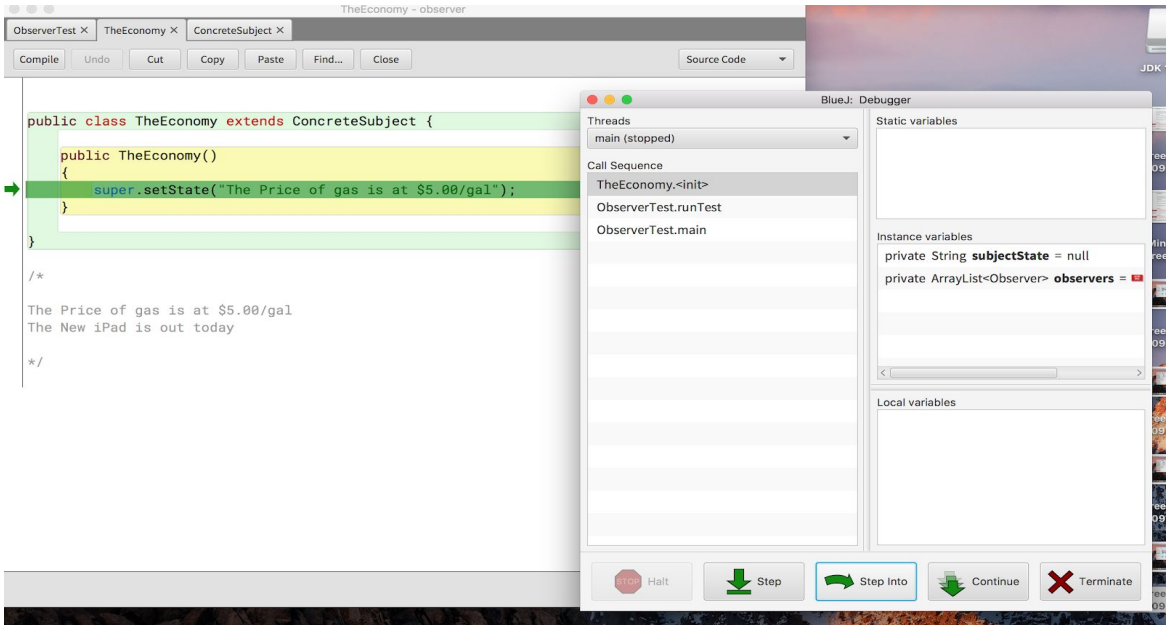
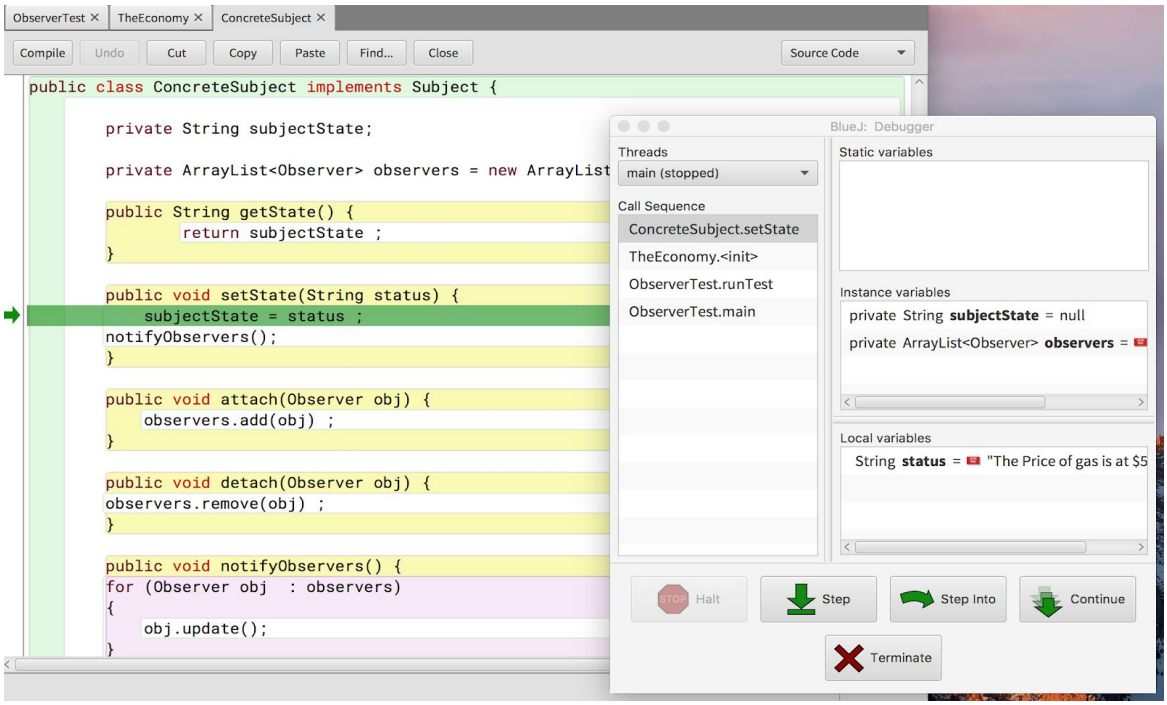
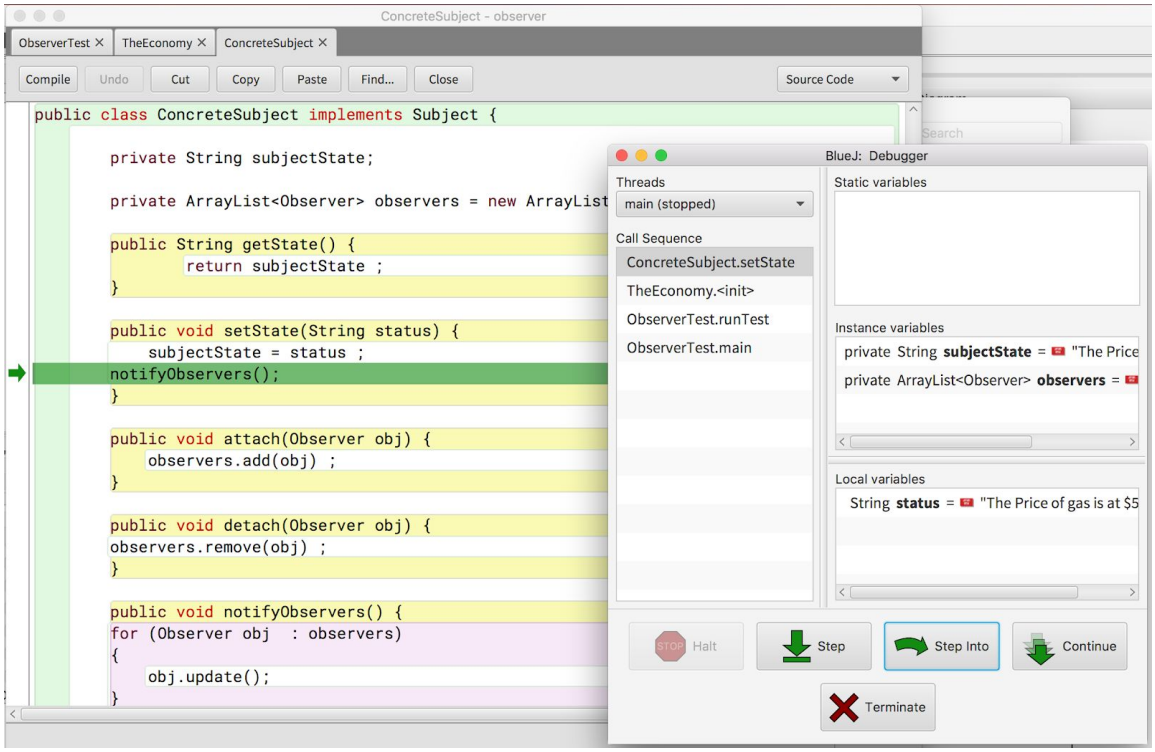
Lab 3: Observer Pattern runtime screenshots

