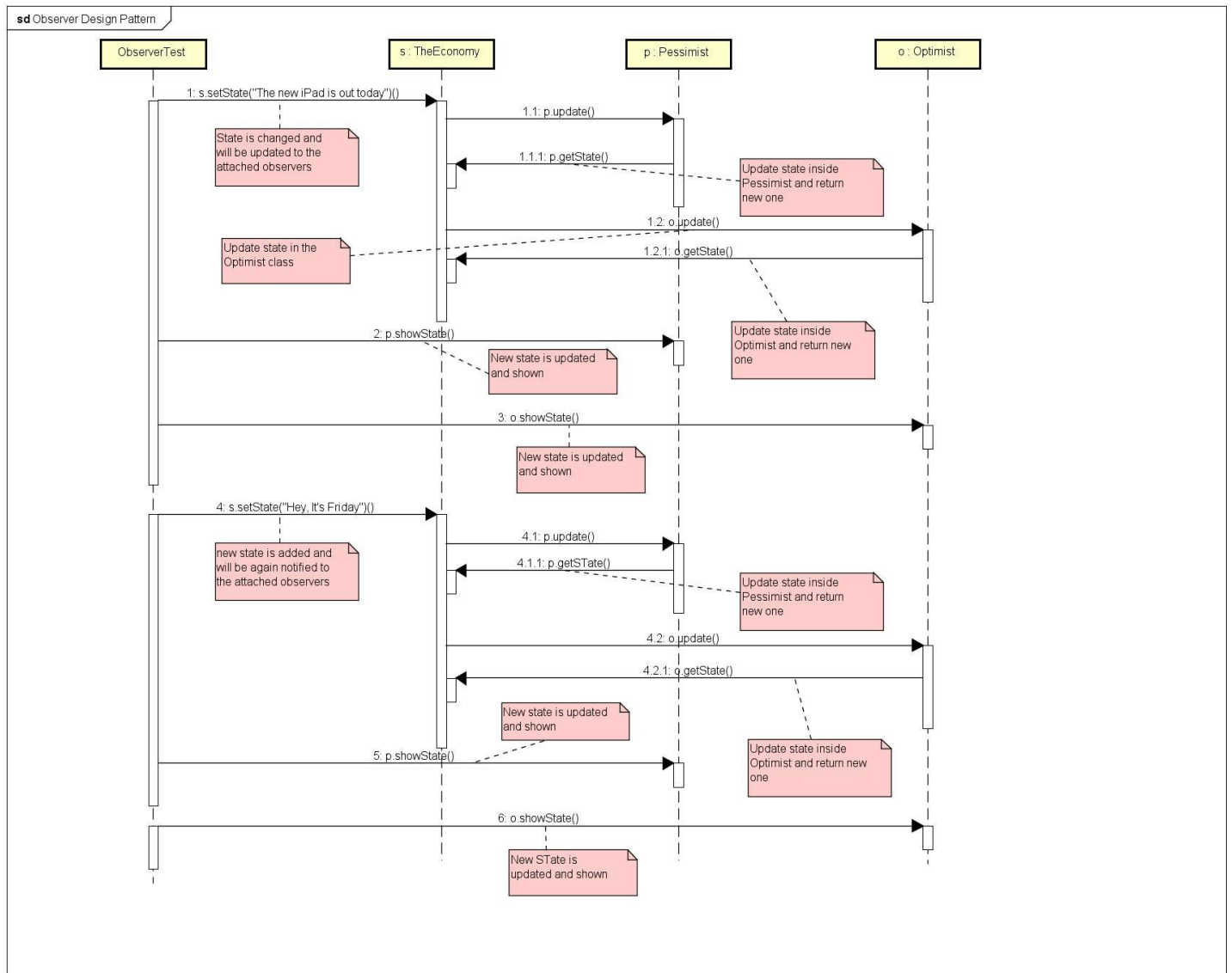
LAB 7

Observer Pattern Sequence Diagram

Name: Pavan Kothawade

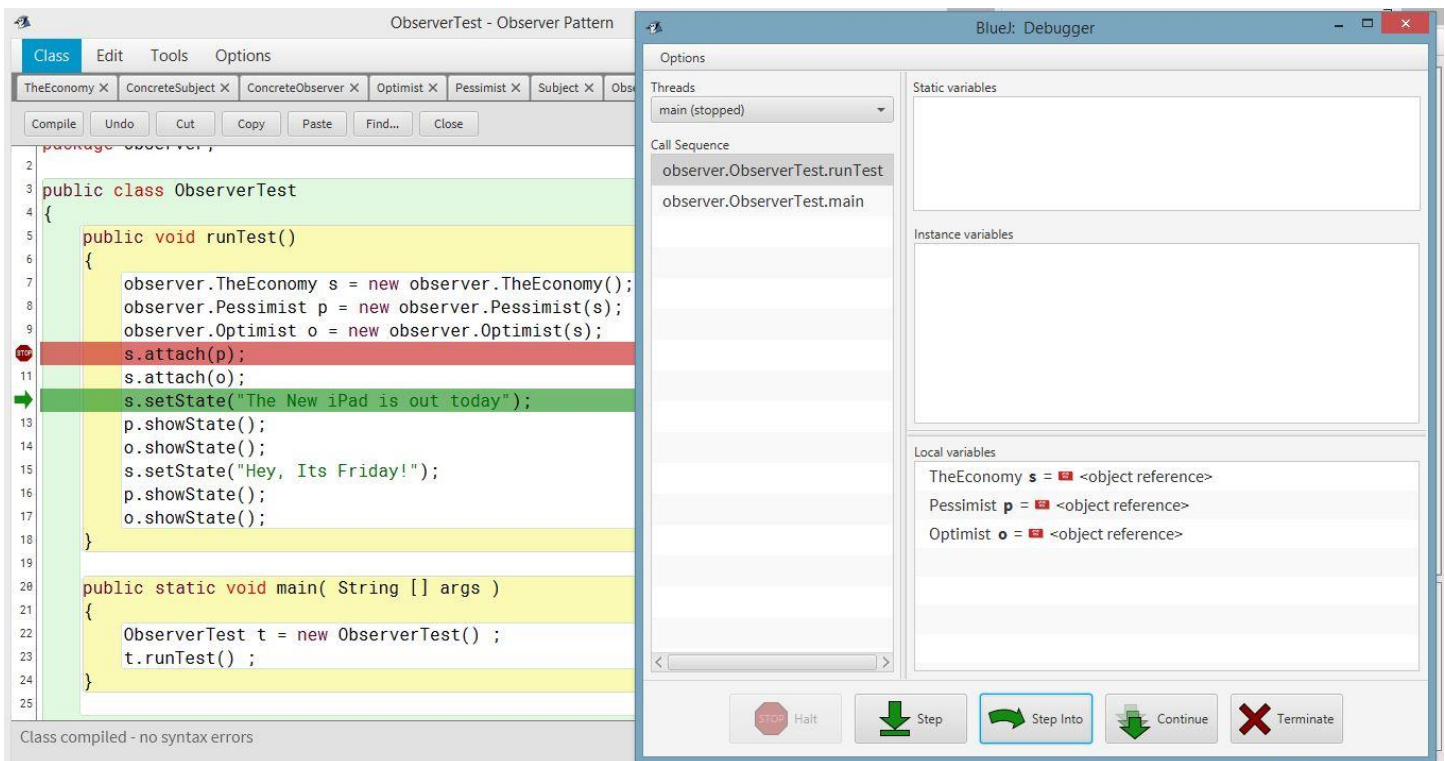
SJSU ID: 012169690

Sequence Diagram

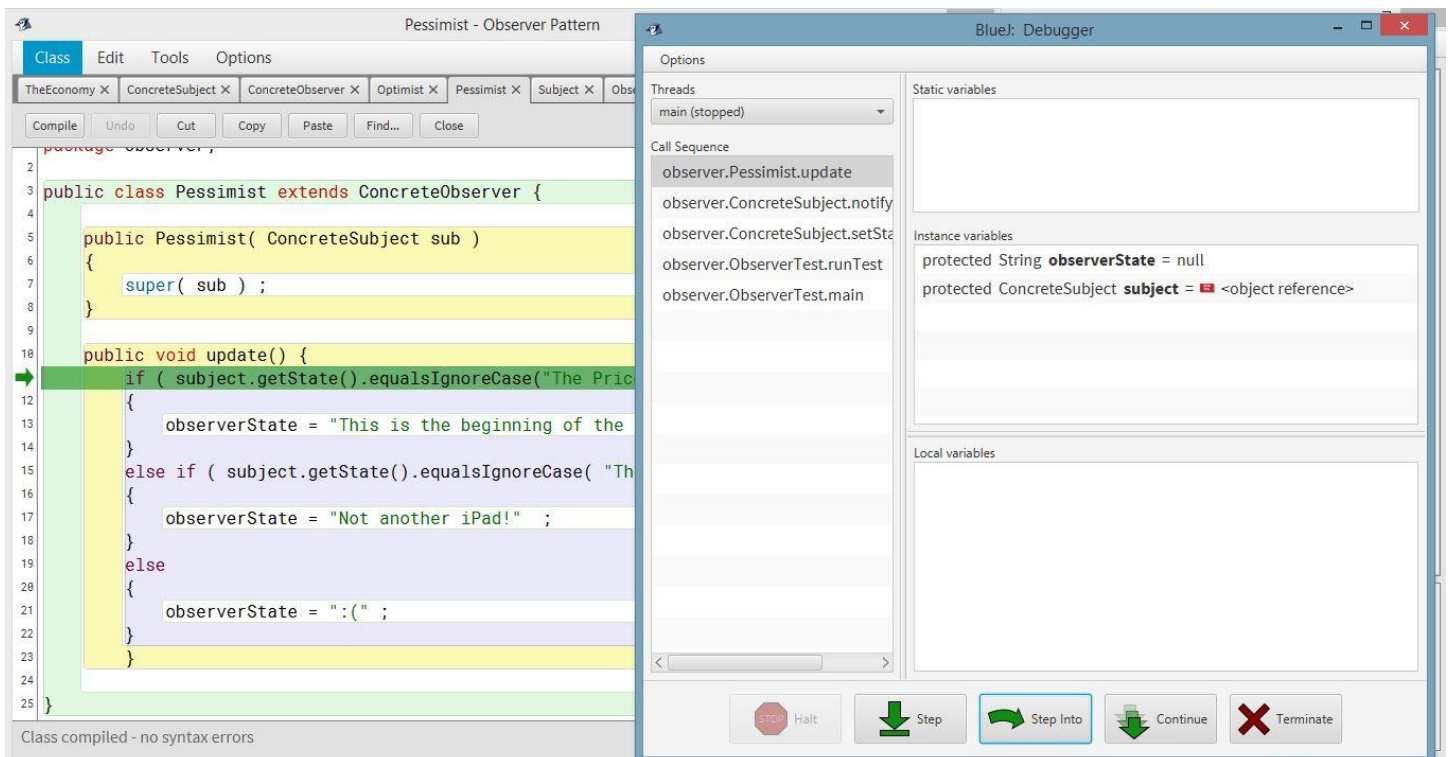


powered by Astah

1: s.setState("The new iPad is out today")- Set new state to TheEconomy class



1.1 p.update()- Update state in pessimist class



1.1.1 p.getState()-Retrieve new state of Pessimist class

The screenshot shows the BlueJ IDE with the `ConcreteSubject` class open. The class is part of the `Observer Pattern` and contains several methods. The `getState()` method is highlighted, showing it returns `subjectState`. The debugger is open on the right, showing the `main` thread stopped at the `observer.ConcreteSubject.getState` call. The instance variables section shows `subjectState` as `"The New iPad is out today"` and `observers` as an `ArrayList<Observer>` object reference.

```
