Software Testing Laboratory
(CS6474)
Assignment 04 :Junit and PyUnit

Tapas Manna
223CS3152
Master of Technology
223cs3152@nitrkl.ac.in

Department of Computer Science Engineering
NIT, Rourkela

February 02, 2024

Contents

1 Create a rectangle class with area and perimeter calculations. Test that class. 3
1.1 Java code . . . 3
1.2 Output screenshot . . . 4
1.3 Python code . . . 4
1.4 Output screenshot . . . 5

2 Create a string class with palindrome and similarity testing. Test that class. 6
2.1 Java code . . . 6
2.2 Output screenshot . . . 7
2.3 Python code . . . 7
2.4 Output screenshot . . . 8

3 Create 5 classes with different constructors and test the similarity of their objects. 9
3.1 Java code . . . 9
3.2 Output screenshot . . . 11
3.3 Python code . . . 11
3.4 Output screenshot . . . 13

4 Create a stack data structure class with push, pop function. Test that class. 14
4.1 Java code . . . 14
4.2 Output screenshot . . . 15
4.3 Python code . . . 15
4.4 Output screenshot . . . 16

5 Create a list data structure with the find and insert operation. Test that class. 17
5.1 Java code . . . 17
5.2 Output screenshot . . . 18
5.3 Python code . . . 18
5.4 Output screenshot . . . 19

1 Create a rectangle class with area and perimeter calculations. Test that class.

1.1 Java code

```java
package junit;

public class q1_rectangle {
  public int area(int width,int height) {
    return width*height;
  }
  public int perimeter(int width,int height) {
    return 2*(width+height);
  }

}
```

```java
package junit;
import static org.junit.Assert.assertEquals;
import org.junit.Before;
import org.junit.Test;

public class q1_rectangleTest {
  private q1_rectangle r1;

  @Before
  public void setUp() {
    r1=new q1_rectangle();
  }

  @Test
  public void testArea() {
    assertEquals(6,r1.area(2, 3));
  }

  @Test
  public void testPerimeter() {
    assertEquals(10,r1.perimeter(2, 3));
  }

}
```