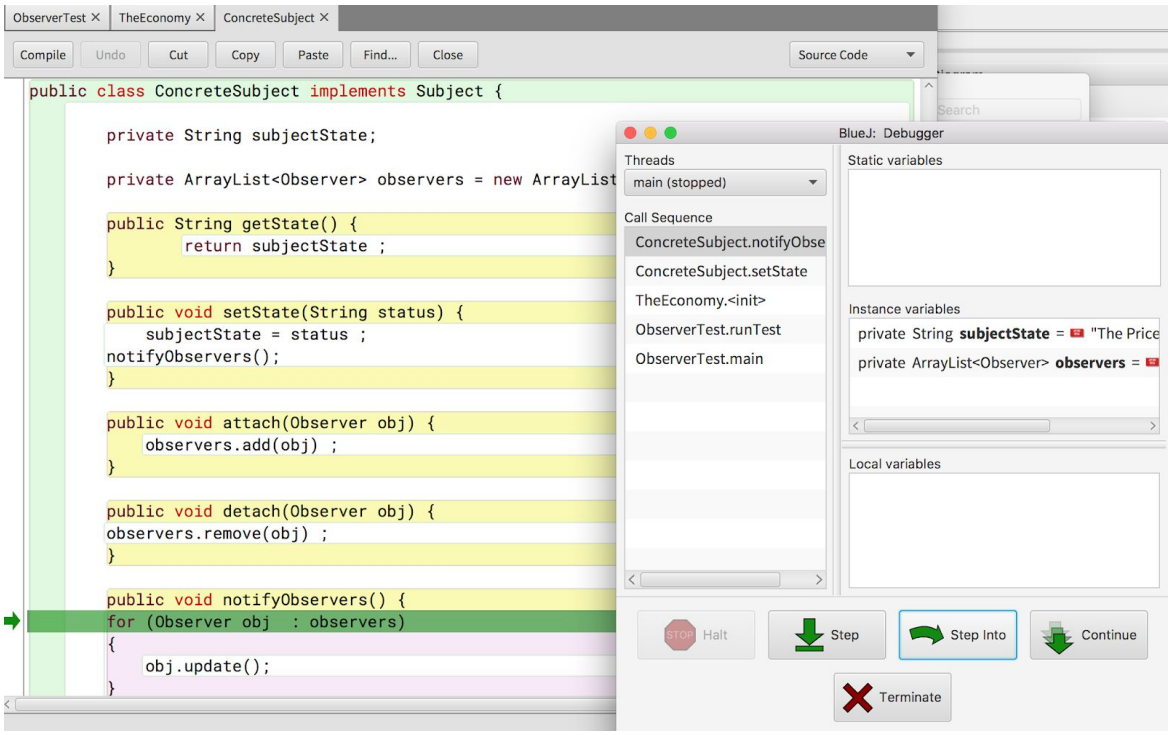
Step1: setState(status: String) : void



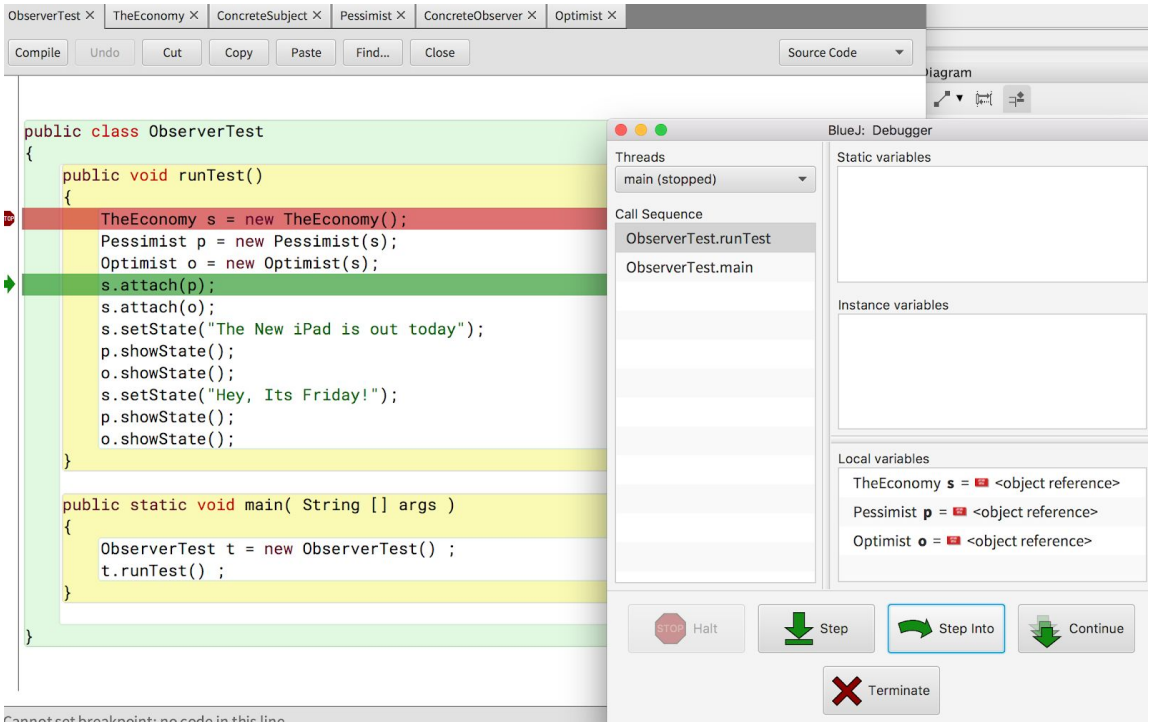


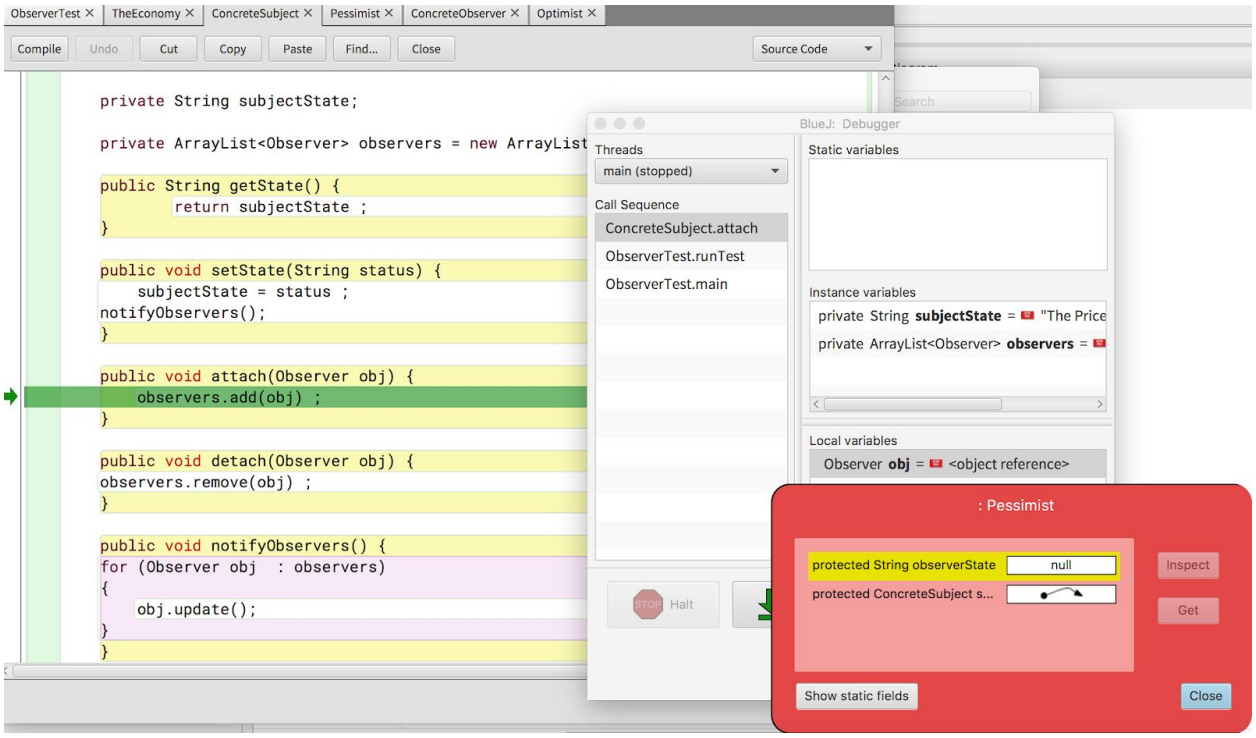
Step 1.1: notifyObservers() : void



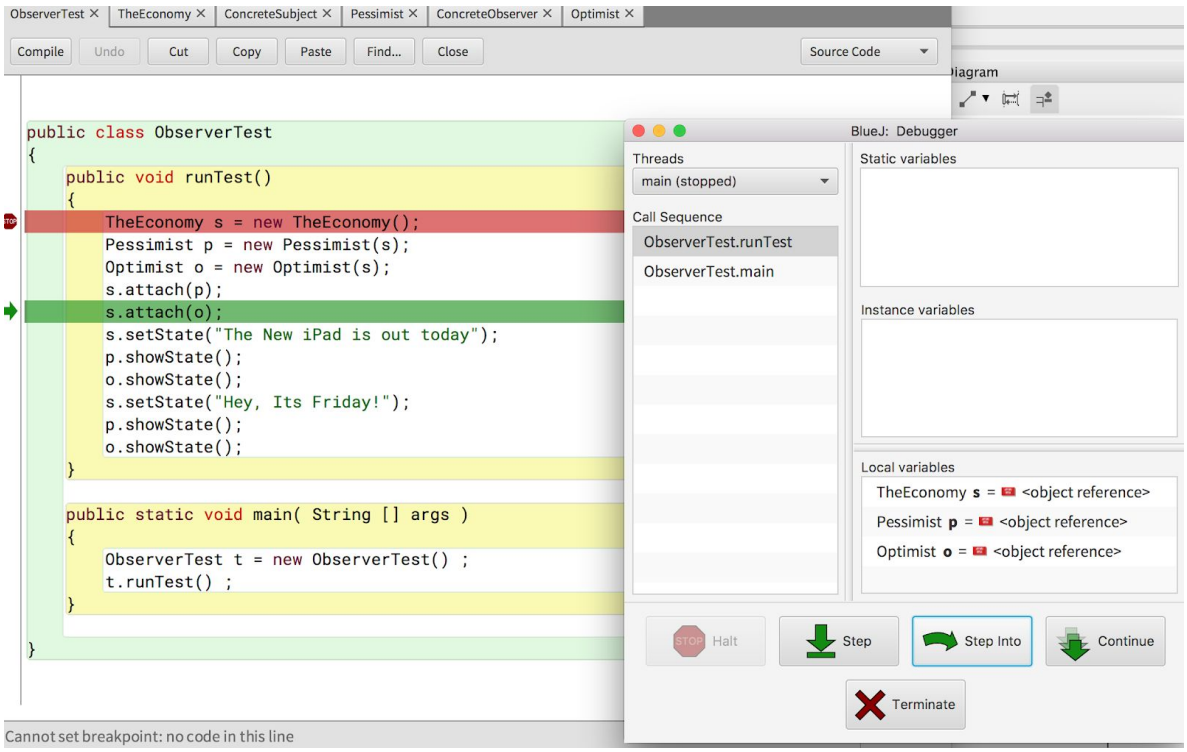


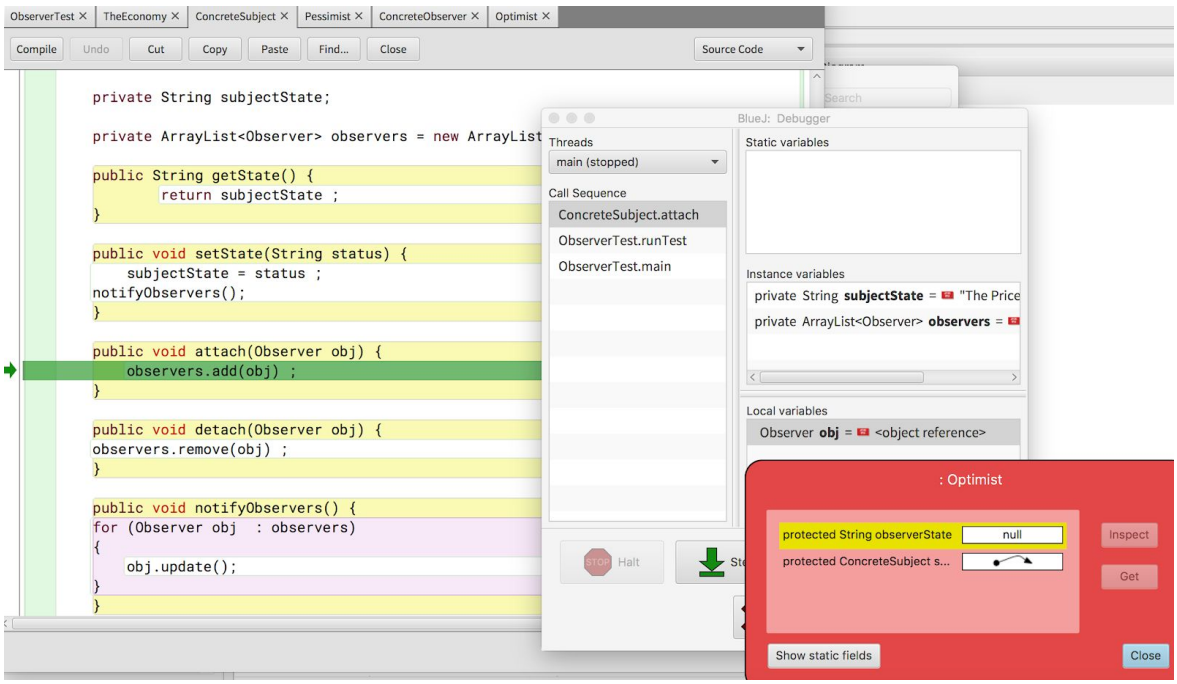
Step 2: attach(obj: Observer) : void



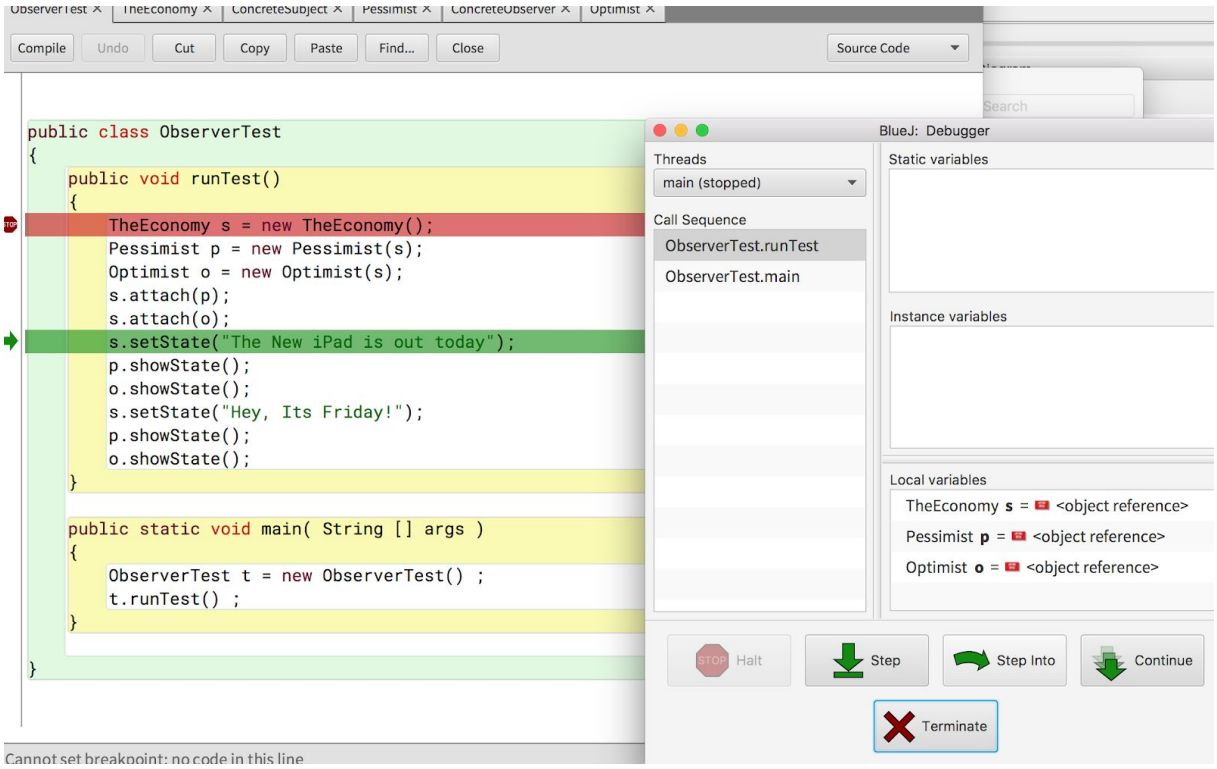


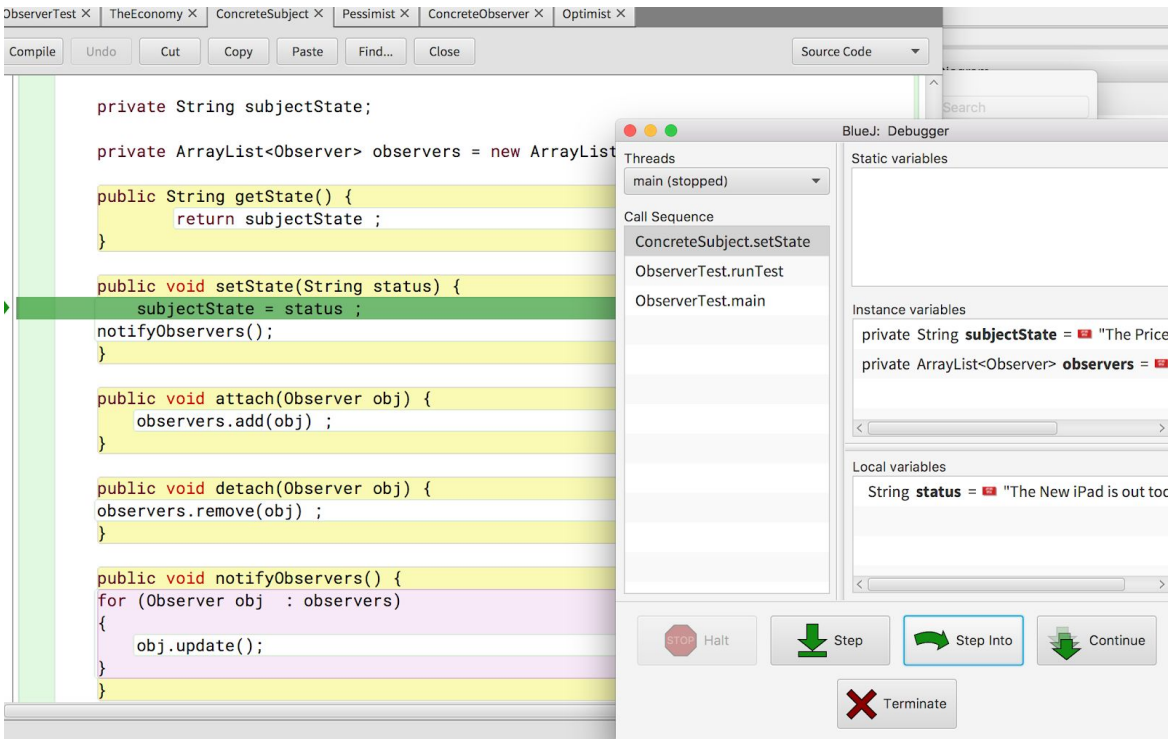
Step 3: attach(obj: Observer) : void



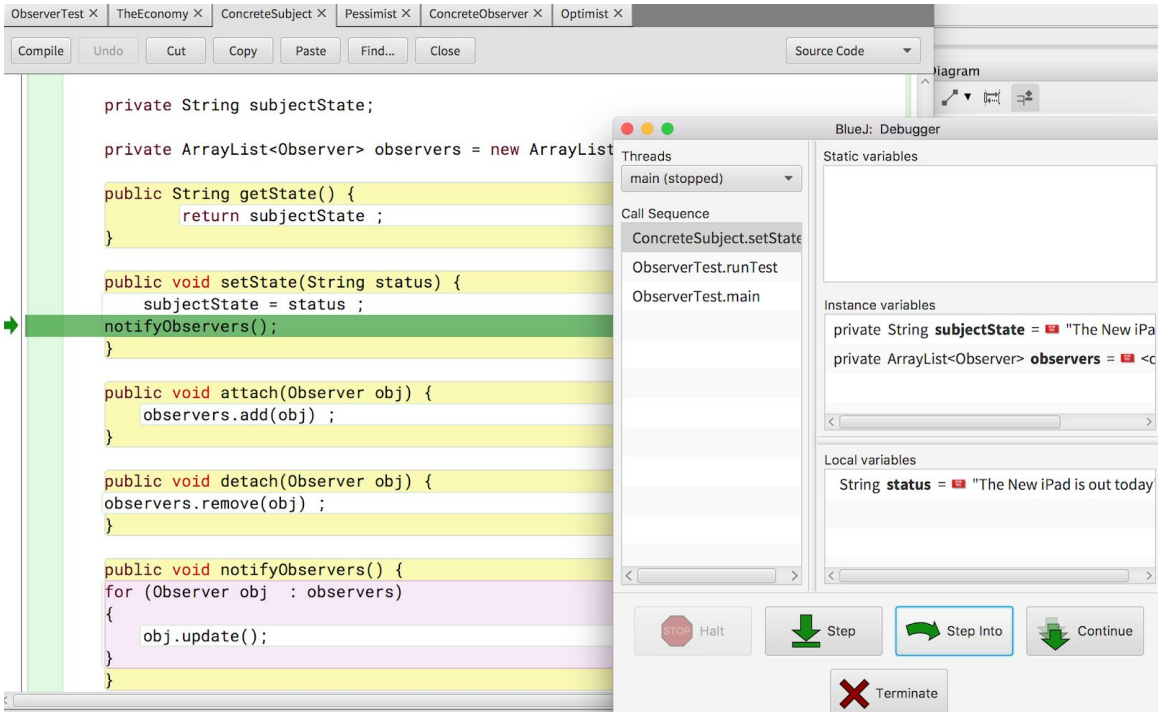


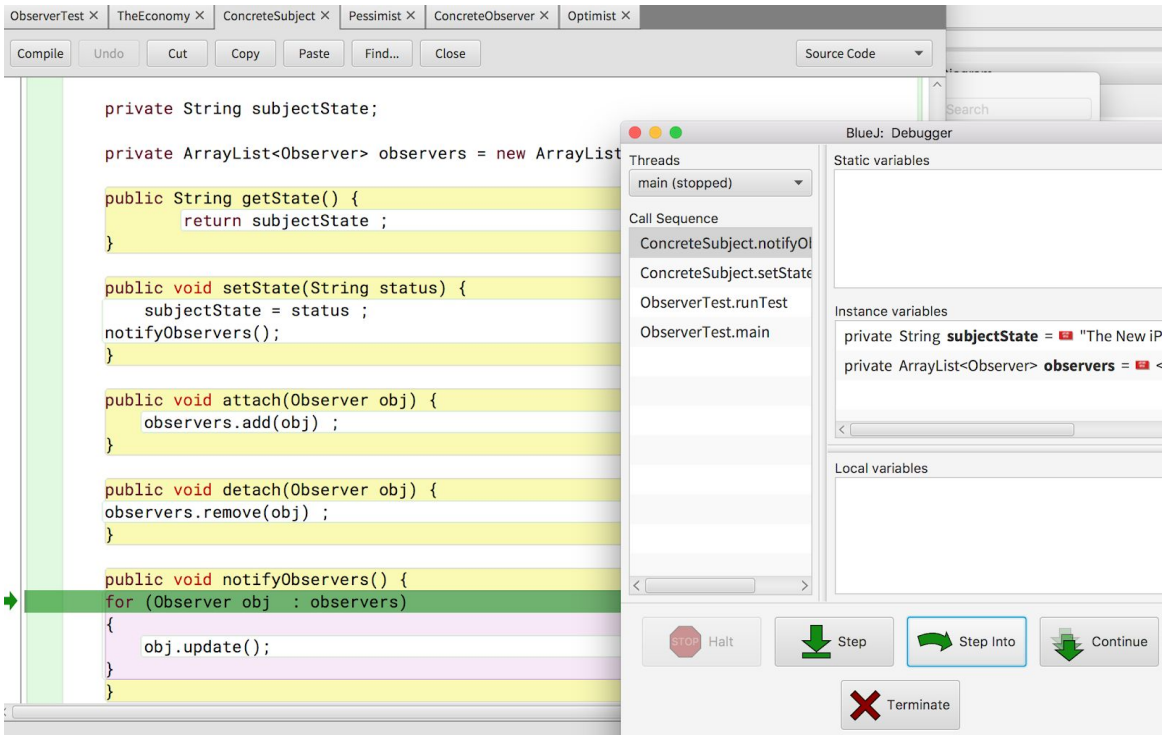
