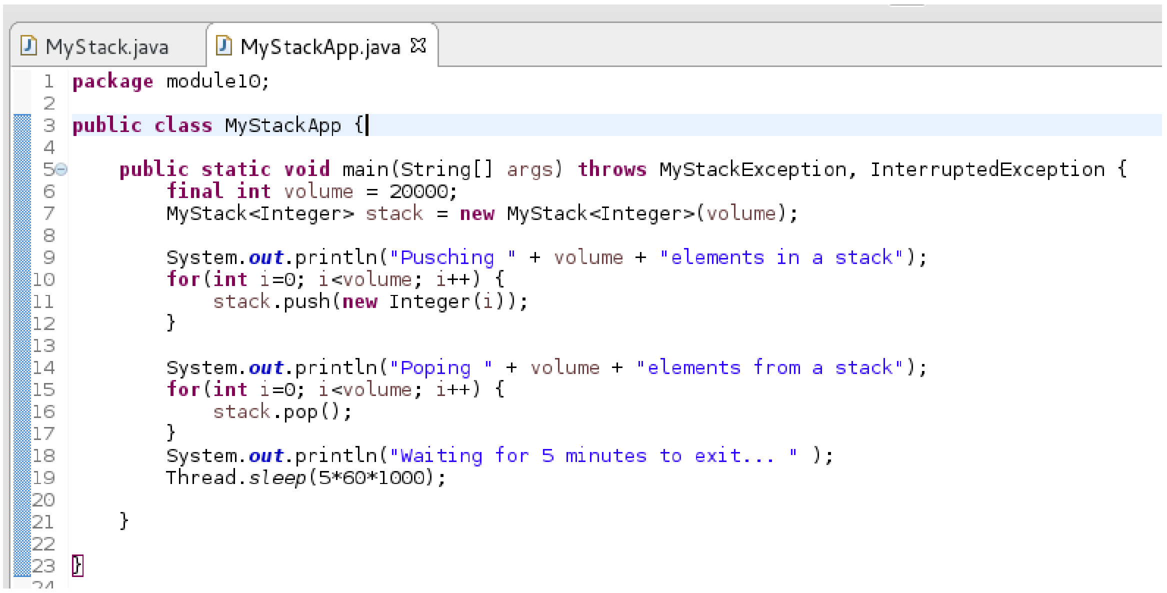
Module 10 - Java
Making Code Robust

Advanced Java Certification Training

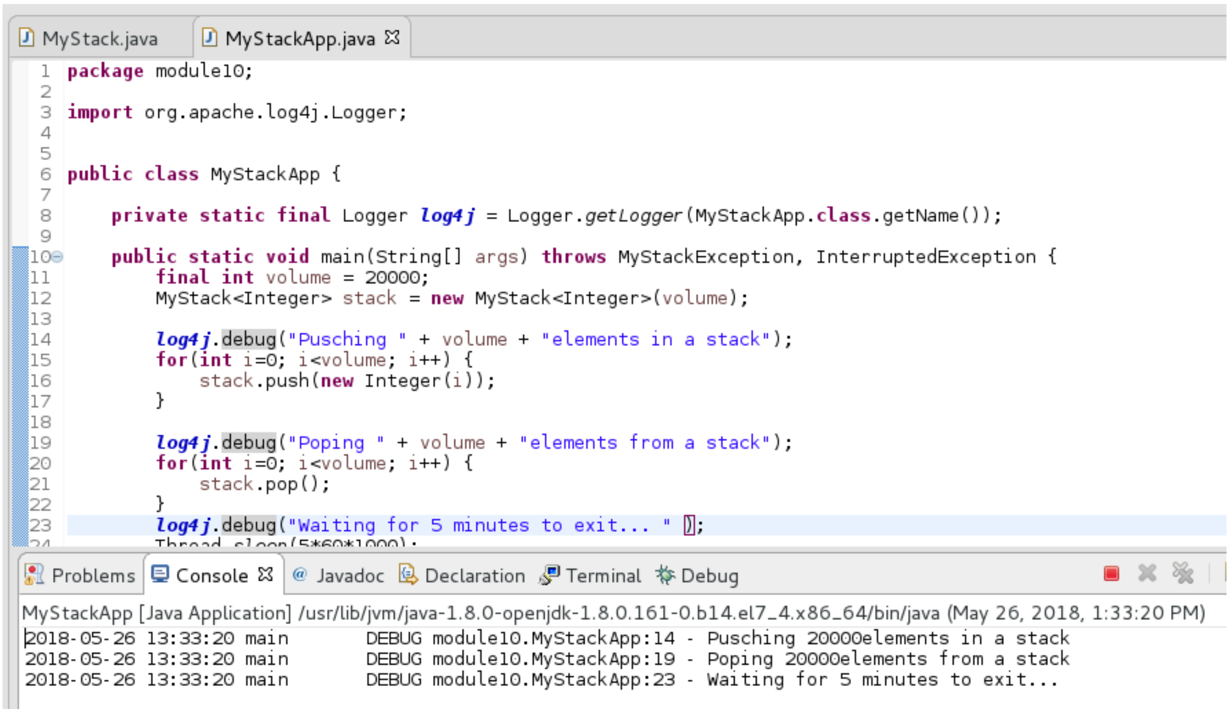
Akram M'Tir

1. Modify the class module10.mem.leaks.MyStack to eliminate the memory leak.

- Run the VisualVM to to confirm the results.