1.2 Output screenshot

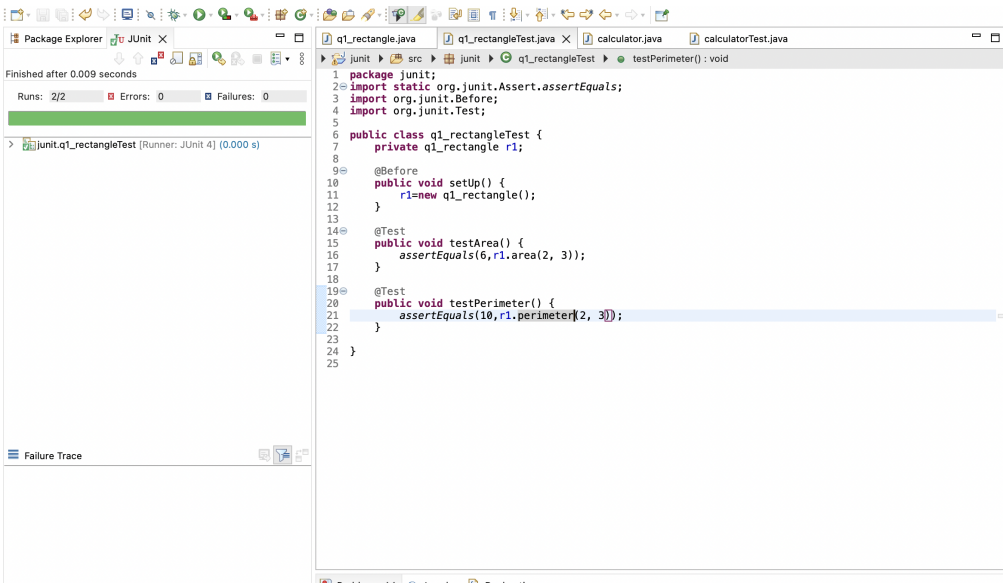

Figure 1: output

1.3 Python code

```python
import unittest
# our code to be tested
class Rectangle:
    def __init__(self,width,height):
        self.width=width
        self.height=height

    def get_perimeter(self):
        return 2*(self.width+self.height)

    def get_area(self):
        return self.height*self.width

    def set_width(self,width):
        if(self.width==0):
            return False
        else:
            self.width=width
            return True

    def set_height(self,height):
        if(height==0):
            return False
        else:

          self.height=height
          return True

# the test based on unittest modules
```

```
30
31 class TestGetAreaRectangle(unittest.TestCase):
32     def runTest(self):
33         rectangle=Rectangle(3,2)
34
35         self.assertEqual(10,rectangle.get_perimeter())
36         self.assertEqual(6,rectangle.get_area())
37
38
39
40
41
42
43 unittest.main()
```

1.4 Output screenshot

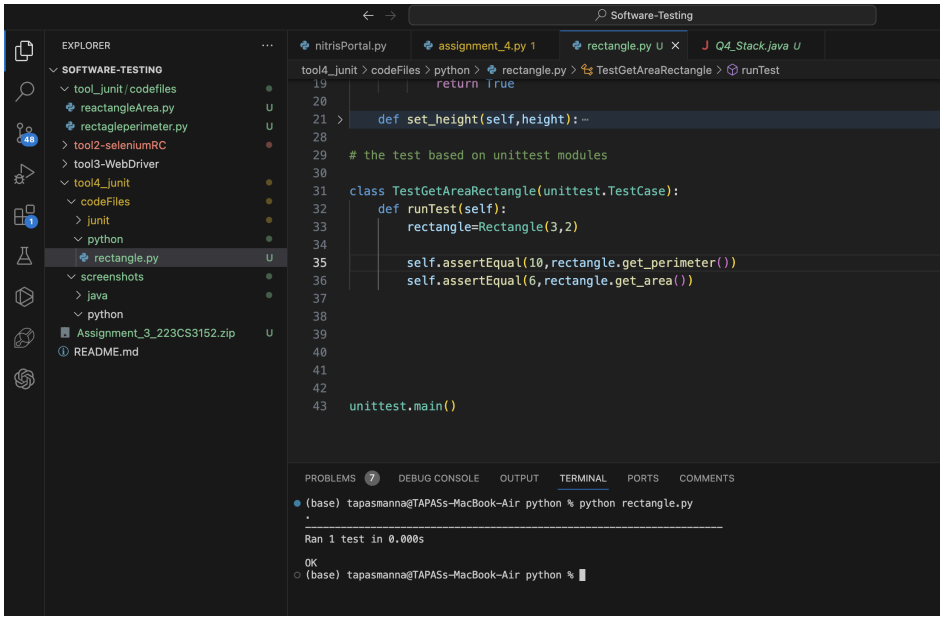

Figure 2: output

2 Create a string class with palindrome and similarity testing. Test that class.

2.1 Java code

```java
package junit;

public class q2_string {
  boolean ispalindrome(String str1) {
    int len=str1.length()-1;
    int i=0;
    while(i<=len) {
      if(str1.charAt(i)!=str1.charAt(len)) {
        return false;
      }
      i++;
      len--;
    }
    return true;
  }

  boolean isSimilar(String str1,String str2) {
    int len1=str1.length();
    int len2=str2.length();
    if(len1!=len2) {
      return false;
    }
    for(int i=0;i<len1;i++) {
      if(str1.charAt(i)!=str2.charAt(i)) {
        return false;
      }
    }

    return true;
  }
}
```

```java
package junit;
import static org.junit.Assert.assertTrue;
import org.junit.Before;
import org.junit.Test;

public class Q2_stringTest {
  private q2_string str;

  @Before
  public void setUp() {
    str=new q2_string();
  }

  @Test
  public void testPallindrome() {
    assertTrue(str.ispalindrome("sas"));
  }

  @Test
  public void testSimilarity() {
    assertTrue(str.isSimilar("krishna", "krishna"));
```

```
  }

}
```