Step 4: setState(status: String) : void



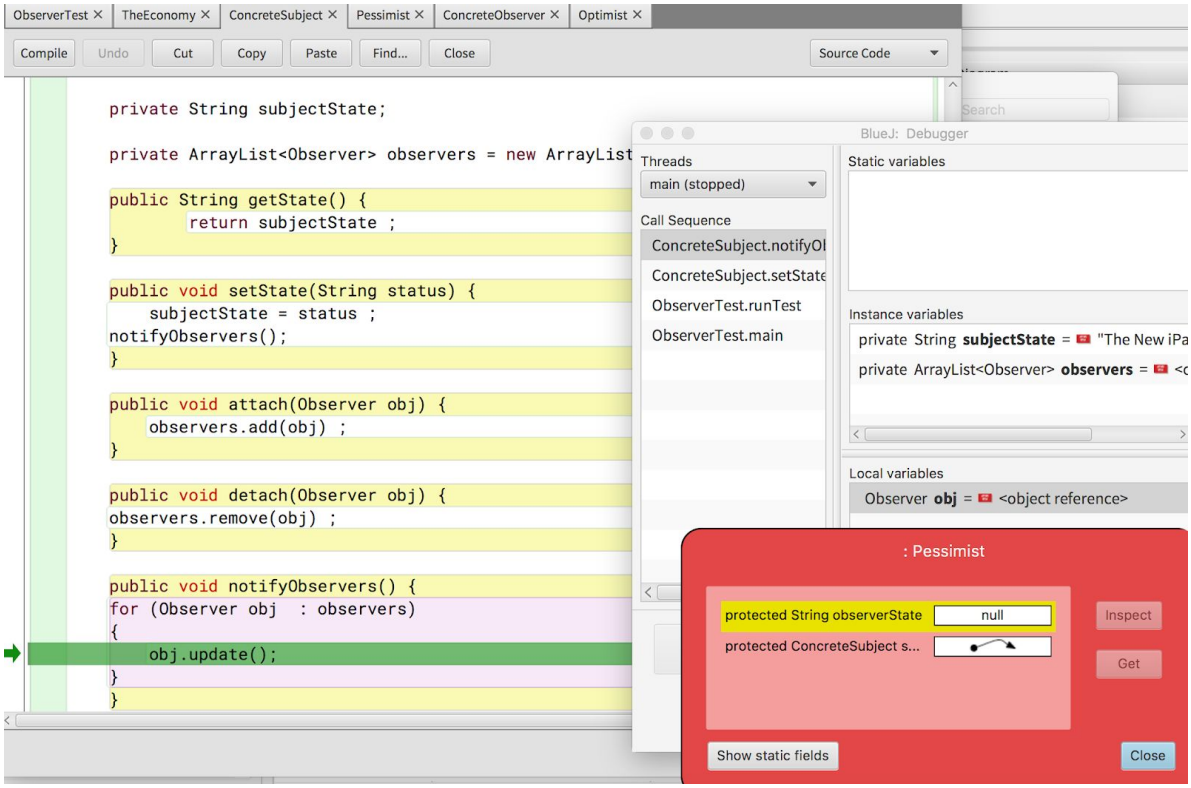


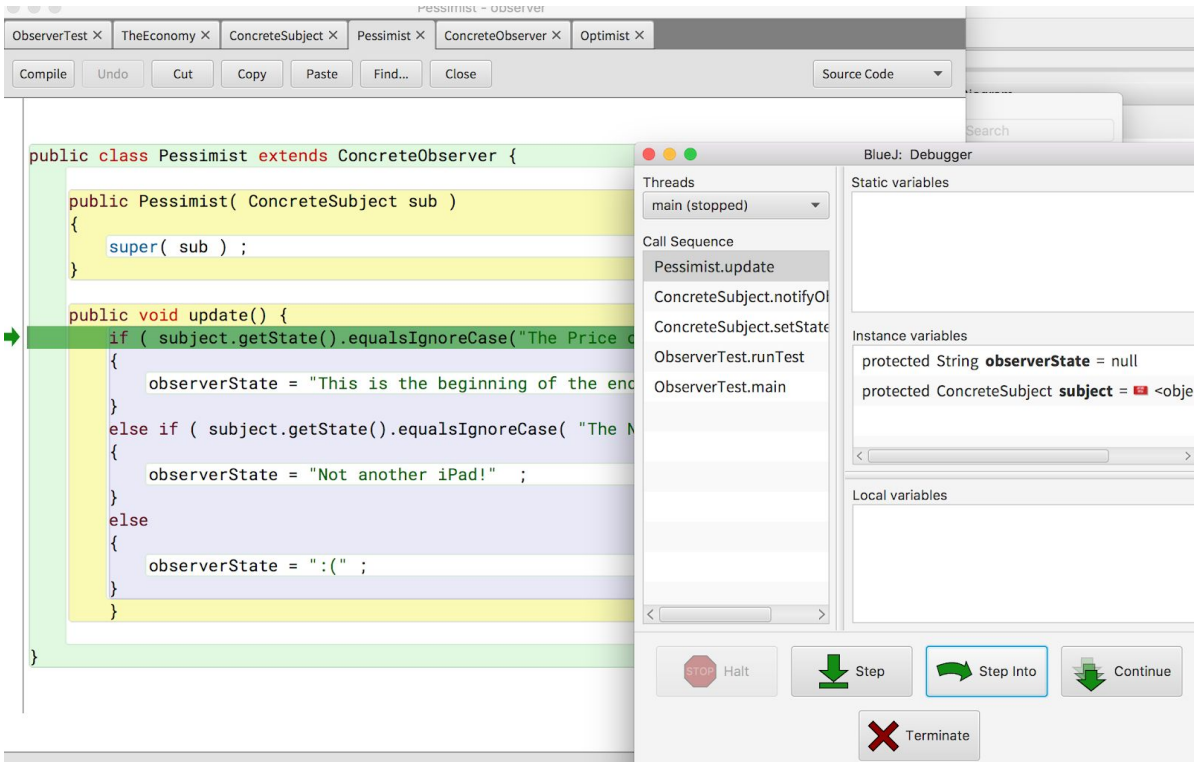
Step 4.1: notifyObservers() : void



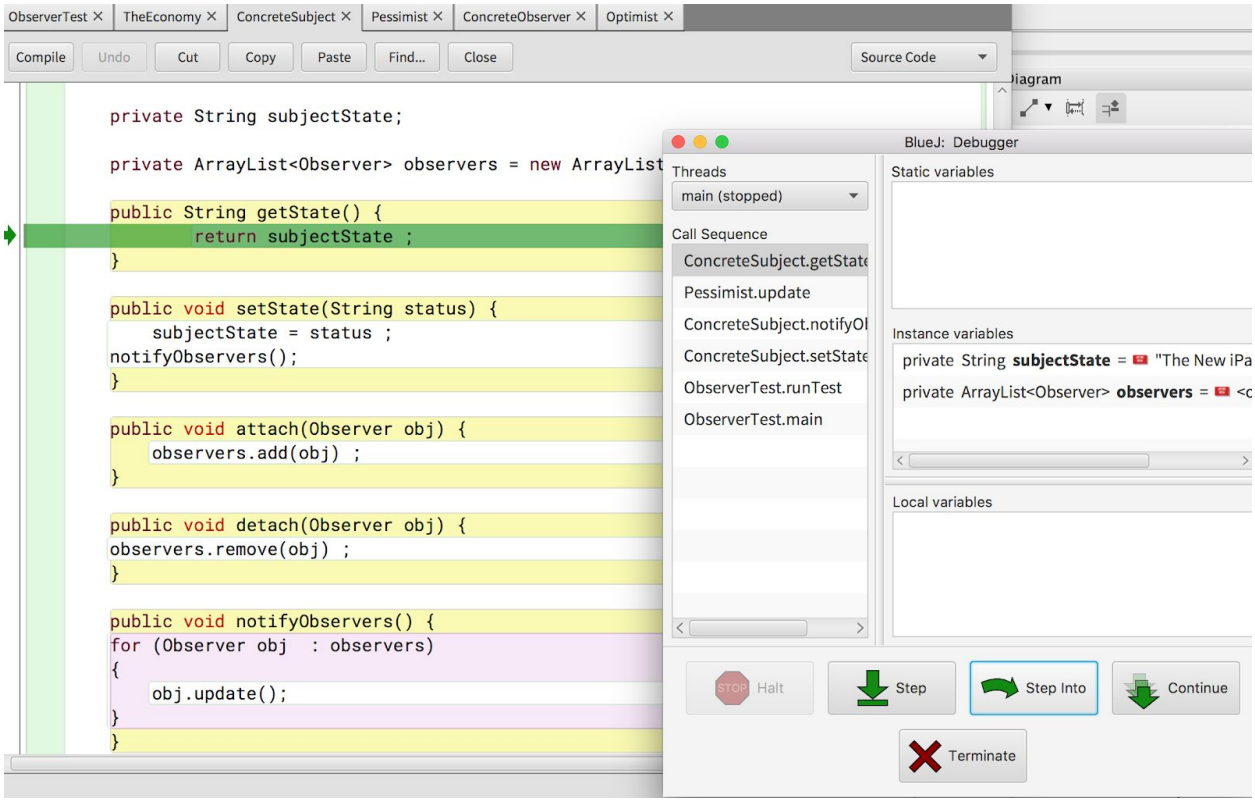


Step 4.1.1: update() : void

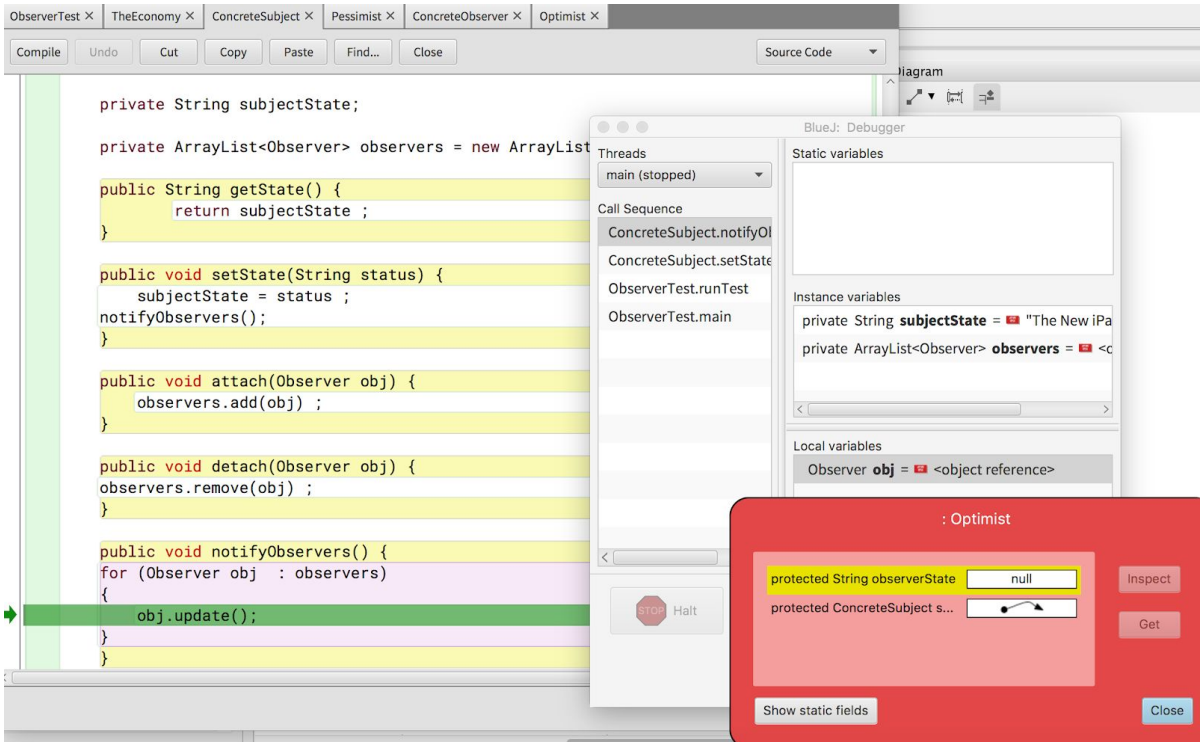


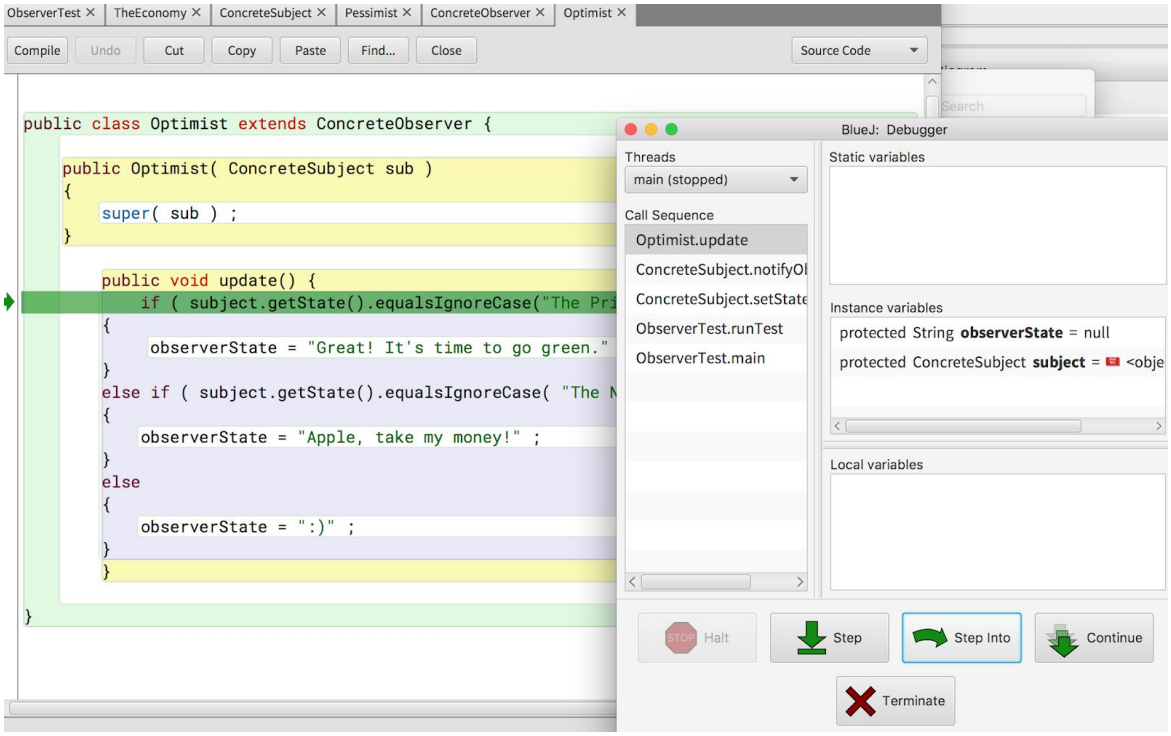


Step 4.1.1.1: getState() : String

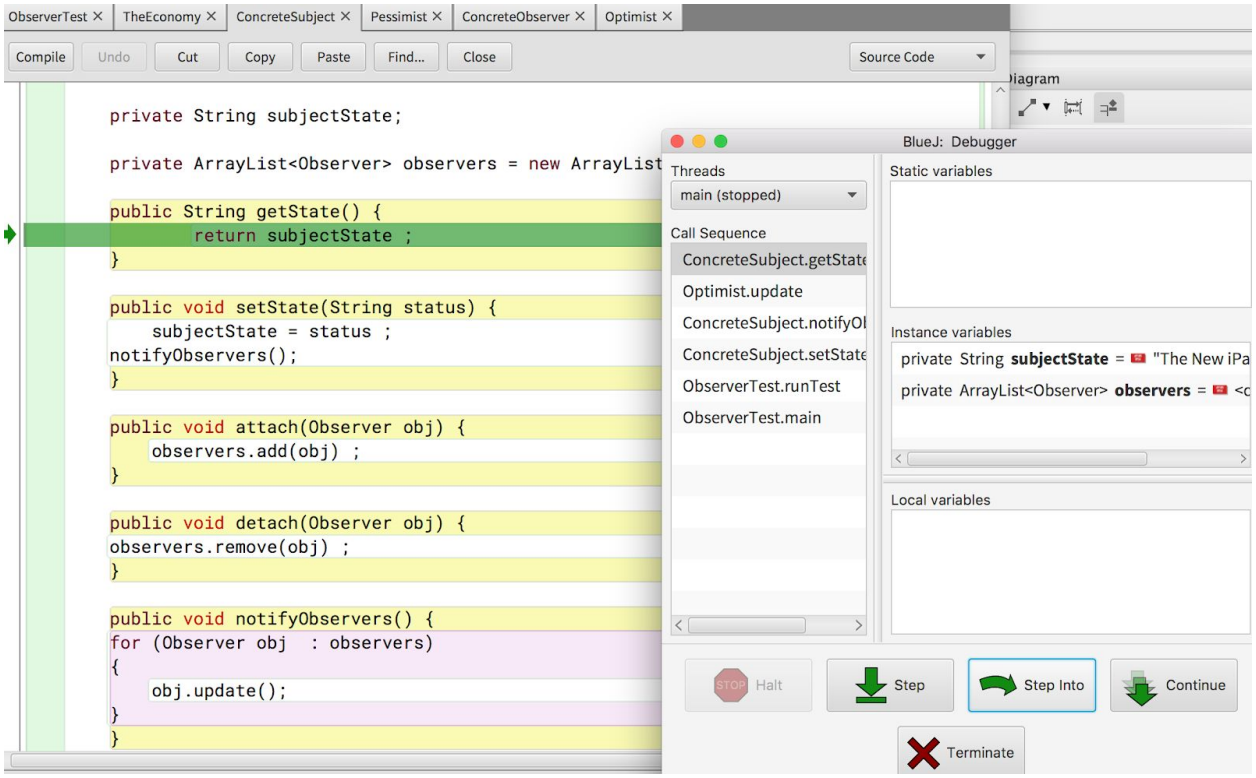


Step 4.1.2: update() : void

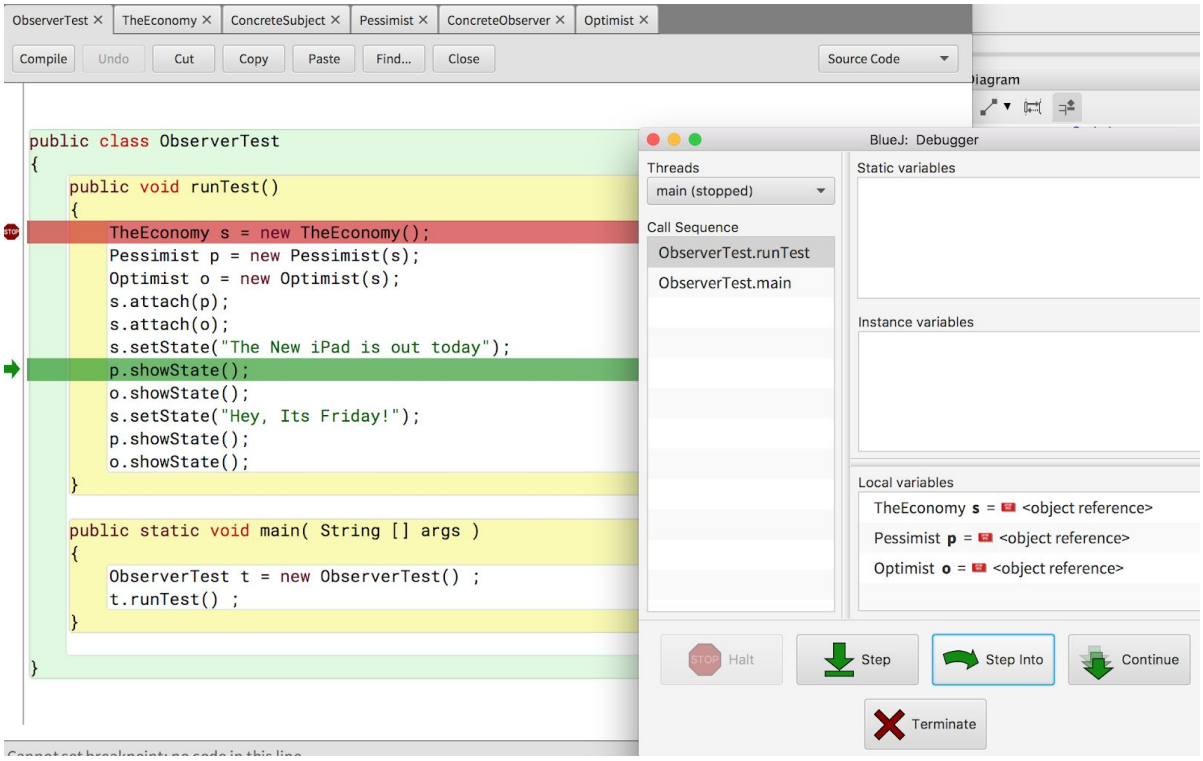


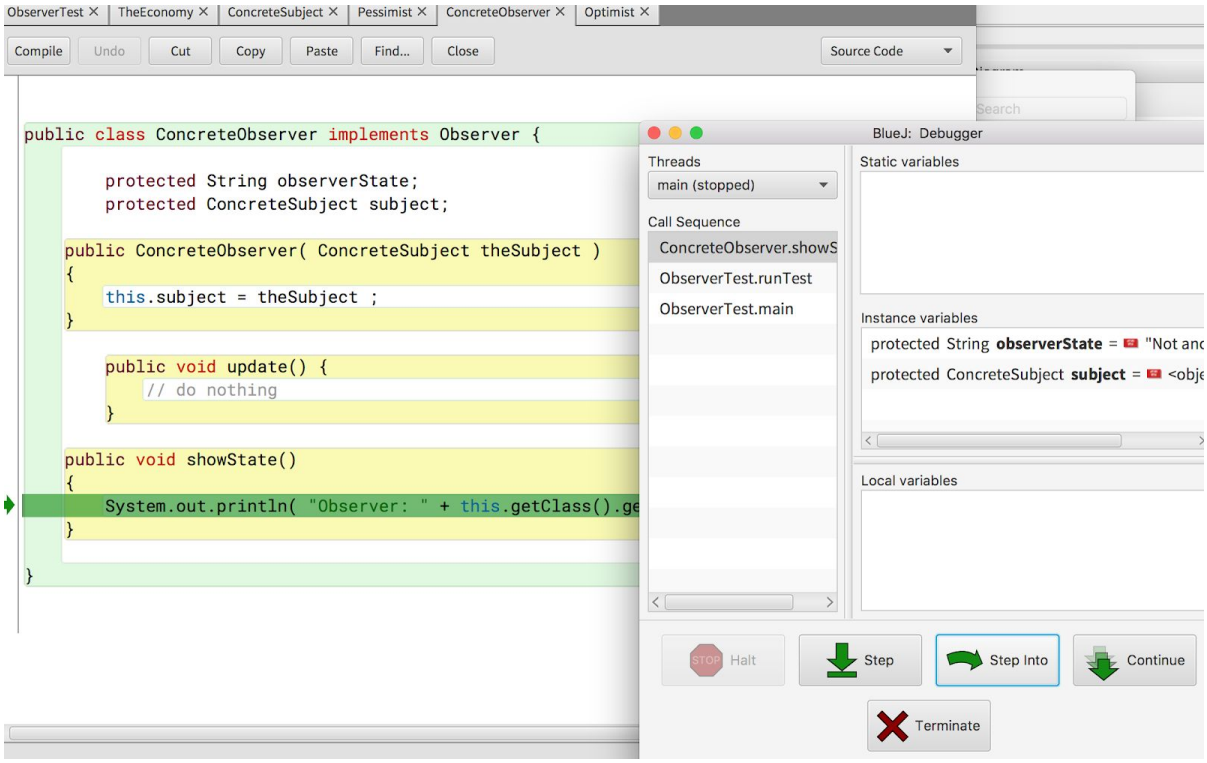