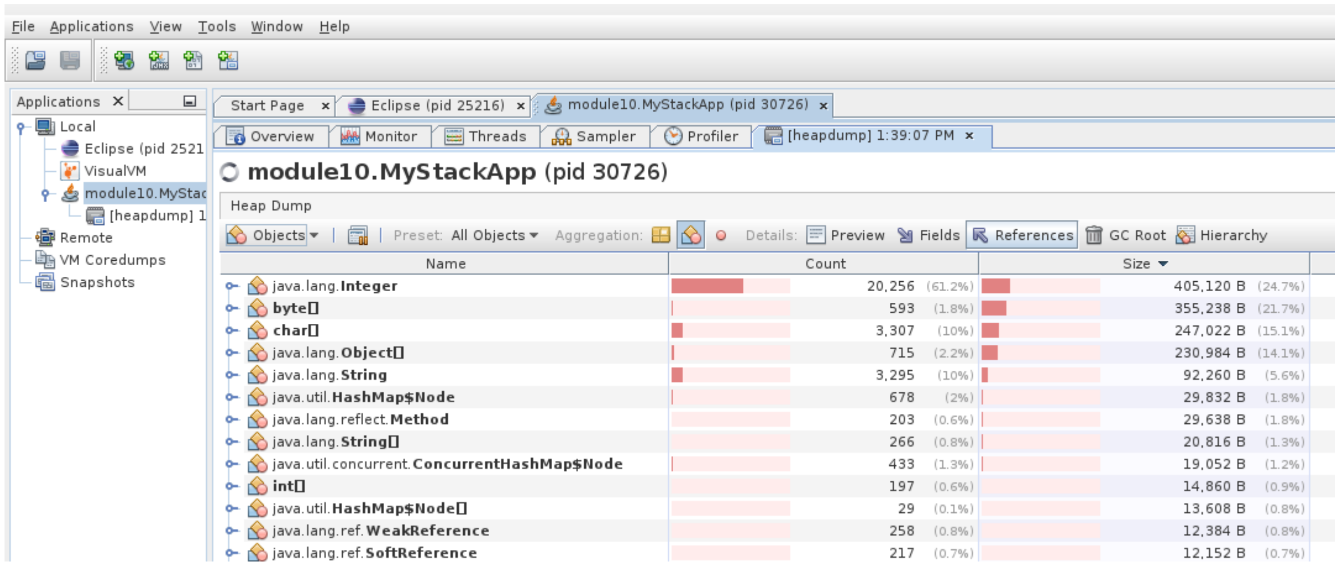


```java
package module10;

public class MyStackApp {

    public static void main(String[] args) throws MyStackException, InterruptedException {
        final int volume = 20000;
        MyStack<Integer> stack = new MyStack<Integer>(volume);

        System.out.println("Pusching " + volume + "elements in a stack");
        for(int i=0; i<volume; i++) {
            stack.push(new Integer(i));
        }

        System.out.println("Poping " + volume + "elements from a stack");
        for(int i=0; i<volume; i++) {
            stack.pop();
        }
        System.out.println("Waiting for 5 minutes to exit... " );
        Thread.sleep(5*60*1000);

    }

}
```



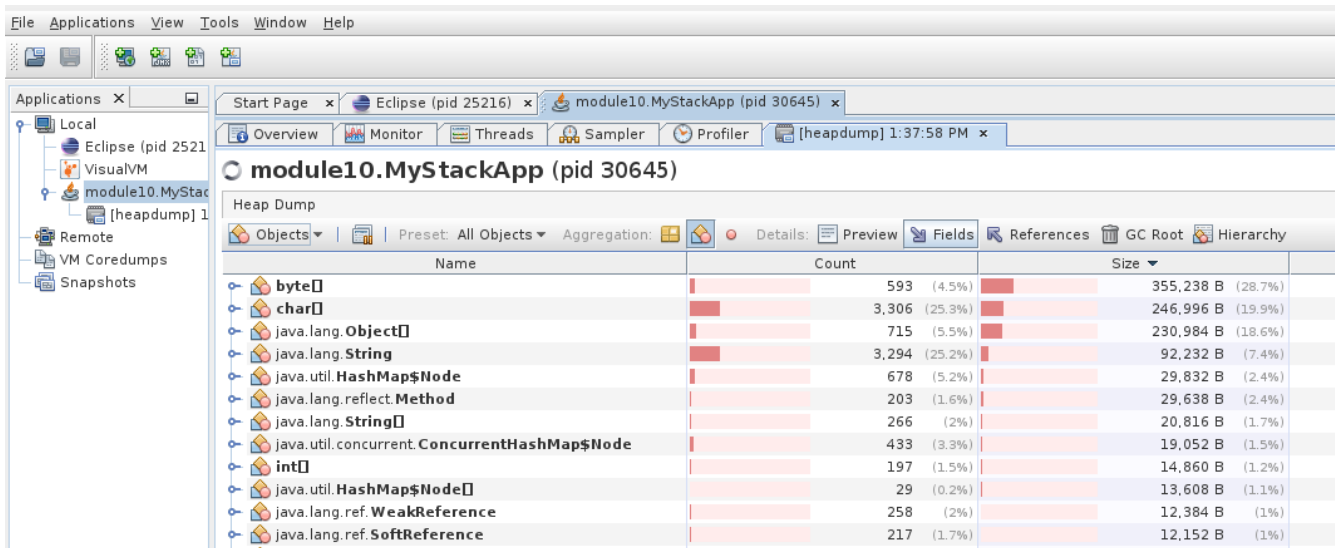
```java
package module10;

import org.apache.log4j.Logger;


public class MyStackApp {

    private static final Logger log4j = Logger.getLogger(MyStackApp.class.getName());

    public static void main(String[] args) throws MyStackException, InterruptedException {
        final int volume = 20000;
        MyStack<Integer> stack = new MyStack<Integer>(volume);

        log4j.debug("Pusching " + volume + "elements in a stack");
        for(int i=0; i<volume; i++) {
            stack.push(new Integer(i));
        }

        log4j.debug("Poping " + volume + "elements from a stack");
        for(int i=0; i<volume; i++) {
            stack.pop();
        }
        log4j.debug("Waiting for 5 minutes to exit... ");
        Thread.sleep(5*60*1000);
```

```
MyStackApp [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 26, 2018, 1:33:20 PM)
2018-05-26 13:33:20 main        DEBUG module10.MyStackApp:14 - Pusching 20000elements in a stack
2018-05-26 13:33:20 main        DEBUG module10.MyStackApp:19 - Poping 20000elements from a stack
2018-05-26 13:33:20 main        DEBUG module10.MyStackApp:23 - Waiting for 5 minutes to exit...
```