private ArrayList<Observer> observers = new ArrayList<>();

public String getState() {
    return subjectState;
}

public void setState(String status) {
    subjectState = status;
    notifyObservers();
}

public void attach(Observer obj) {
    observers.add(obj);
}

public void detach(Observer obj) {
    observers.remove(obj);
}

public void notifyObservers() {
    for (Observer obj : observers) {
        obj.update();
    }
}
```

Class compiled - no syntax errors

BlueJ: Debugger

Options

Threads: main (stopped)

Call Sequence

- observer.ConcreteSubject.getState
- observer.Pessimist.update
- observer.ConcreteSubject.notify
- observer.ConcreteSubject.setSta
- observer.ObserverTest.runTest
- observer.ObserverTest.main

Static variables

Instance variables

- private String **subjectState** = "The New iPad is out today"
- private ArrayList<Observer> **observers** = <object reference>

Local variables

Buttons: Halt, Step, Step Into, Continue, Terminate

1.2 o.update()- Update state in the Optimist class

The screenshot shows the BlueJ IDE with the `Optimist` class open. The class extends `ConcreteObserver` and implements the `update()` method. The debugger is open on the right, showing the `main` thread stopped at the `observer.Optimist.update` call. The instance variables section shows `observerState` as `null` and `subject` as a `ConcreteSubject` object reference.

```
package observer;

public class Optimist extends ConcreteObserver {

    public Optimist( ConcreteSubject sub ) {
        super( sub );
    }

    public void update() {
        if ( subject.getState().equalsIgnoreCase("The New iPad is out today") ) {
            observerState = "Great! It's time to go green";
        } else if ( subject.getState().equalsIgnoreCase( "The New iPad is out today" ) ) {
            observerState = "Apple, take my money!";
        } else {
            observerState = ":";
        }
    }
}
```