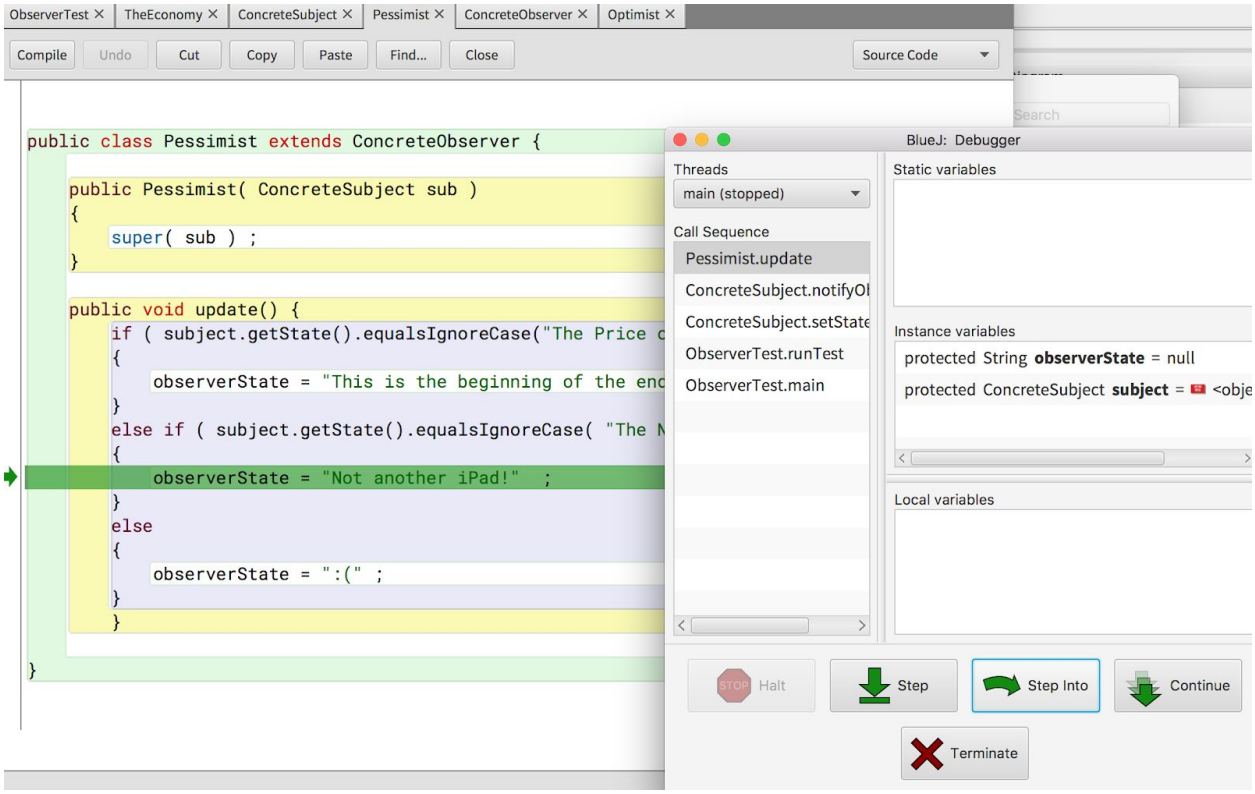
Step 4.1.2.1: getState() : String



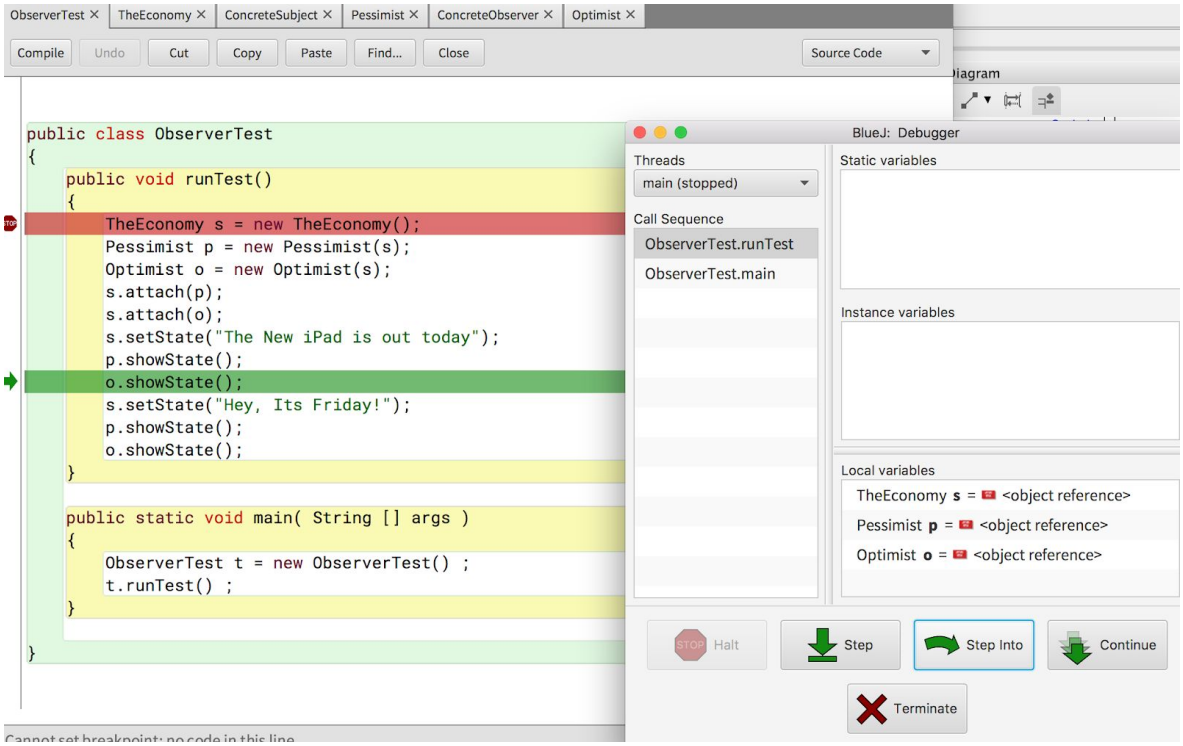
Step 5: showState() : void

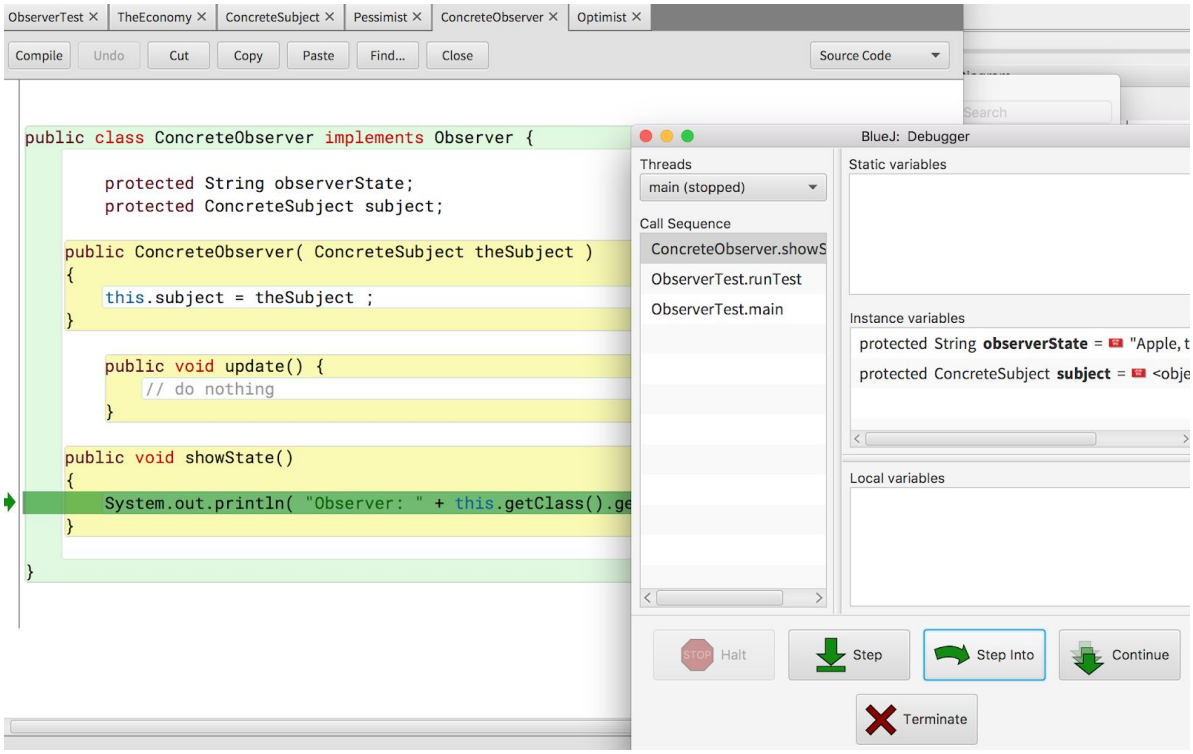


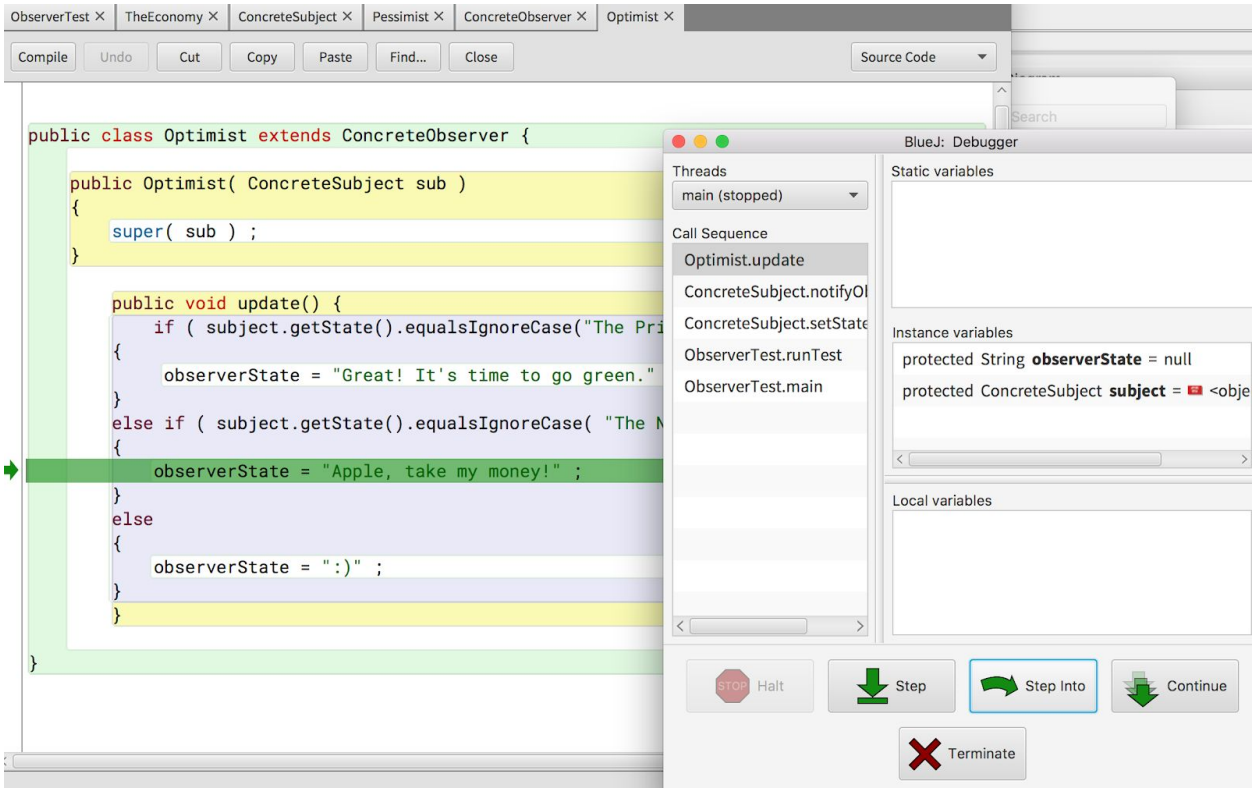



ObserverTest
TheEconomy
ConcreteSubject
Pessimist
ConcreteObserver
Optimist
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code
public class Pessimist extends ConcreteObserver {
public Pessimist(ConcreteSubject sub)
{
super(sub) ;
}
public void update() {
if (subject.getState().equalsIgnoreCase("The Price
{
observerState = "This is the beginning of the enc
}
else if (subject.getState().equalsIgnoreCase("The
{
observerState = "Not another iPad!" ;
}
else
{
observerState = ":(" ;
}
}
}
BlueJ: Debugger
Threads
main (stopped)
Call Sequence
Pessimist.update
ConcreteSubject.notifyO
ConcreteSubject.setState
ObserverTest.runTest
ObserverTest.main
Static variables
Instance variables
protected String observerState = null
protected ConcreteSubject subject = <obje
Local variables
Halt
Step
Step Into
Continue
Terminate