module10.MyStackApp (pid 30726)

Name	Count		Size	
java.lang.**Integer**	20,256	(61.2%)	405,120 B	(24.7%)
byte[]	593	(1.8%)	355,238 B	(21.7%)
char[]	3,307	(10%)	247,022 B	(15.1%)
java.lang.**Object[]**	715	(2.2%)	230,984 B	(14.1%)
java.lang.**String**	3,295	(10%)	92,260 B	(5.6%)
java.util.**HashMap$Node**	678	(2%)	29,832 B	(1.8%)
java.lang.reflect.**Method**	203	(0.6%)	29,638 B	(1.8%)
java.lang.**String[]**	266	(0.8%)	20,816 B	(1.3%)
java.util.concurrent.**ConcurrentHashMap$Node**	433	(1.3%)	19,052 B	(1.2%)
int[]	197	(0.6%)	14,860 B	(0.9%)
java.util.**HashMap$Node[]**	29	(0.1%)	13,608 B	(0.8%)
java.lang.ref.**WeakReference**	258	(0.8%)	12,384 B	(0.8%)
java.lang.ref.**SoftReference**	217	(0.7%)	12,152 B	(0.7%)



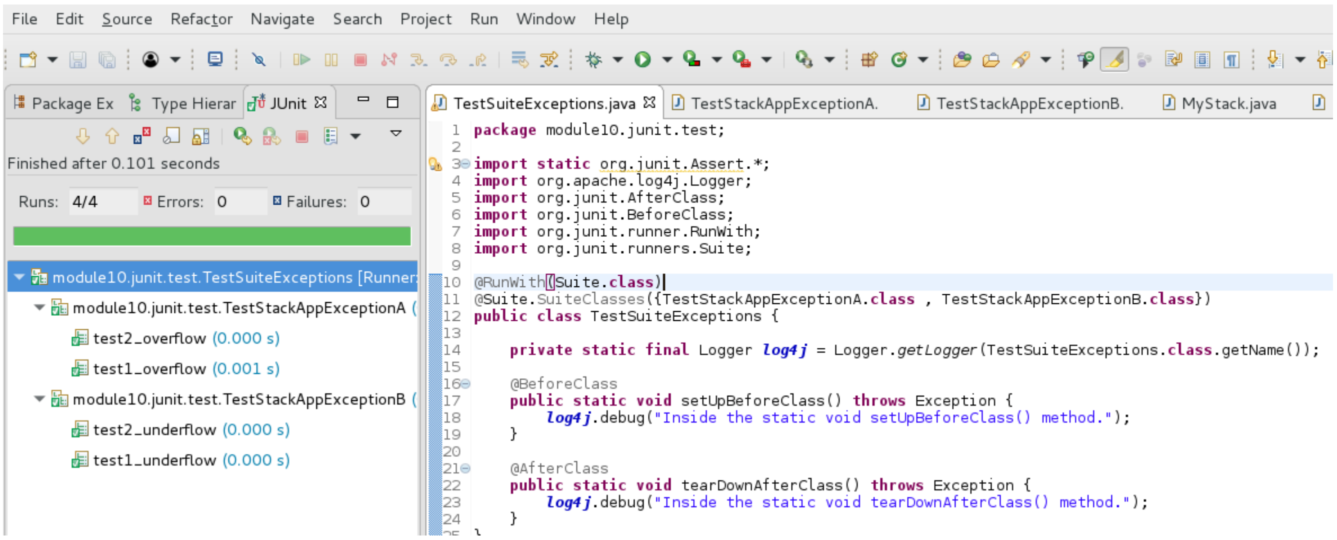
module10.MyStackApp (pid 30645)