2.2 Output screenshot

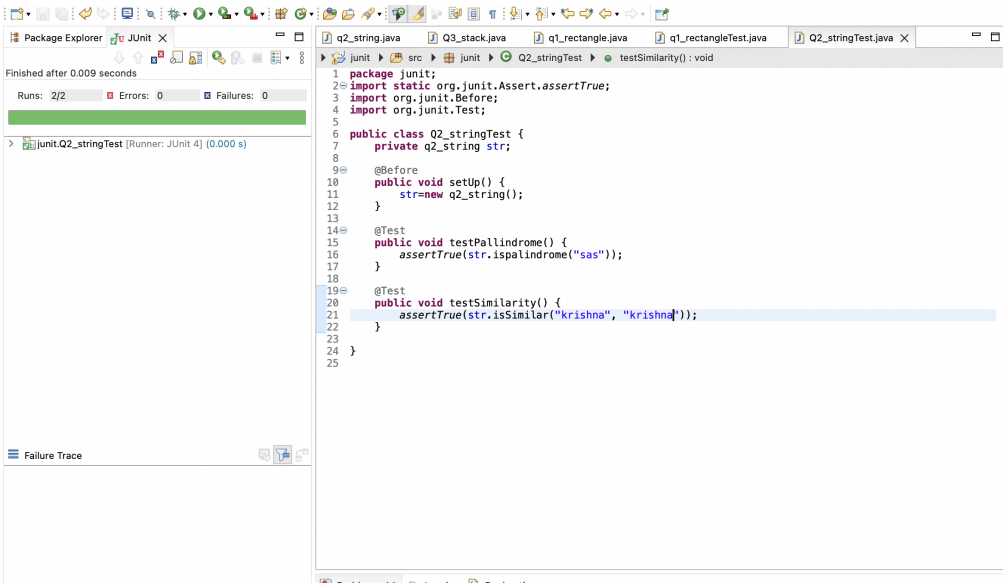

Figure 3: output

2.3 Python code

```python
import unittest

class String:
    def is_palindrome(self, str1):
        len_str1 = len(str1) - 1
        i = 0
        while i <= len_str1:
            if str1[i] != str1[len_str1]:
                return False
            i += 1
            len_str1 -= 1
        return True

    def is_similar(self, str1, str2):
        len_str1 = len(str1)
        len_str2 = len(str2)
        if len_str1 != len_str2:
            return False
        for i in range(len_str1):
            if str1[i] != str2[i]:
                return False
        return True

class TestStringMethods(unittest.TestCase):
    def test_is_palindrome(self):
```

```
        s = String()
        self.assertTrue(s.is_palindrome("aba"))

    def test_is_similar(self):
        s = String()
        self.assertTrue(s.is_similar("hello", "hello"))

if __name__ == '__main__':
    unittest.main()
```