Class compiled - no syntax errors

BlueJ: Debugger

Options

Threads: main (stopped)

Call Sequence

- observer.Optimist.update
- observer.ConcreteSubject.notify
- observer.ConcreteSubject.setSta
- observer.ObserverTest.runTest
- observer.ObserverTest.main

Static variables

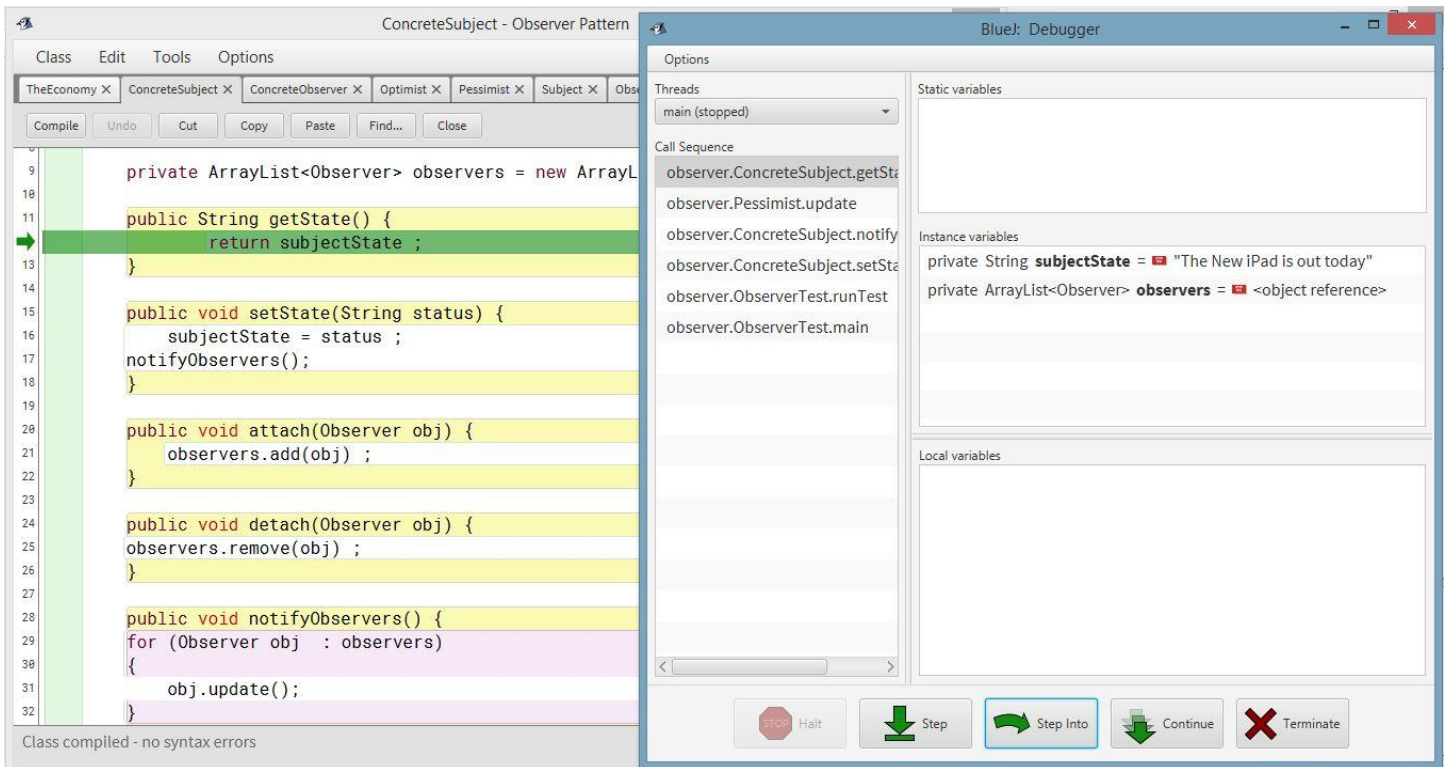
Instance variables

- protected String **observerState** = null
- protected ConcreteSubject **subject** = <object reference>