Name	Count		Size	
byte[]	593	(4.5%)	355,238 B	(28.7%)
char[]	3,306	(25.3%)	246,996 B	(19.9%)
java.lang.**Object[]**	715	(5.5%)	230,984 B	(18.6%)
java.lang.**String**	3,294	(25.2%)	92,232 B	(7.4%)
java.util.**HashMap$Node**	678	(5.2%)	29,832 B	(2.4%)
java.lang.reflect.**Method**	203	(1.6%)	29,638 B	(2.4%)
java.lang.**String[]**	266	(2%)	20,816 B	(1.7%)
java.util.concurrent.**ConcurrentHashMap$Node**	433	(3.3%)	19,052 B	(1.5%)
int[]	197	(1.5%)	14,860 B	(1.2%)
java.util.**HashMap$Node[]**	29	(0.2%)	13,608 B	(1.1%)
java.lang.ref.**WeakReference**	258	(2%)	12,384 B	(1%)
java.lang.ref.**SoftReference**	217	(1.7%)	12,152 B	(1%)

2. Write the relevant JUNIT4 test cases for module10.mem.MyStack.

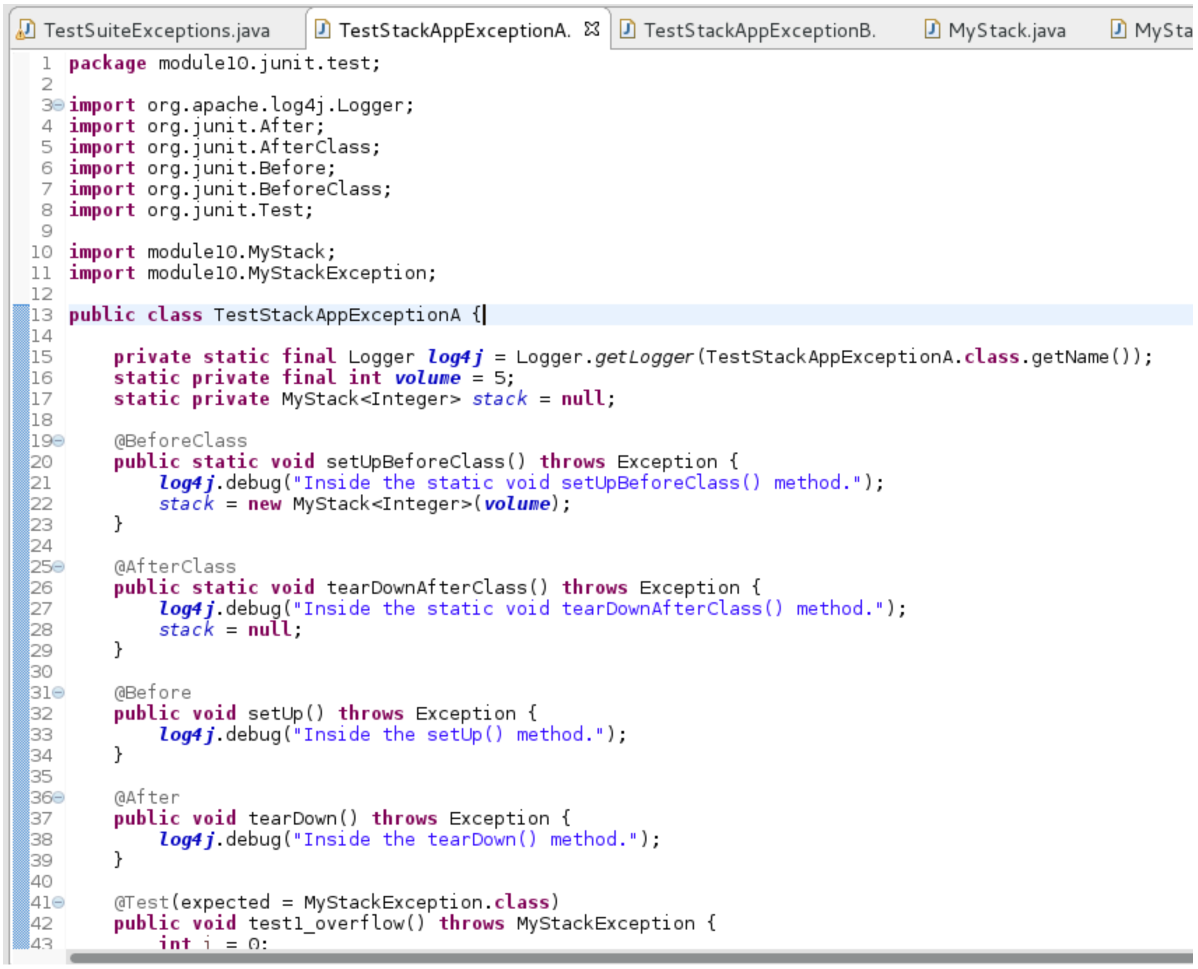
- Test all the methods
- Use the various ways to test the exceptions as discussed in this module.