2.4 Output screenshot

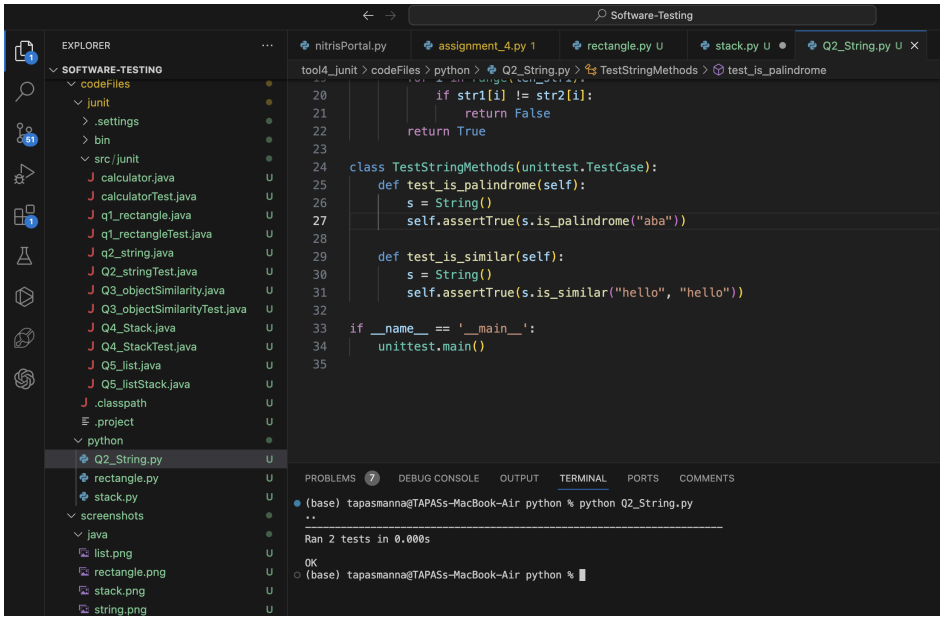

Figure 4: output

3 Create 5 classes with different constructors and test the similarity of their objects.

3.1 Java code

```
1  package junit;
2
3  public class Q3_objectSimilarity {
4
5  }
6
7  //Class 1
8  class department {
9    private String name;
10
11
12    public department(String name) {
13        this.name = name;
14    }
15
16
17    public String getName() {
18        return name;
19    }
20 }
21
22 //Class 2
23 class subject {
24   private String name;
25
26
27    public subject(String name) {
28        this.name = name;
29    }
30
31
32    public String getName() {
33        return name;
34    }
35 }
36
37 //Class 3
38
39 class book {
40   private String name;
41   private double price;
42
43   public book(String name, double price) {
44      this.name = name;
45      this.price = price;
46   }
47
48   public String getName() {
49      return name;
50   }
51
52   public double getPrice() {
53      return price;
```

```
  }
}

//Class 4
class student {
  private String name;


  public student(String name) {
     this.name = name;
  }

    public String getTitle() {
     return name;
  }
}

//class 5
class  result {
  private double cgpa;


  public result(double cgpa) {
     this.cgpa = cgpa;

  }


  public double getresult() {
     return cgpa;
  }


}
```

```
package junit;
import static org.junit.Assert.assertNotEquals;
import static org.junit.Assert.assertEquals;
import org.junit.Before;
import org.junit.Test;

public class Q3_objectSimilarityTest {

  private department d1,d2;
  private student stu1,stu2;
  private book b1,b2;
  private subject sub1,sub2;
  private result r1,r2;

  @Before
  public void setUp() {
    d1=new department("Computer Science");
    sub1=new subject("Maths");
    b1=new book("computer", 300.0);
    stu1=new student("Tapas");
    r1=new result(8.0);

    d2=d1;
    sub2=new subject("Maths");
```

```
    b2=new book("computer",300);
    stu2=new student("Tapas");
    r2=new result(8.0);
  }

  @Test
  public void testSimilarity() {

    assertEquals(d1,d2);
    assertNotEquals(sub1,sub2);
    assertNotEquals(b1,b2);
    assertNotEquals(stu1,stu2);
    assertNotEquals(r1,r2);
  }

}
```