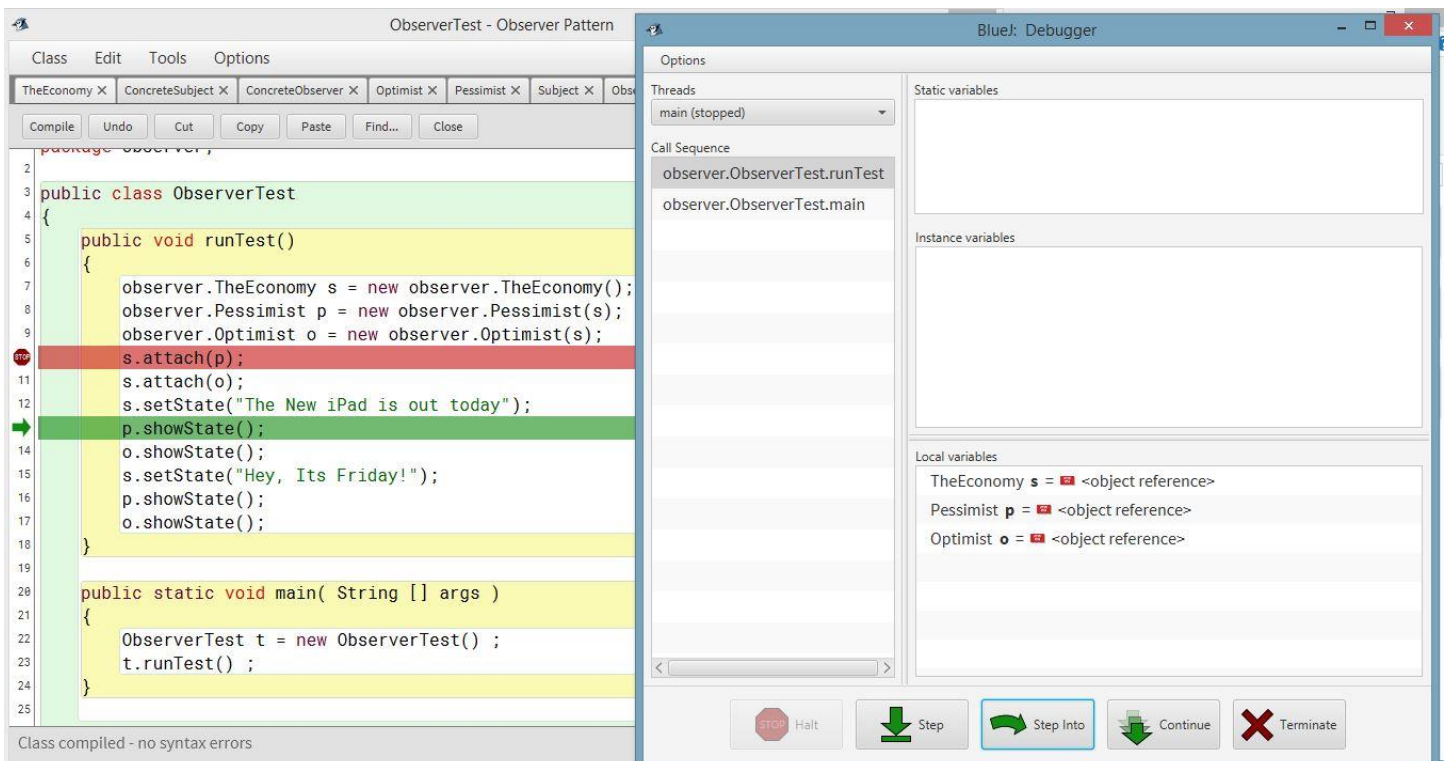
Local variables

Buttons: Halt, Step, Step Into, Continue, Terminate

1.2.1 o.getState()- Retrieve new state of Optimist class



2. p.showState()- Display the current state of Pessimist class



3. o.showState()-Display the current state of Optimist class

The screenshot shows the BlueJ IDE with the `ObserverTest` class open. The code is as follows:

```
package observer;  
  
public class ObserverTest  
{  
    public void runTest()  
    {  
        observer.TheEconomy s = new observer.TheEconomy();  
        observer.Pessimist p = new observer.Pessimist(s);  
        observer.Optimist o = new observer.Optimist(s);  
        s.attach(p);  
        s.attach(o);  
        s.setState("The New iPad is out today");  
        p.showState();  
        o.showState();  
        s.setState("Hey, Its Friday!");  
        p.showState();  
        o.showState();  
    }  
  
    public static void main( String [] args )  
    {  
        ObserverTest t = new ObserverTest() ;  
        t.runTest() ;  
    }  
}
```

The debugger window on the right shows the following state:

- Threads:** main (stopped)
- Call Sequence:** observer.ObserverTest.runTest, observer.ObserverTest.main
- Static variables:** (empty)
- Instance variables:** (empty)
- Local variables:**
 - TheEconomy s = <object reference>
 - Pessimist p = <object reference>
 - Optimist o = <object reference>

The status bar at the bottom indicates "Class compiled - no syntax errors".