A Test Suite

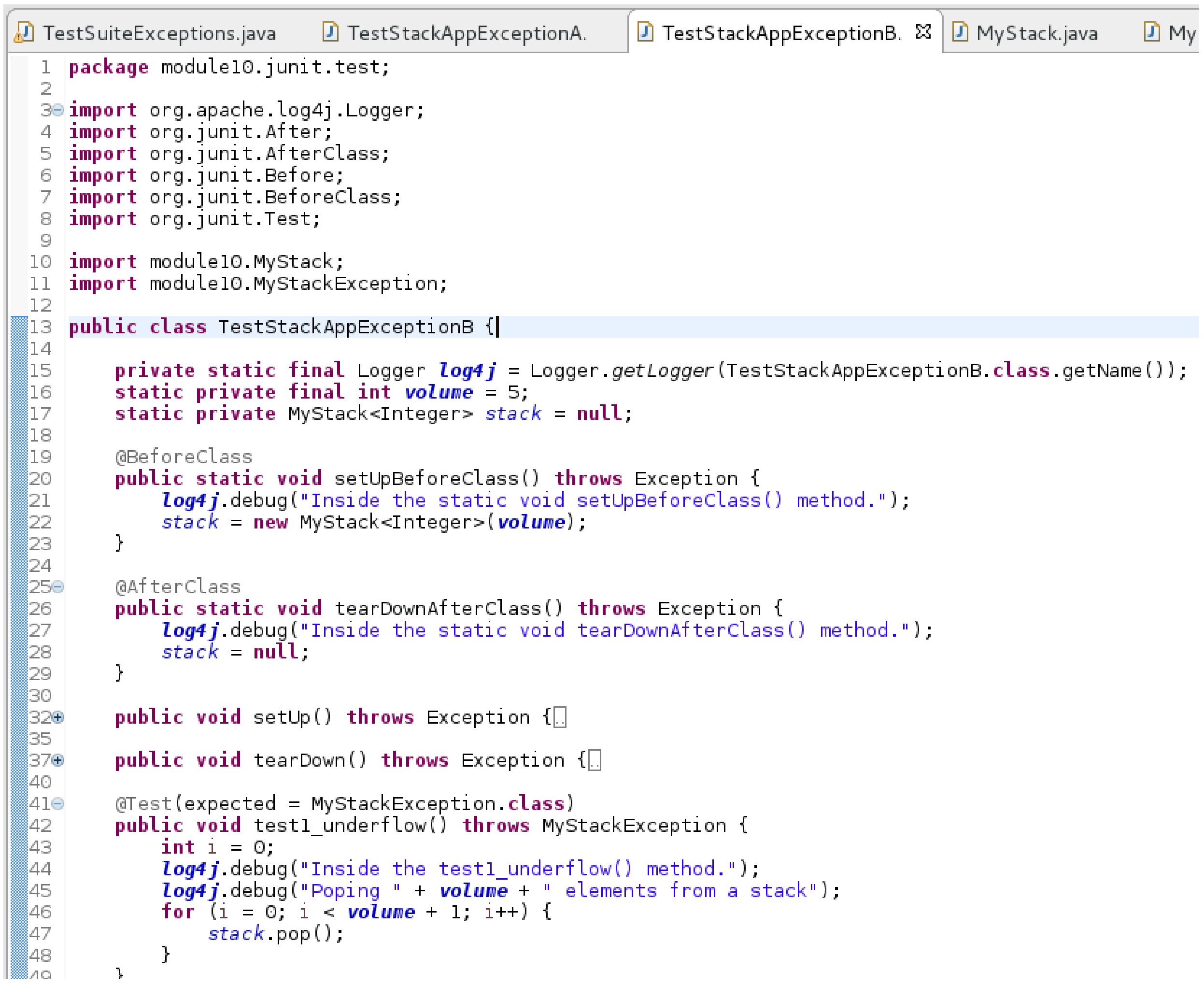


```
Finished after 0.101 seconds

Runs: 4/4    Errors: 0    Failures: 0

module10.junit.test.TestSuiteExceptions [Runner
  module10.junit.test.TestStackAppExceptionA (
    test2_overflow (0.000 s)
    test1_overflow (0.001 s)
  module10.junit.test.TestStackAppExceptionB (
    test2_underflow (0.000 s)
    test1_underflow (0.000 s)
```

```java
package module10.junit.test;

import static org.junit.Assert.*;
import org.apache.log4j.Logger;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({TestStackAppExceptionA.class , TestStackAppExceptionB.class})
public class TestSuiteExceptions {

    private static final Logger log4j = Logger.getLogger(TestSuiteExceptions.class.getName());

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        log4j.debug("Inside the static void setUpBeforeClass() method.");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        log4j.debug("Inside the static void tearDownAfterClass() method.");
    }
```