3.2 Output screenshot

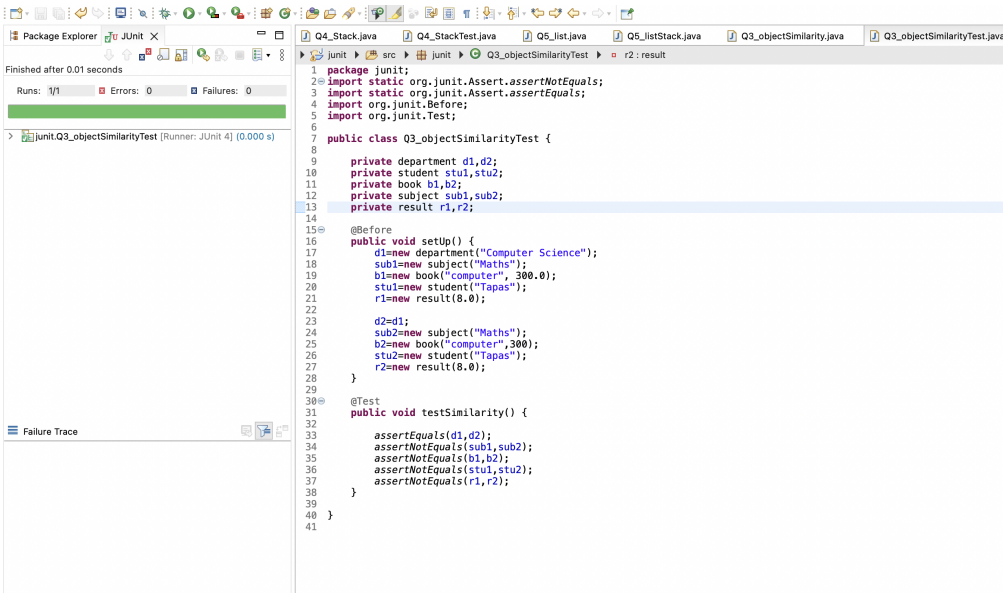

Figure 5: output

3.3 Python code

```
import unittest


class Department:
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self.name


class Subject:
```

```
13      def __init__(self, name):
14          self.name = name
15
16      def get_name(self):
17          return self.name
18
19
20  class Book:
21      def __init__(self, name, price):
22          self.name = name
23          self.price = price
24
25      def get_name(self):
26          return self.name
27
28      def get_price(self):
29          return self.price
30
31
32  class Student:
33      def __init__(self, name):
34          self.name = name
35
36      def get_title(self):
37          return self.name
38
39
40  class Result:
41      def __init__(self, cgpa):
42          self.cgpa = cgpa
43
44      def get_result(self):
45          return self.cgpa
46
47
48  class TestObjectSimilarity(unittest.TestCase):
49      def test_similarity(self):
50          department_obj1 = Department("Computer")
51          department_obj2 = department_obj1
52
53          subject_obj1 = Subject("Mathematics")
54          subject_obj2 = Subject("Mathematics")
55
56          book_obj1 = Book("Introduction to Python", 100.0)
57          book_obj2 = Book("Introduction to Python",100.0)
58
59          student_obj1 = Student("Krishna")
60          student_obj2 = Student("Krishna")
61
62          result_obj1 = Result(8.75)
63          result_obj2 = Result(8.75)
64
65          self.assertEqual(department_obj1, department_obj2)
66          self.assertNotEqual(subject_obj1, subject_obj2)
67          self.assertNotEqual(book_obj1, book_obj2)
68          self.assertNotEqual(student_obj1, student_obj2)
69          self.assertNotEqual(result_obj1, result_obj2)
70
71
```

```
72 if __name__ == '__main__':
73     unittest.main()
```