4. s.setState()- Set new state in TheEconomy class

The screenshot shows the BlueJ IDE with the `ObserverTest` class open. The code is as follows:

```
package observer;  
  
public class ObserverTest  
{  
    public void runTest()  
    {  
        observer.TheEconomy s = new observer.TheEconomy();  
        observer.Pessimist p = new observer.Pessimist(s);  
        observer.Optimist o = new observer.Optimist(s);  
        s.attach(p);  
        s.attach(o);  
        s.setState("The New iPad is out today");  
        p.showState();  
        o.showState();  
        s.setState("Hey, Its Friday!");  
        p.showState();  
        o.showState();  
    }  
  
    public static void main( String [] args )  
    {  
        ObserverTest t = new ObserverTest() ;  
        t.runTest() ;  
    }  
}
```

The debugger window on the right shows the following state:

- Threads:** main (stopped)
- Call Sequence:** observer.ObserverTest.runTest, observer.ObserverTest.main
- Static variables:** (empty)
- Instance variables:** (empty)
- Local variables:**
 - TheEconomy s = <object reference>
 - Pessimist p = <object reference>
 - Optimist o = <object reference>

The status bar at the bottom indicates "Class compiled - no syntax errors".

4.1 p.update()- Update state change in Pessimist class

The screenshot shows an IDE with the 'Pessimist - Observer Pattern' project. The 'Pessimist' class is open, showing the 'update()' method. The debugger window is also open, showing the 'Call Sequence' and 'Instance variables'.

```
package observer;

public class Pessimist extends ConcreteObserver {

    public Pessimist( ConcreteSubject sub )
    {
        super( sub );
    }

    public void update() {
        if ( subject.getState().equalsIgnoreCase("The Price")
        {
            observerState = "This is the beginning of the
        }
        else if ( subject.getState().equalsIgnoreCase( "The
        {
            observerState = "Not another iPad!" ;
        }
        else
        {
            observerState = ":( " ;
        }
    }
}
```

Class compiled - no syntax errors

Debugger window (BlueJ: Debugger) shows:

- Threads: main (stopped)
- Call Sequence: observer.Pessimist.update, observer.ConcreteSubject.notify, observer.ConcreteSubject.setSta, observer.ObserverTest.runTest, observer.ObserverTest.main
- Instance variables: protected String observerState = null, protected ConcreteSubject subject = <object reference>
- Local variables: (empty)

Debugger controls: Halt, Step, Step Into, Continue, Terminate.

4.1.1 p.getState()- Retrieving Pessimists class state

The screenshot shows an IDE with the 'ConcreteSubject - Observer Pattern' project. The 'ConcreteSubject' class is open, showing the 'getState()' method. The debugger window is also open, showing the 'Call Sequence' and 'Instance variables'.

```
private ArrayList<Observer> observers = new ArrayList<>();

public String getState() {
    return subjectState ;
}

public void setState(String status) {
    subjectState = status ;
    notifyObservers();
}

public void attach(Observer obj) {
    observers.add(obj) ;
}

public void detach(Observer obj) {
    observers.remove(obj) ;
}

public void notifyObservers() {
    for (Observer obj : observers)
    {
        obj.update();
    }
}
```

Class compiled - no syntax errors

Debugger window (BlueJ: Debugger) shows:

- Threads: main (stopped)
- Call Sequence: observer.ConcreteSubject.getSta, observer.Pessimist.update, observer.ConcreteSubject.notify, observer.ConcreteSubject.setSta, observer.ObserverTest.runTest, observer.ObserverTest.main
- Instance variables: private String subjectState = "The New iPad is out today", private ArrayList<Observer> observers = <object reference>
- Local variables: (empty)

Debugger controls: Halt, Step, Step Into, Continue, Terminate.