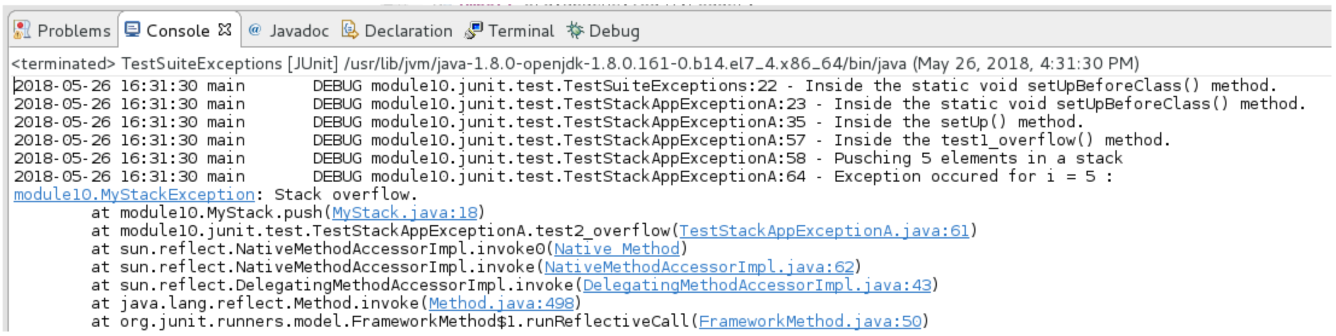


```java
package module10.junit.test;

import org.apache.log4j.Logger;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import module10.MyStack;
import module10.MyStackException;

public class TestStackAppExceptionA {

    private static final Logger log4j = Logger.getLogger(TestStackAppExceptionA.class.getName());
    static private final int volume = 5;
    static private MyStack<Integer> stack = null;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        log4j.debug("Inside the static void setUpBeforeClass() method.");
        stack = new MyStack<Integer>(volume);
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        log4j.debug("Inside the static void tearDownAfterClass() method.");
        stack = null;
    }

    @Before
    public void setUp() throws Exception {
        log4j.debug("Inside the setUp() method.");
    }

    @After
    public void tearDown() throws Exception {
        log4j.debug("Inside the tearDown() method.");
    }

    @Test(expected = MyStackException.class)
    public void test1_overflow() throws MyStackException {
        int i = 0;
```



TestSuiteExceptions.java   TestStackAppExceptionA.   TestStackAppExceptionB.   MyStack.java   My

```java
package module10.junit.test;

import org.apache.log4j.Logger;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import module10.MyStack;
import module10.MyStackException;

public class TestStackAppExceptionB {

    private static final Logger log4j = Logger.getLogger(TestStackAppExceptionB.class.getName());
    static private final int volume = 5;
    static private MyStack<Integer> stack = null;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        log4j.debug("Inside the static void setUpBeforeClass() method.");
        stack = new MyStack<Integer>(volume);
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        log4j.debug("Inside the static void tearDownAfterClass() method.");
        stack = null;
    }

    public void setUp() throws Exception {

    public void tearDown() throws Exception {

    @Test(expected = MyStackException.class)
    public void test1_underflow() throws MyStackException {
        int i = 0;
        log4j.debug("Inside the test1_underflow() method.");
        log4j.debug("Poping " + volume + " elements from a stack");
        for (i = 0; i < volume + 1; i++) {
            stack.pop();
        }
    }
```