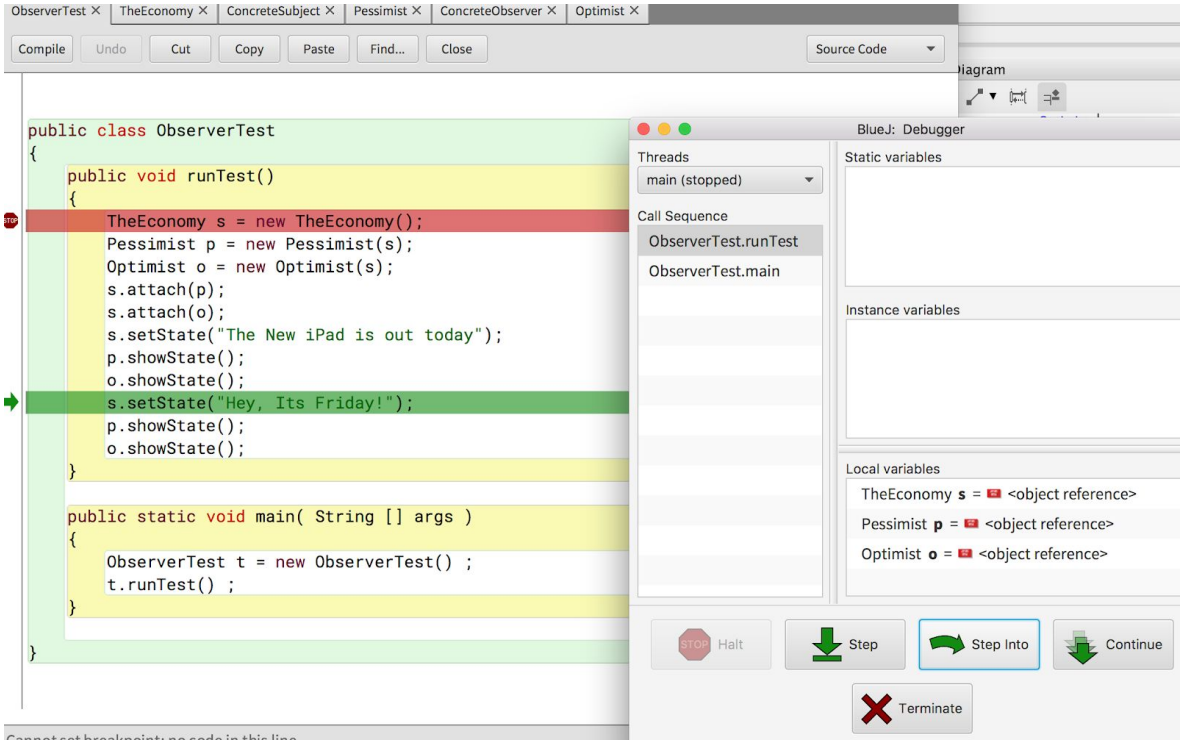
Step 6: showState() : void

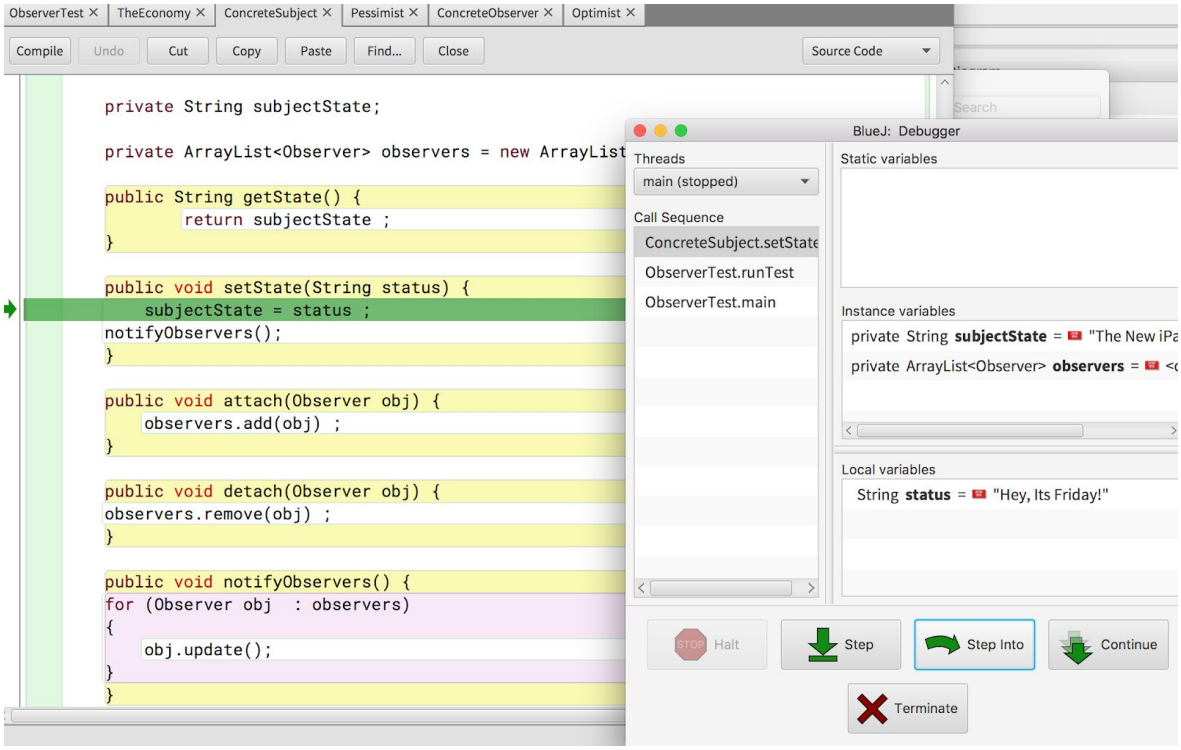




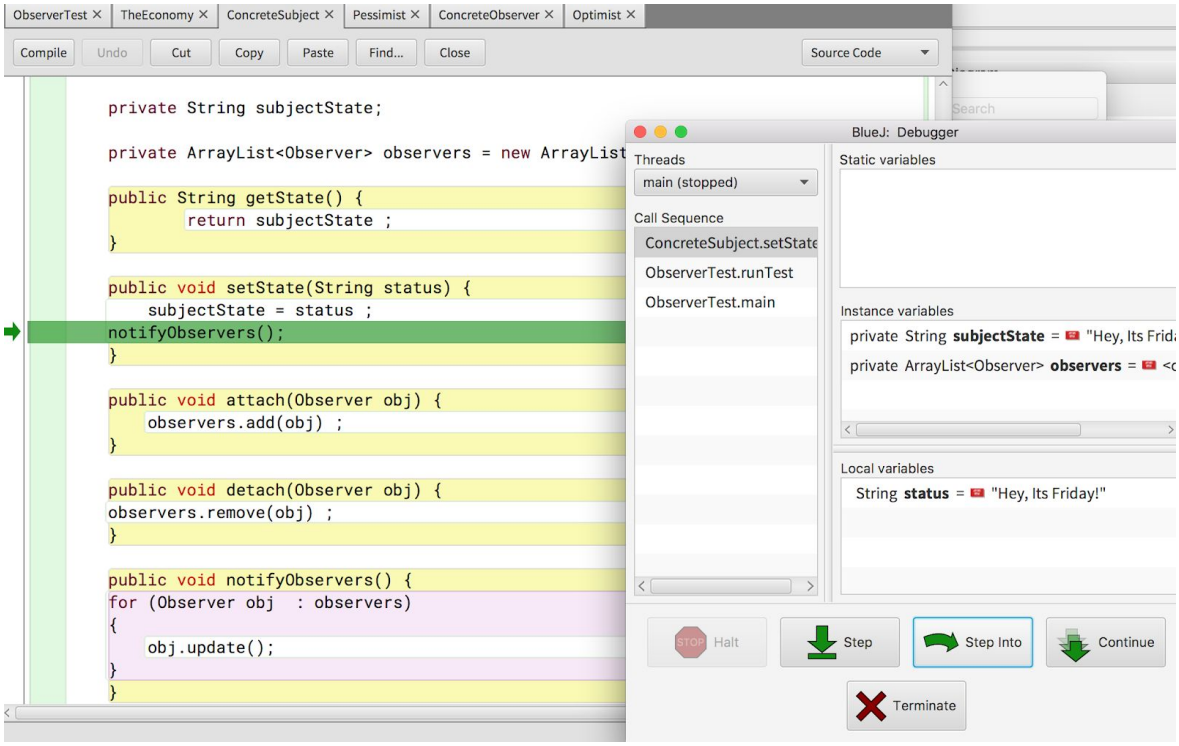

ObserverTest
TheEconomy
ConcreteSubject
Pessimist
ConcreteObserver
Optimist
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code
public class Optimist extends ConcreteObserver {
public Optimist(ConcreteSubject sub)
{
super(sub) ;
}
public void update() {
if (subject.getState().equalsIgnoreCase("The Pri
{
observerState = "Great! It's time to go green."
}
else if (subject.getState().equalsIgnoreCase("The N
{
observerState = "Apple, take my money!" ;
}
else
{
observerState = ":)" ;
}
}
}
BlueJ: Debugger
Threads
main (stopped)
Call Sequence
Optimist.update
ConcreteSubject.notifyOl
ConcreteSubject.setState
ObserverTest.runTest
ObserverTest.main
Static variables
Instance variables
protected String observerState = null
protected ConcreteSubject subject = <obje
Local variables
Halt
Step
Step Into
Continue
Terminate