3.4 Output screenshot

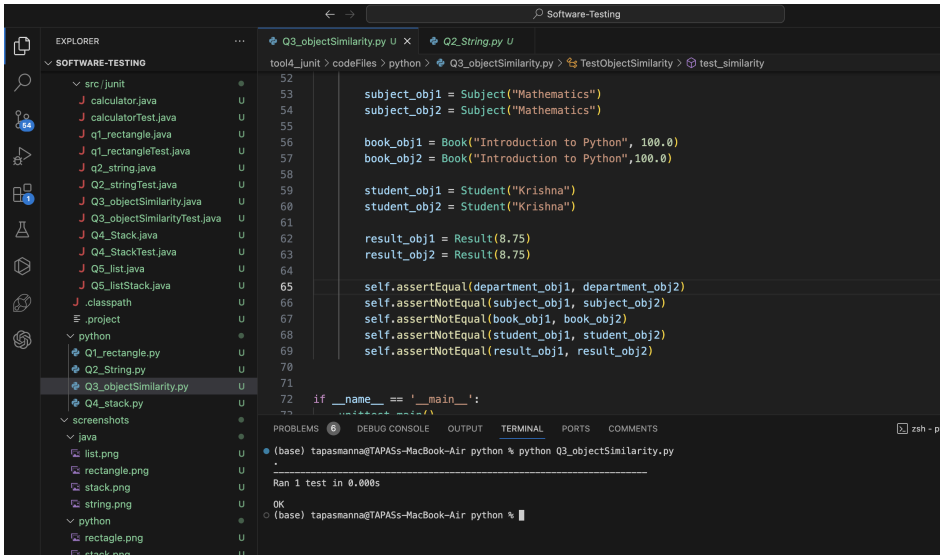

Figure 6: output

4 Create a stack data structure class with push, pop function. Test that class.

4.1 Java code

```java
package junit;

public class Q4_Stack {
  private int maxSize;
    private int top;
    private int[] stackArray;

    public Q4_Stack(int size) {
        maxSize = size;
        stackArray = new int[maxSize];
        top = -1;
    }

    public void push(int value) {
        if (top < maxSize - 1) {
            stackArray[++top] = value;
            System.out.println("Pushed: " + value);
        } else {
            System.out.println("Stack Overflow. Cannot push " + value);
        }
    }

    public int pop() {
        if (top >= 0) {
            int poppedValue = stackArray[top--];
            System.out.println("Popped: " + poppedValue);
            return poppedValue;
        } else {
            System.out.println("Stack Underflow. Cannot pop");
            return -1; // Return a sentinel value or throw an exception as per
    your requirement
        }
    }

}
```

```java
package junit;
import static org.junit.Assert.assertEquals;
import org.junit.Before;
import org.junit.Test;

public class Q4_StackTest {
  private Q4_Stack st;

  @Before
  public void setUp() {
    st=new Q4_Stack(10);
  }

  @Test
  public void testStack() {
    st.push(10);
    st.push(8);
```

```
    assertEquals(8,st.pop());
    assertEquals(10,st.pop());
  }

}
```