4.2 o.update()- Updating state change in Optimist class

The screenshot shows an IDE with the 'Pessimist - Observer Pattern' project. The 'Pessimist' class is open, showing the `update()` method. The debugger window is also open, showing the 'Call Sequence' and 'Instance variables'.

```
package observer;

public class Pessimist extends ConcreteObserver {

    public Pessimist( ConcreteSubject sub )
    {
        super( sub );
    }

    public void update() {
        if ( subject.getState().equalsIgnoreCase("The Price")
        {
            observerState = "This is the beginning of the
        }
        else if ( subject.getState().equalsIgnoreCase( "The
        {
            observerState = "Not another iPad!" ;
        }
        else
        {
            observerState = ":( " ;
        }
    }
}
```

Class compiled - no syntax errors

Debugger window (BlueJ: Debugger) shows the following:

- Threads: main (stopped)
- Call Sequence: observer.Pessimist.update, observer.ConcreteSubject.notify, observer.ConcreteSubject.setSta, observer.ObserverTest.runTest, observer.ObserverTest.main
- Instance variables: protected String **observerState** = null, protected ConcreteSubject **subject** = <object reference>
- Local variables: (empty)

Debugger controls: Halt, Step, Step Into, Continue, Terminate.

4.2.1 o.getState()- Retrieving Optimist class state

The screenshot shows an IDE with the 'ConcreteSubject - Observer Pattern' project. The 'ConcreteSubject' class is open, showing the `getState()` method. The debugger window is also open, showing the 'Call Sequence' and 'Instance variables'.

```
private ArrayList<Observer> observers = new ArrayList<>();

public String getState() {
    return subjectState ;
}

public void setState(String status) {
    subjectState = status ;
    notifyObservers();
}

public void attach(Observer obj) {
    observers.add(obj) ;
}

public void detach(Observer obj) {
    observers.remove(obj) ;
}

public void notifyObservers() {
    for (Observer obj : observers)
    {
        obj.update();
    }
}
```

Class compiled - no syntax errors

Debugger window (BlueJ: Debugger) shows the following:

- Threads: main (stopped)
- Call Sequence: observer.ConcreteSubject.getSta, observer.Pessimist.update, observer.ConcreteSubject.notify, observer.ConcreteSubject.setSta, observer.ObserverTest.runTest, observer.ObserverTest.main
- Instance variables: private String **subjectState** = "The New iPad is out today", private ArrayList<Observer> **observers** = <object reference>
- Local variables: (empty)

Debugger controls: Halt, Step, Step Into, Continue, Terminate.

5. p.showState()- Show current state of Pessimist class

The screenshot shows the BlueJ IDE with the `ObserverTest` class open. A breakpoint is set at line 15, `p.showState();`. The debugger window on the right shows the call sequence: `observer.ObserverTest.runTest` and `observer.ObserverTest.main`. The local variables section shows: `TheEconomy s = <object reference>`, `Pessimist p = <object reference>`, and `Optimist o = <object reference>`. The `runTest` method is highlighted in yellow, and the `main` method is also highlighted in yellow.