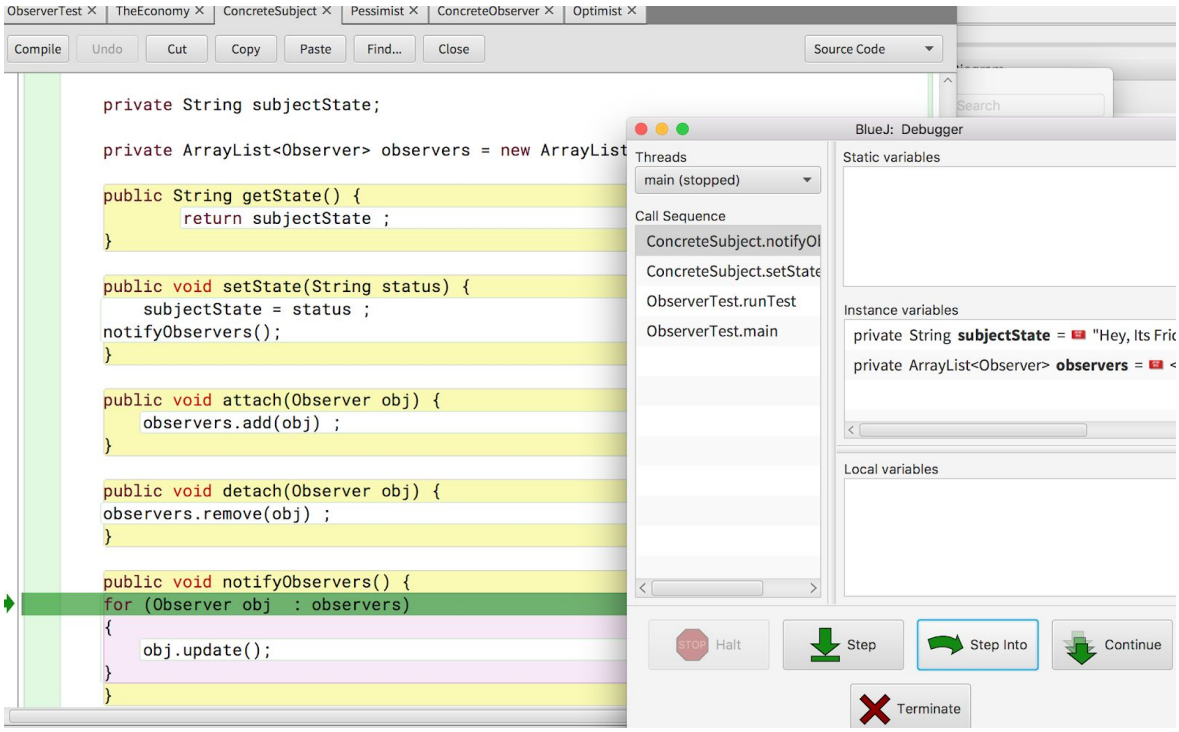
Step 7: setState(status: String) : void



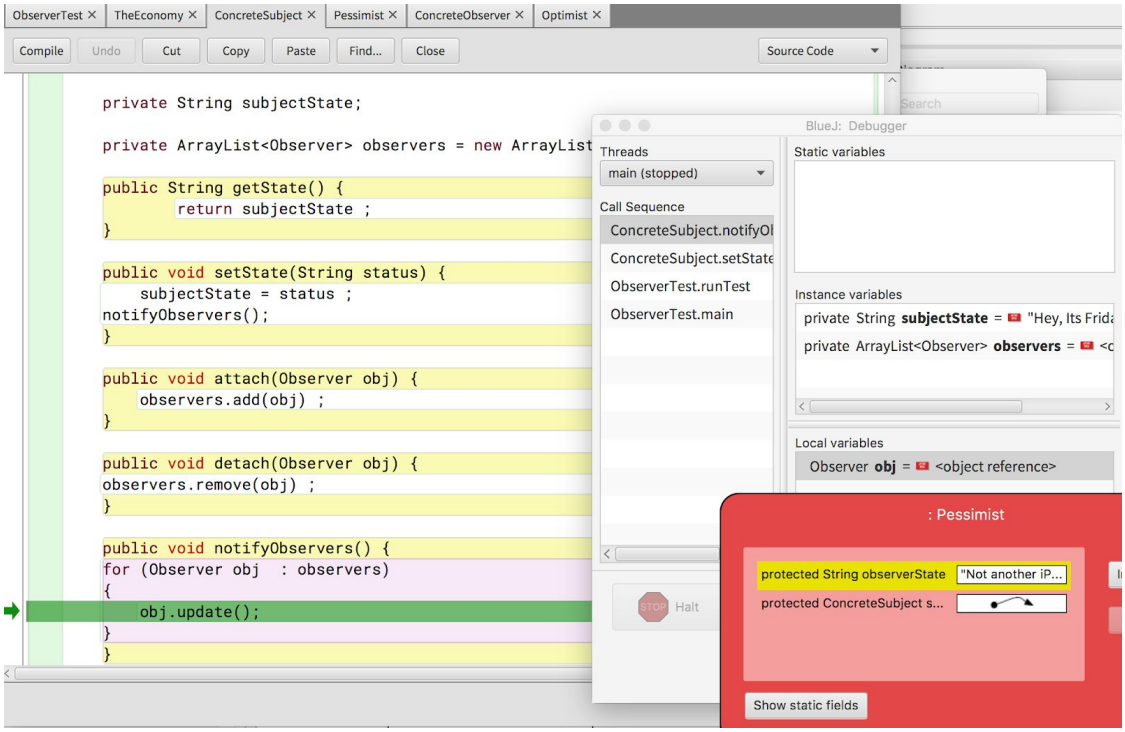


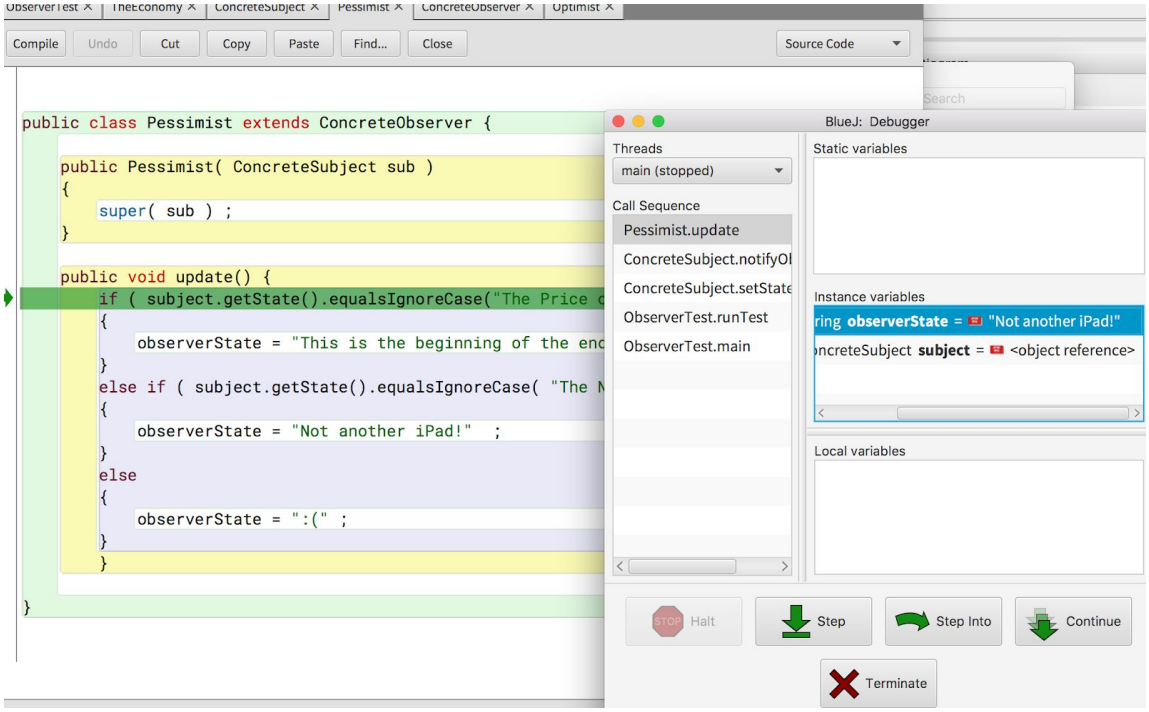
Step 7.1: notifyObservers() : void



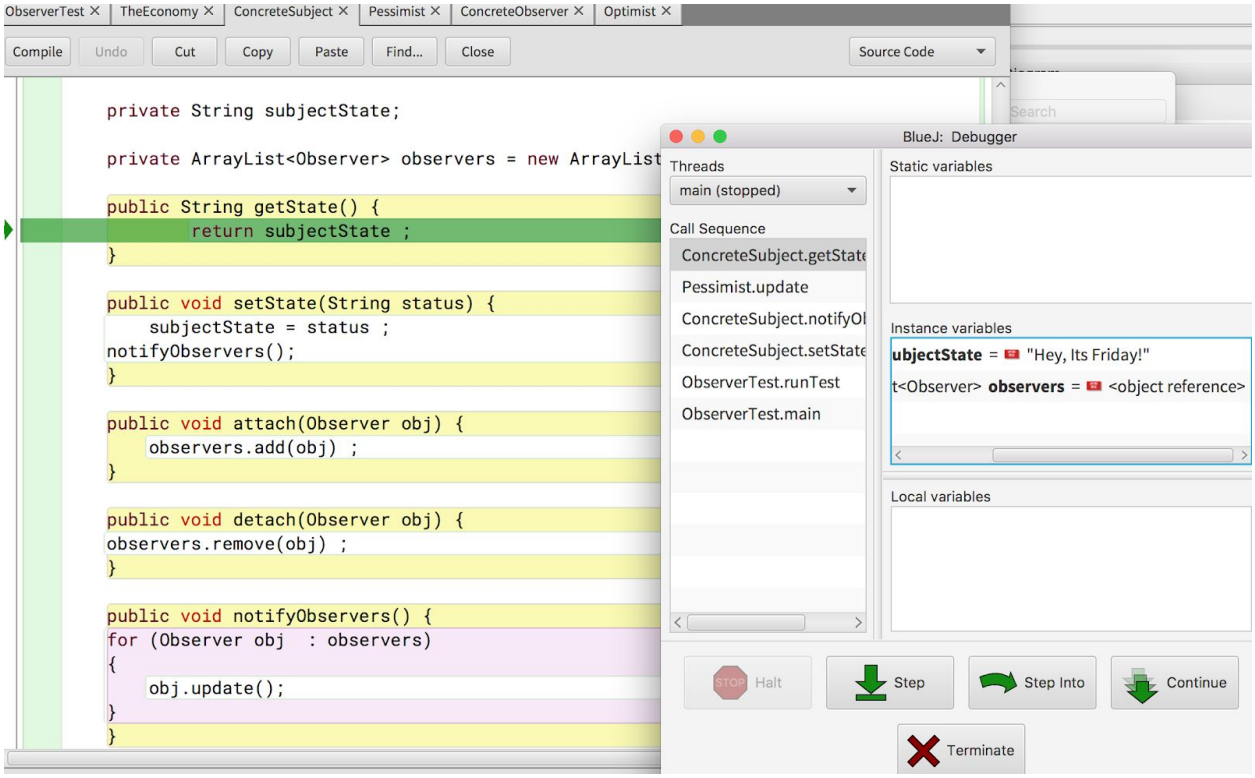


Step 7.1.1: update() : void

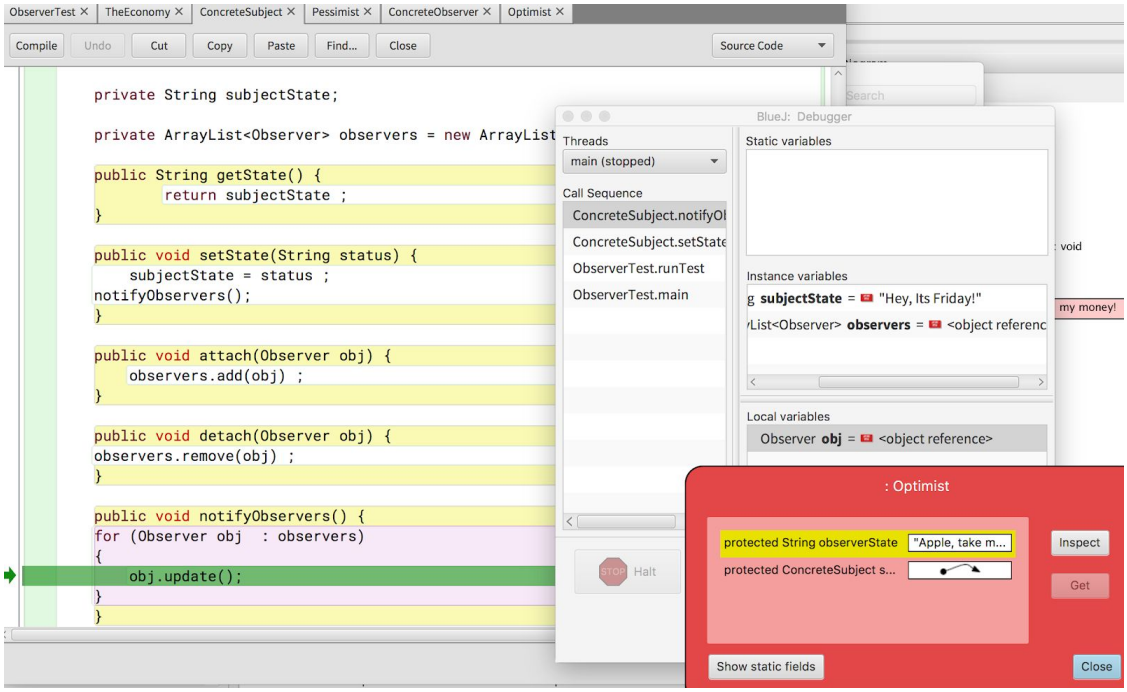




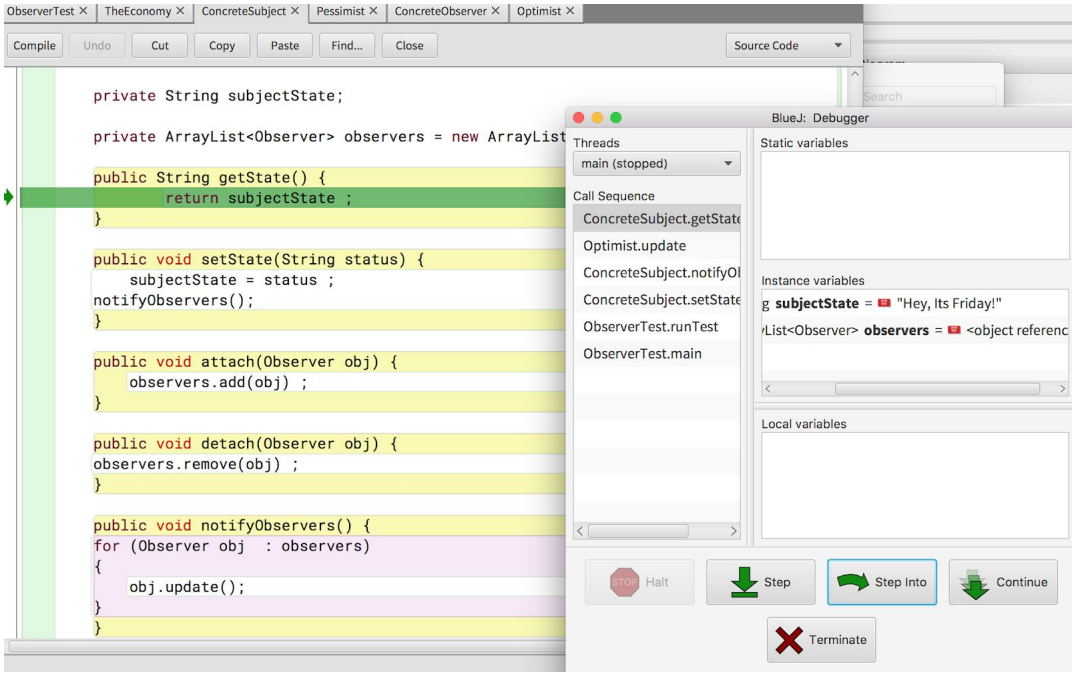
Step 7.1.1.1: getState() : String



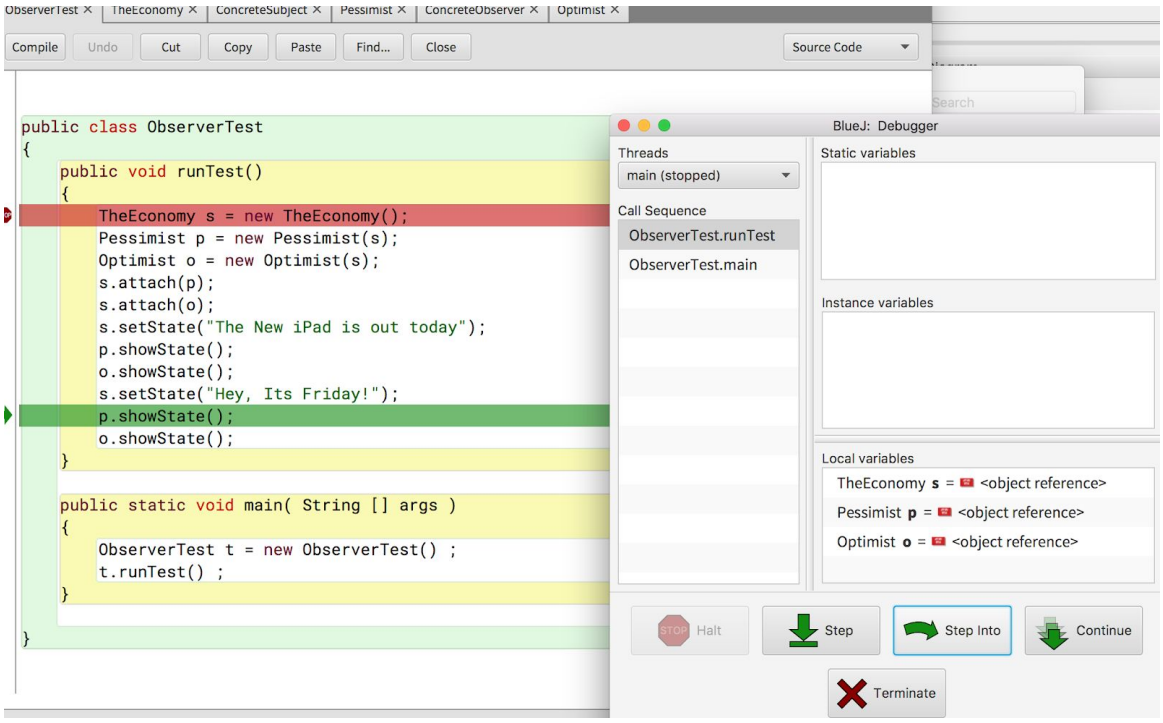
Step 7.1.2: update() : void

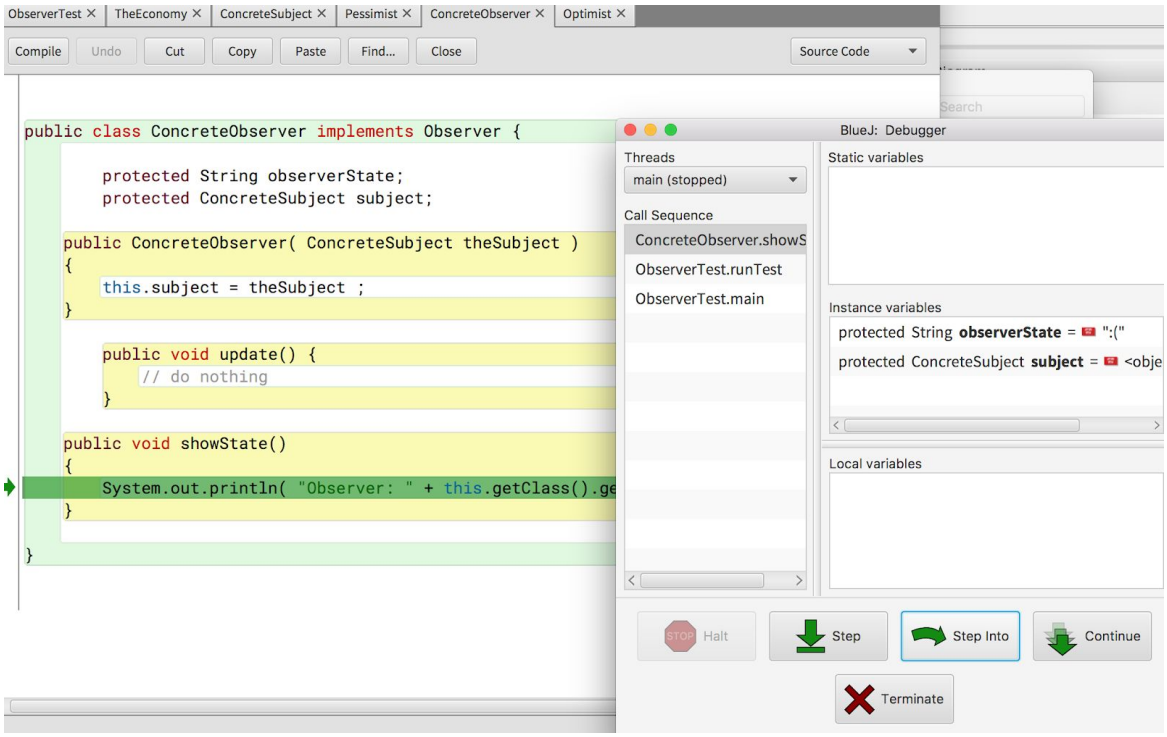


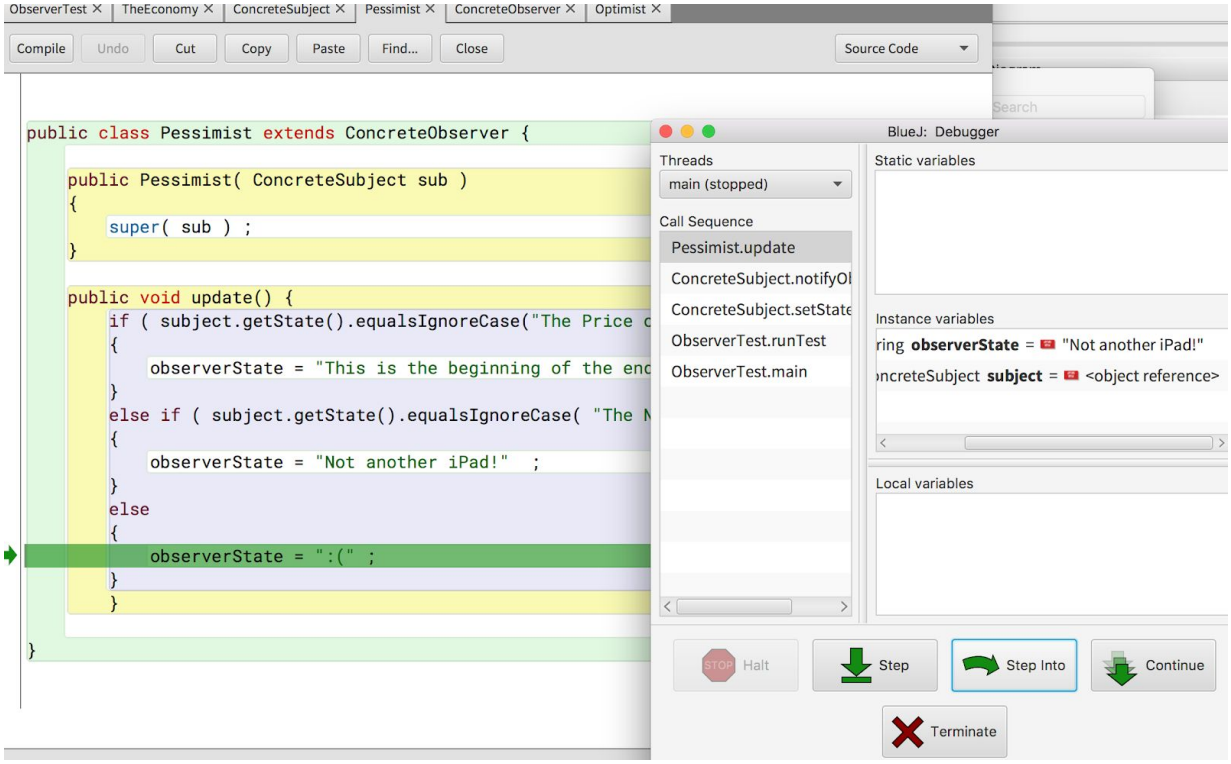
Step 7.1.2.1: getState() : String



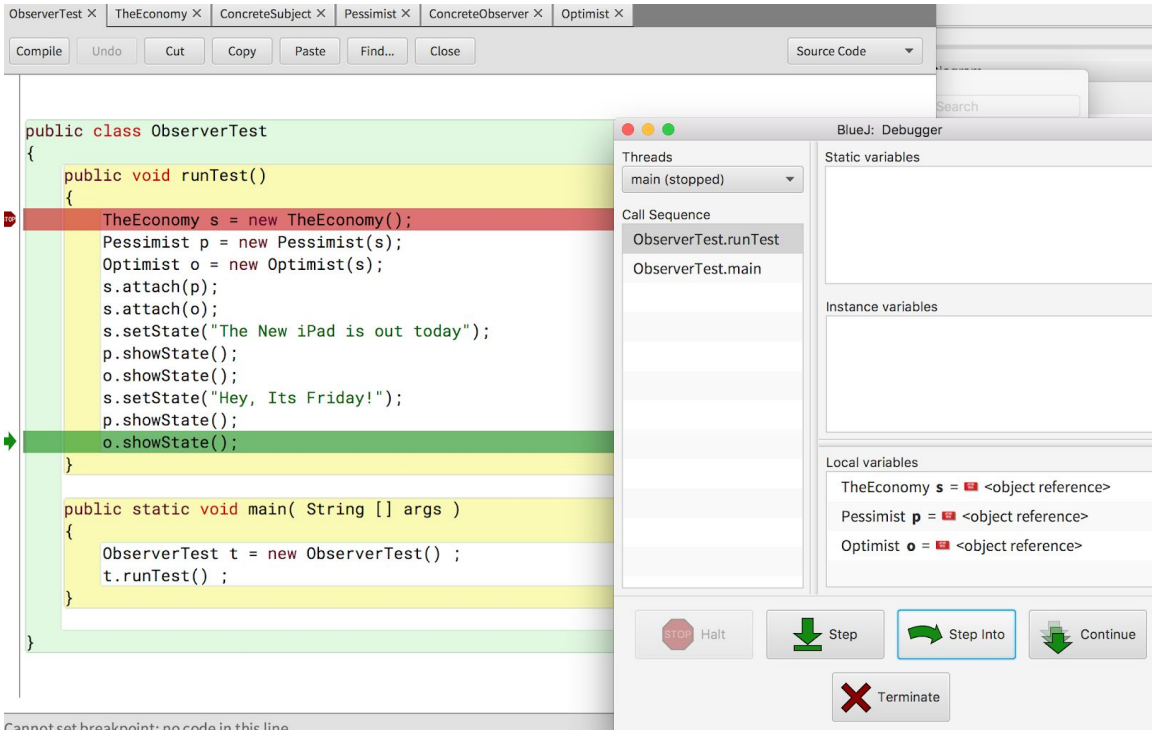
Step 8: showState() : void

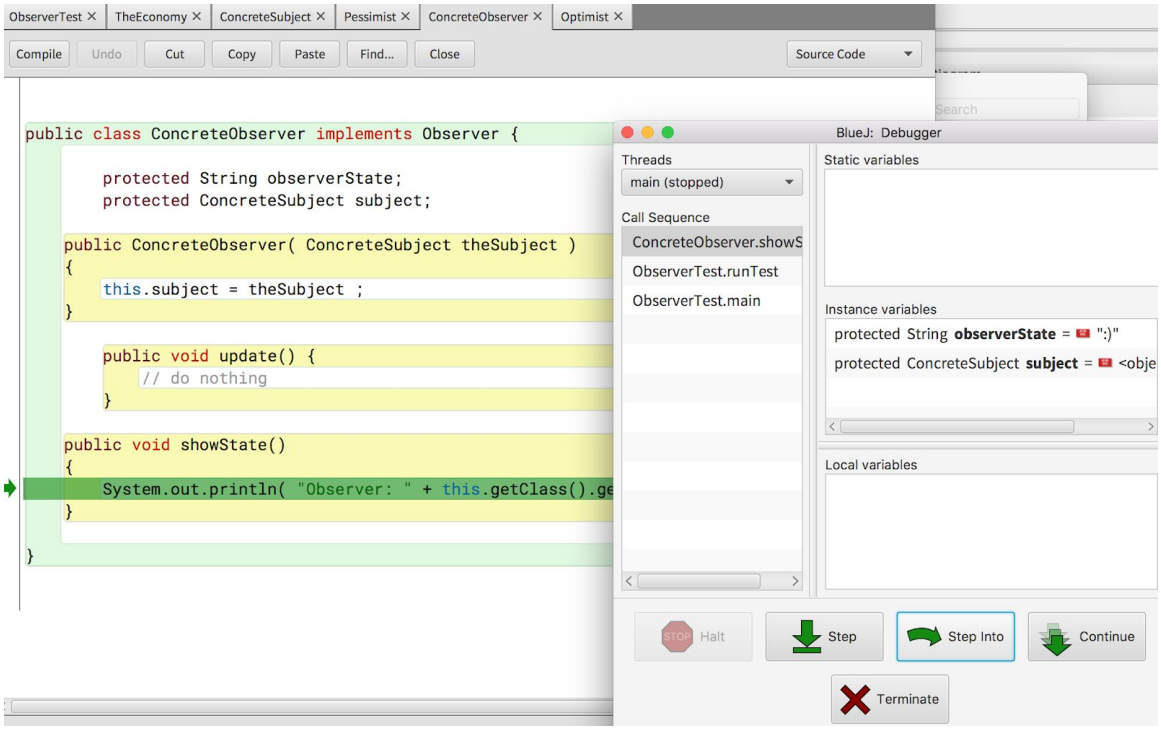


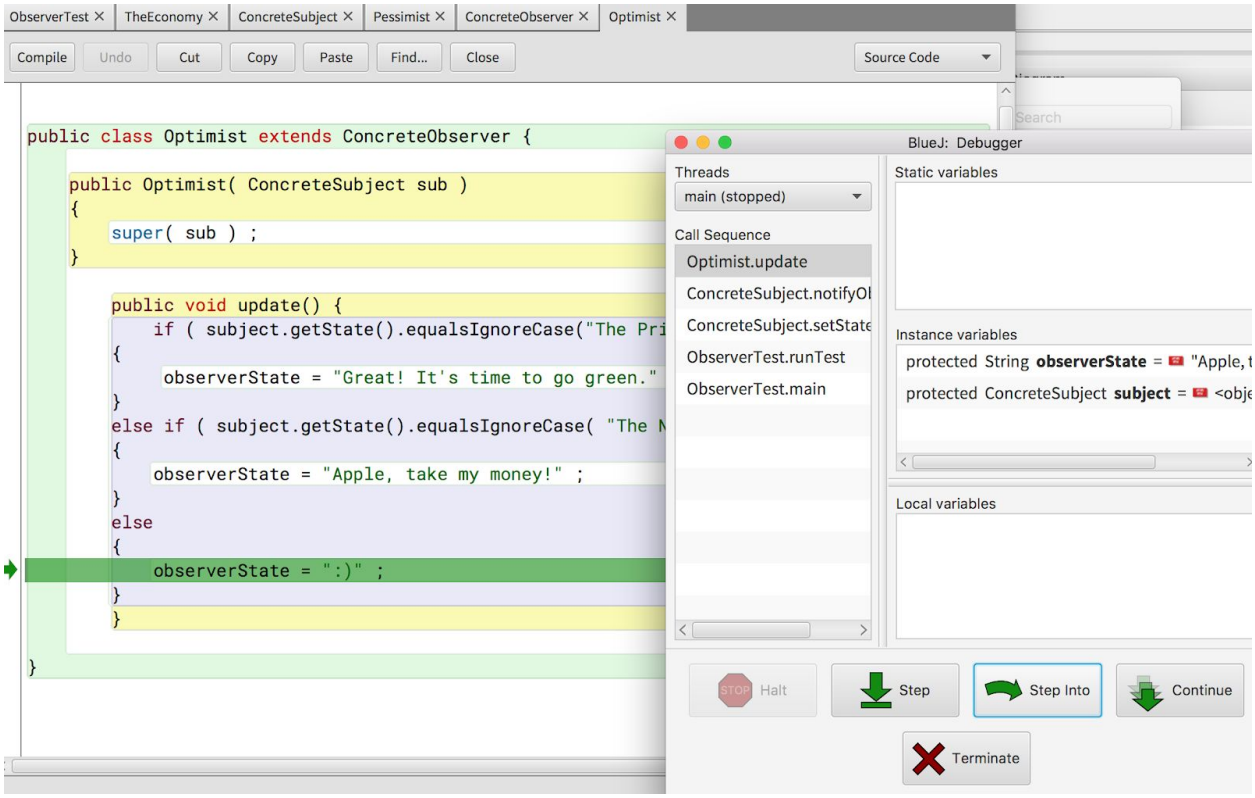



ObserverTest
TheEconomy
ConcreteSubject
Pessimist
ConcreteObserver
Optimist
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code
public class Pessimist extends ConcreteObserver {
public Pessimist(ConcreteSubject sub)
{
super(sub) ;
}
public void update() {
if (subject.getState().equalsIgnoreCase("The Price c
{
observerState = "This is the beginning of the enc
}
else if (subject.getState().equalsIgnoreCase("The N
{
observerState = "Not another iPad!" ;
}
else
{