4.2 Output screenshot

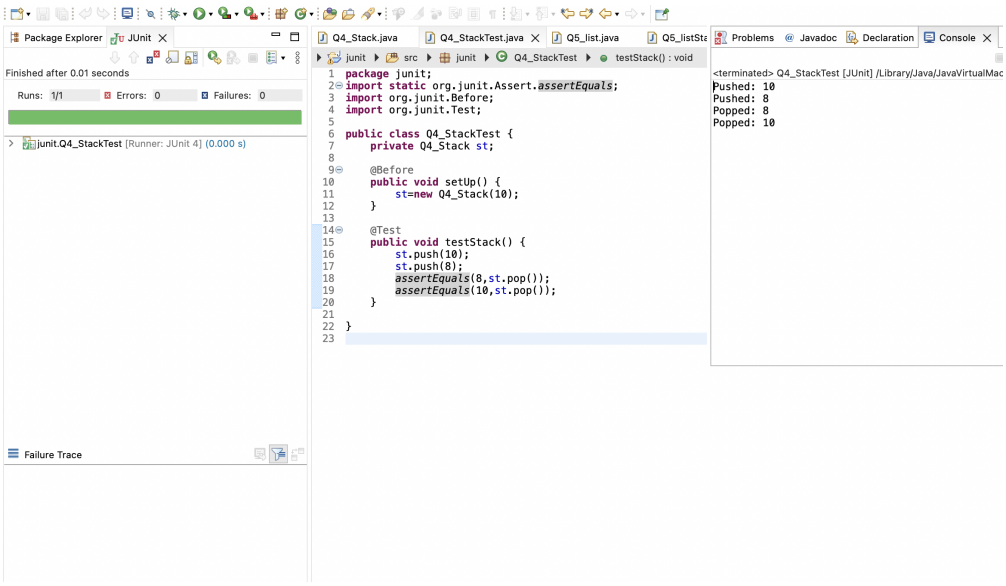

Figure 7: output

4.3 Python code

```python
import unittest

class Stack:
    def __init__(self, size):
        self.max_size = size
        self.stack_array = [0] * self.max_size
        self.top = -1

    def push(self, value):
        if self.top < self.max_size - 1:
            self.top += 1
            self.stack_array[self.top] = value
            print("Pushed:", value)
        else:
            print("Stack Overflow. Cannot push", value)

    def pop(self):
        if self.top >= 0:
            popped_value = self.stack_array[self.top]
            self.top -= 1
            print("Popped:", popped_value)
            return popped_value
        else:
```

```
            print("Stack Underflow. Cannot pop")
            return -1

class TestStack(unittest.TestCase):
    def setUp(self):
        self.st = Stack(5)

    def test_push_pop(self):
        self.st.push(6)
        self.st.push(8)
        self.assertEqual(8, self.st.pop())
        self.assertEqual(6, self.st.pop())

if __name__ == '__main__':
    unittest.main()
```

4.4 Output screenshot

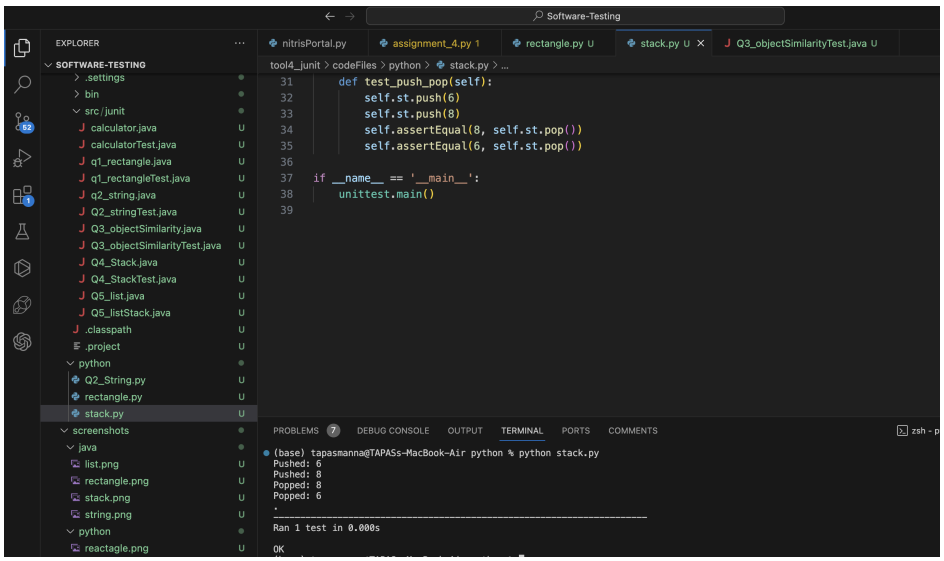

Figure 8: output

5 Create a list data structure with the find and insert operation. Test that class.

5.1 Java code

```java
package junit;

import java.util.ArrayList;

public class Q5_list {
   private ArrayList<Integer> items;

      public Q5_list() {
          this.items = new ArrayList<Integer>();
      }

      public boolean find(int key) {
          return items.contains(key);
      }

      public void insert(int data) {
          items.add(data);
      }
}
```

```java
package junit;
import static org.junit.Assert.assertTrue;
import org.junit.Before;
import org.junit.Test;

public class Q5_listStack {
  private Q5_list ls;

    @Before
    public void setUp() {
      ls=new Q5_list();
    }

    @Test
    public void testList() {
      ls.insert(10);
      ls.insert(78);
      ls.insert(7);
      ls.insert(70);
      ls.insert(78);
      assertTrue(ls.find(10));
      assertTrue(ls.find(7));

    }
}
```