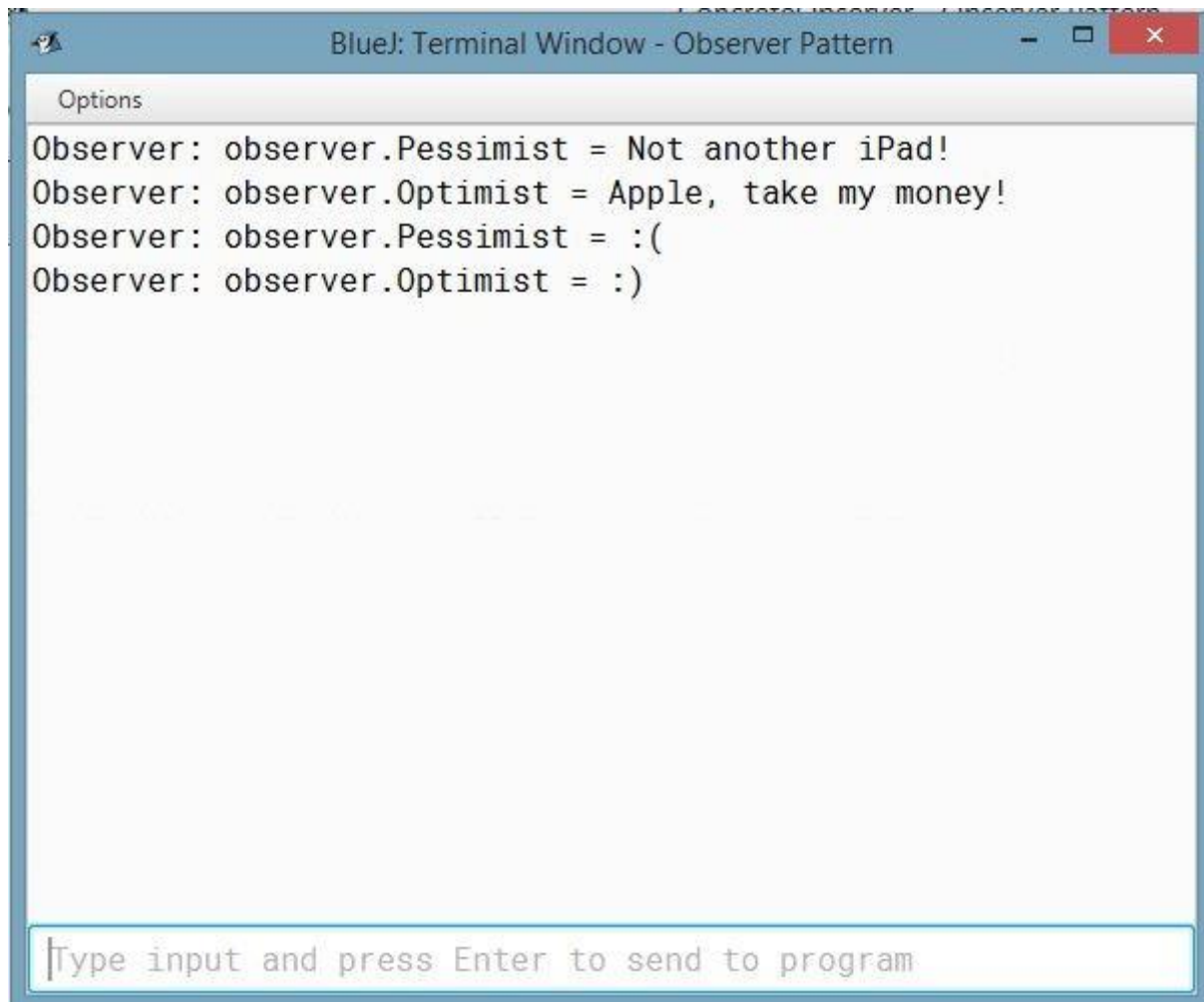
```
1 package observer;
2
3 public class ObserverTest
4 {
5     public void runTest()
6     {
7         observer.TheEconomy s = new observer.TheEconomy();
8         observer.Pessimist p = new observer.Pessimist(s);
9         observer.Optimist o = new observer.Optimist(s);
10        s.attach(p);
11        s.attach(o);
12        s.setState("The New iPad is out today");
13        p.showState();
14        o.showState();
15        s.setState("Hey, Its Friday!");
16        p.showState();
17        o.showState();
18    }
19
20    public static void main( String [] args )
21    {
22        ObserverTest t = new ObserverTest() ;
23        t.runTest() ;
24    }
25 }
```

6. o.showState()- Show current state of Optimist class

The screenshot shows the BlueJ IDE with the `ObserverTest` class open. A breakpoint is set at line 16, `o.showState();`. The debugger window on the right shows the call sequence: `observer.ObserverTest.runTest` and `observer.ObserverTest.main`. The local variables section shows: `TheEconomy s = <object reference>`, `Pessimist p = <object reference>`, and `Optimist o = <object reference>`. The `runTest` method is highlighted in yellow, and the `main` method is also highlighted in yellow.

```
1 package observer;
2
3 public class ObserverTest
4 {
5     public void runTest()
6     {
7         observer.TheEconomy s = new observer.TheEconomy();
8         observer.Pessimist p = new observer.Pessimist(s);
9         observer.Optimist o = new observer.Optimist(s);
10        s.attach(p);
11        s.attach(o);
12        s.setState("The New iPad is out today");
13        p.showState();
14        o.showState();
15        s.setState("Hey, Its Friday!");
16        p.showState();
17        o.showState();
18    }
19
20    public static void main( String [] args )
21    {
22        ObserverTest t = new ObserverTest() ;
23        t.runTest() ;
24    }
25 }
```

OUTPUT:



```
Observer: observer.Pessimist = Not another iPad!
Observer: observer.Optimist = Apple, take my money!
Observer: observer.Pessimist = :(
Observer: observer.Optimist = :)

Type input and press Enter to send to program
```