observerState = ":(" ;
}
}
}
BlueJ: Debugger
Threads
main (stopped)
Call Sequence
Pessimist.update
ConcreteSubject.notifyOl
ConcreteSubject.setState
ObserverTest.runTest
ObserverTest.main
Static variables
Instance variables
ring observerState = "Not another iPad!"
ncreteSubject subject = <object reference>
Local variables
Halt
Step
Step Into
Continue
Terminate

Step 9: showState() : void






ObserverTest
TheEconomy
ConcreteSubject
Pessimist
ConcreteObserver
Optimist
Compile
Undo
Cut
Copy
Paste
Find...
Close
Source Code
public class Optimist extends ConcreteObserver {
public Optimist(ConcreteSubject sub)
{
super(sub) ;
}
public void update() {
if (subject.getState().equalsIgnoreCase("The Pri
{
observerState = "Great! It's time to go green."
}
else if (subject.getState().equalsIgnoreCase("The N
{
observerState = "Apple, take my money!" ;
}
else
{
observerState = ":)" ;
}
}
}
BlueJ: Debugger
Threads
main (stopped)
Call Sequence
Optimist.update
ConcreteSubject.notifyOl
ConcreteSubject.setState
ObserverTest.runTest
ObserverTest.main
Static variables
Instance variables
protected String observerState = "Apple, t
protected ConcreteSubject subject = <obje
Local variables
Halt
Step
Step Into
Continue
Terminate