5.2 Output screenshot

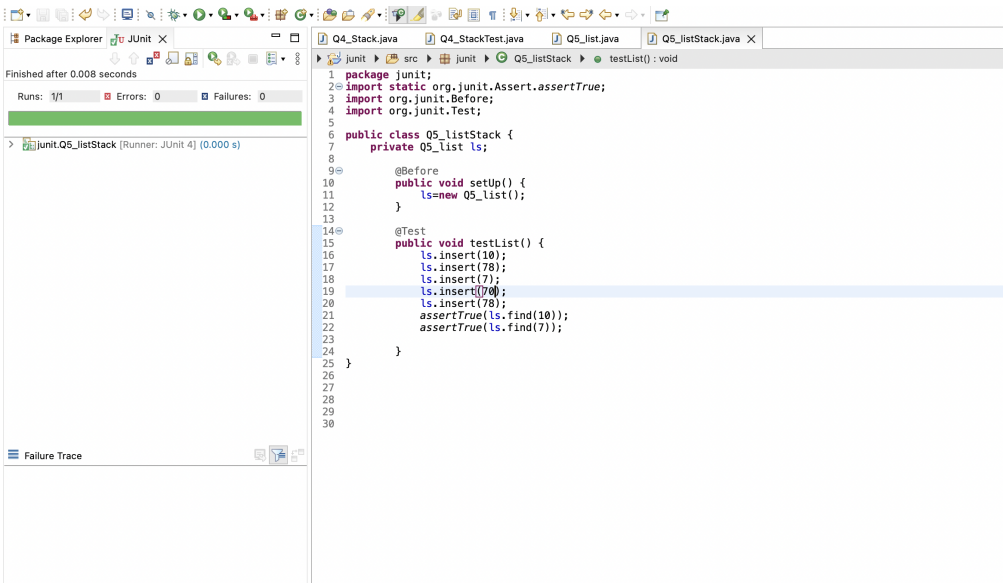

Figure 9: output

5.3 Python code

```python
import unittest

class Q5List:
    def __init__(self):
        self.items = []

    def find(self, key):
        return key in self.items

    def insert(self, data):
        self.items.append(data)


class TestQ5List(unittest.TestCase):
    def test_find_insert(self):
        ls = Q5List()

        self.assertFalse(ls.find(42))
        self.assertFalse(ls.find(10))

        ls.insert(42)
        self.assertTrue(ls.find(42))
        self.assertFalse(ls.find(10))

        ls.insert(10)
        self.assertTrue(ls.find(42))
        self.assertTrue(ls.find(10))

if __name__ == '__main__':
```

```
30      unittest.main()
```

5.4 Output screenshot

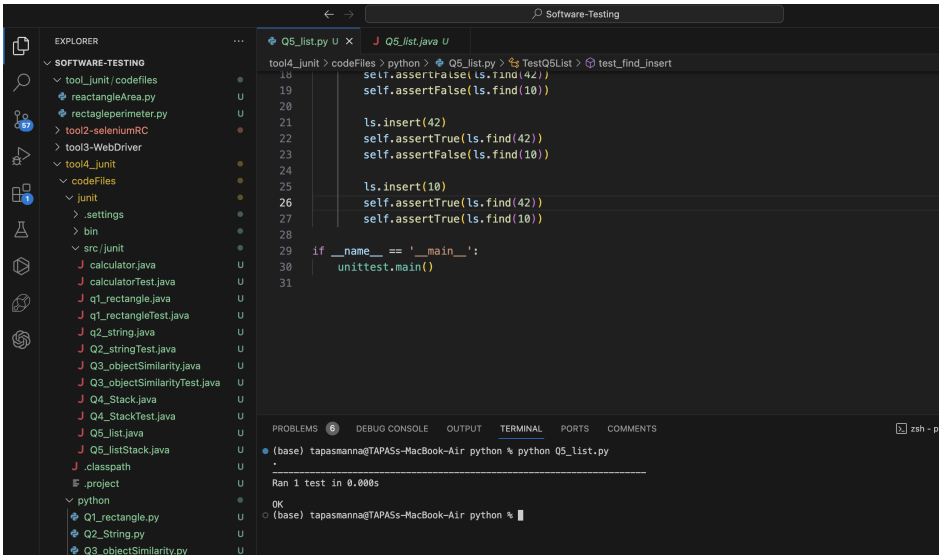

Figure 10: output