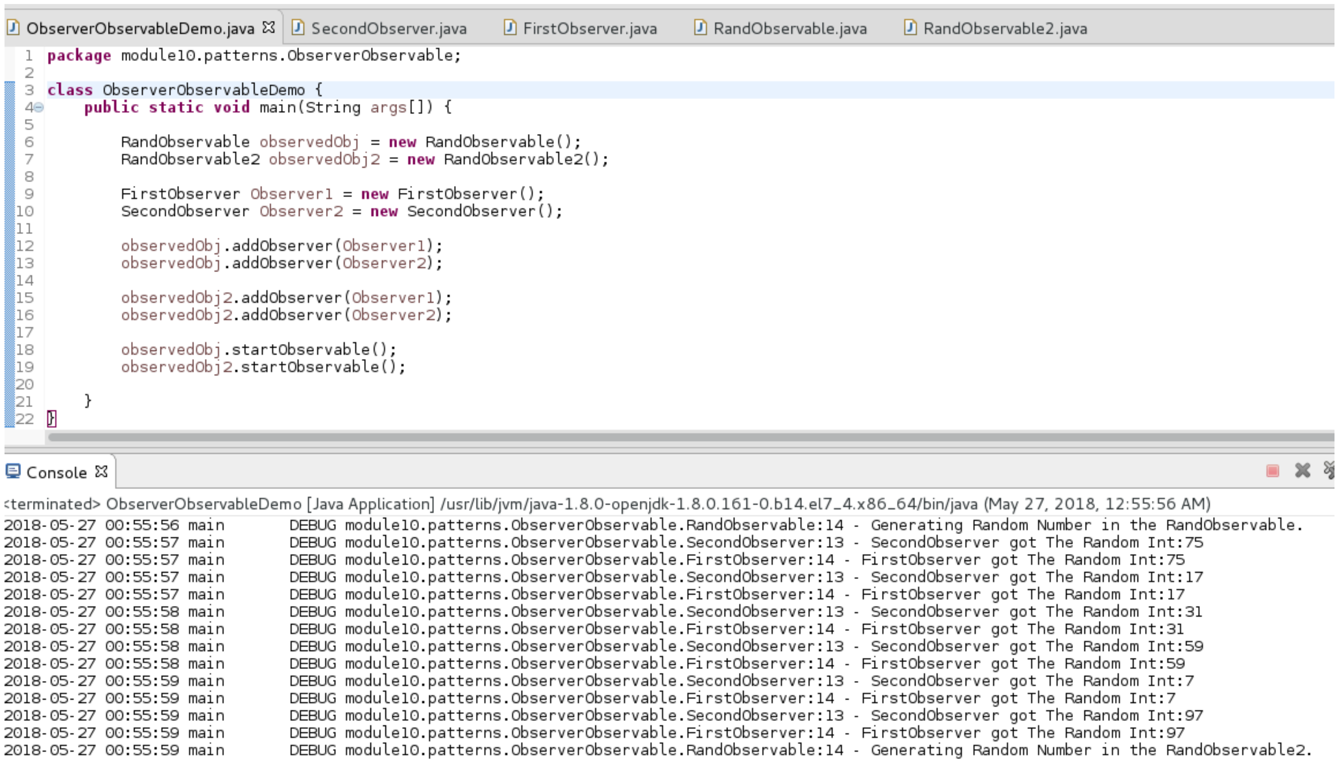


Problems   Console   Javadoc   Declaration   Terminal   Debug

```
<terminated> TestSuiteExceptions [JUnit] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 26, 2018, 4:31:30 PM)
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestSuiteExceptions:22 - Inside the static void setUpBeforeClass() method.
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestStackAppExceptionA:23 - Inside the static void setUpBeforeClass() method.
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestStackAppExceptionA:35 - Inside the setUp() method.
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestStackAppExceptionA:57 - Inside the test1_overflow() method.
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestStackAppExceptionA:58 - Pusching 5 elements in a stack
2018-05-26 16:31:30 main        DEBUG module10.junit.test.TestStackAppExceptionA:64 - Exception occured for i = 5 :
module10.MyStackException: Stack overflow.
        at module10.MyStack.push(MyStack.java:18)
        at module10.junit.test.TestStackAppExceptionA.test2_overflow(TestStackAppExceptionA.java:61)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
```

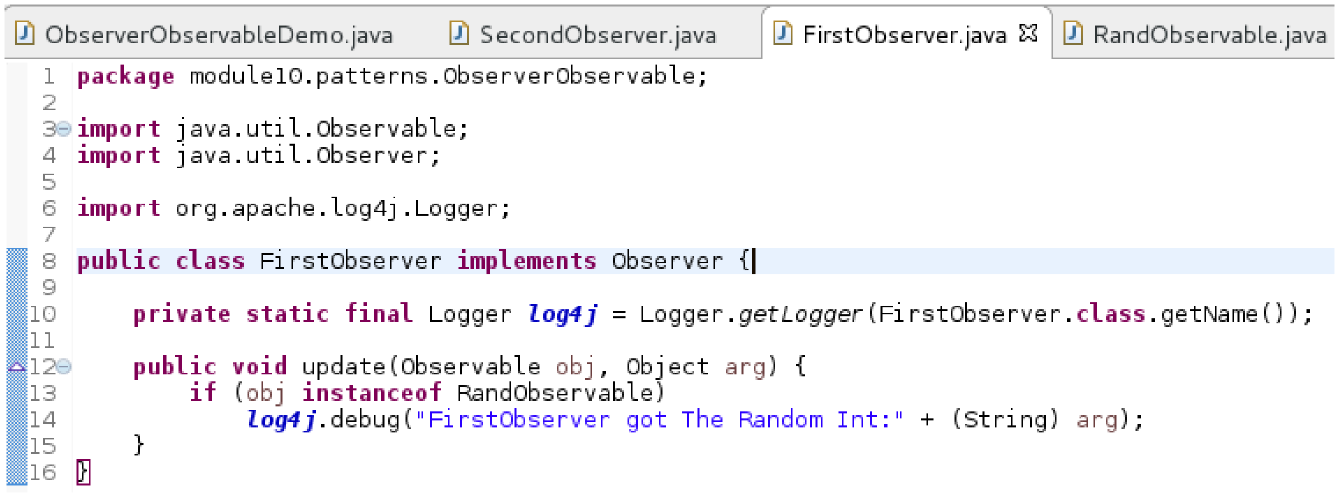
3. Write a programme to generate N random numbers from a subject.

- Notify observers only if a number is odd.
- Use the java.util.Observer & java.util.Observable
- The observer can be registered to multiple subjects.
- Observer only consumes odd random numbers
- Ignores notification from other subjects, test this with a dummy subject.

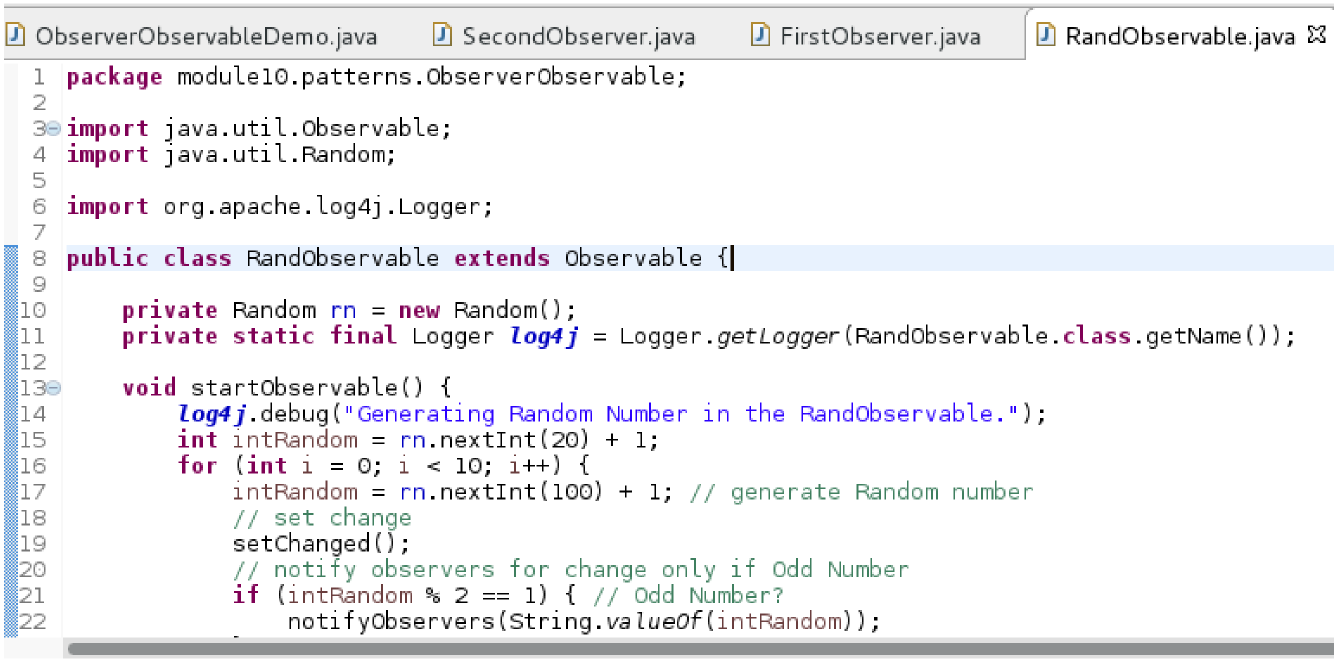


```java
package module10.patterns.ObserverObservable;

class ObserverObservableDemo {
    public static void main(String args[]) {

        RandObservable observedObj  = new RandObservable();
        RandObservable2 observedObj2 = new RandObservable2();

        FirstObserver Observer1 = new FirstObserver();
        SecondObserver Observer2 = new SecondObserver();

        observedObj.addObserver(Observer1);
        observedObj.addObserver(Observer2);

        observedObj2.addObserver(Observer1);
        observedObj2.addObserver(Observer2);

        observedObj.startObservable();
        observedObj2.startObservable();

    }
}
```

```
<terminated> ObserverObservableDemo [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 27, 2018, 12:55:56 AM)
2018-05-27 00:55:56 main        DEBUG module10.patterns.ObserverObservable.RandObservable:14 - Generating Random Number in the RandObservable.
2018-05-27 00:55:57 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:75
2018-05-27 00:55:57 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:75
2018-05-27 00:55:57 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:17
2018-05-27 00:55:57 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:17
2018-05-27 00:55:58 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:31
2018-05-27 00:55:58 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:31
2018-05-27 00:55:58 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:59
2018-05-27 00:55:58 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:59
2018-05-27 00:55:59 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:7
2018-05-27 00:55:59 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:7
2018-05-27 00:55:59 main        DEBUG module10.patterns.ObserverObservable.SecondObserver:13 - SecondObserver got The Random Int:97
2018-05-27 00:55:59 main        DEBUG module10.patterns.ObserverObservable.FirstObserver:14 - FirstObserver got The Random Int:97
2018-05-27 00:55:59 main        DEBUG module10.patterns.ObserverObservable.RandObservable:14 - Generating Random Number in the RandObservable2.
```



```java
package module10.patterns.ObserverObservable;

import java.util.Observable;
import java.util.Observer;

import org.apache.log4j.Logger;

public class FirstObserver implements Observer {

    private static final Logger log4j = Logger.getLogger(FirstObserver.class.getName());

    public void update(Observable obj, Object arg) {
        if (obj instanceof RandObservable)
            log4j.debug("FirstObserver got The Random Int:" + (String) arg);
    }
}
```



```java
package module10.patterns.ObserverObservable;

import java.util.Observable;
import java.util.Random;

import org.apache.log4j.Logger;

public class RandObservable extends Observable {

    private Random rn = new Random();
    private static final Logger log4j = Logger.getLogger(RandObservable.class.getName());

    void startObservable() {
        log4j.debug("Generating Random Number in the RandObservable.");
        int intRandom = rn.nextInt(20) + 1;
        for (int i = 0; i < 10; i++) {
            intRandom = rn.nextInt(100) + 1; // generate Random number
            // set change
            setChanged();
            // notify observers for change only if Odd Number
            if (intRandom % 2 == 1) { // Odd Number?
                notifyObservers(String.valueOf(intRandom));
```