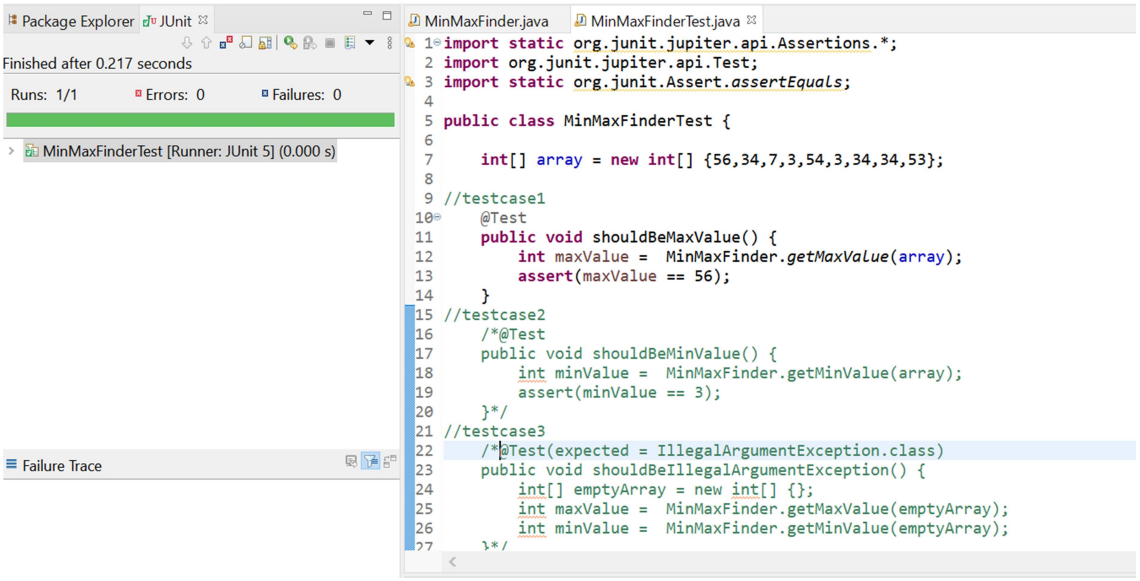
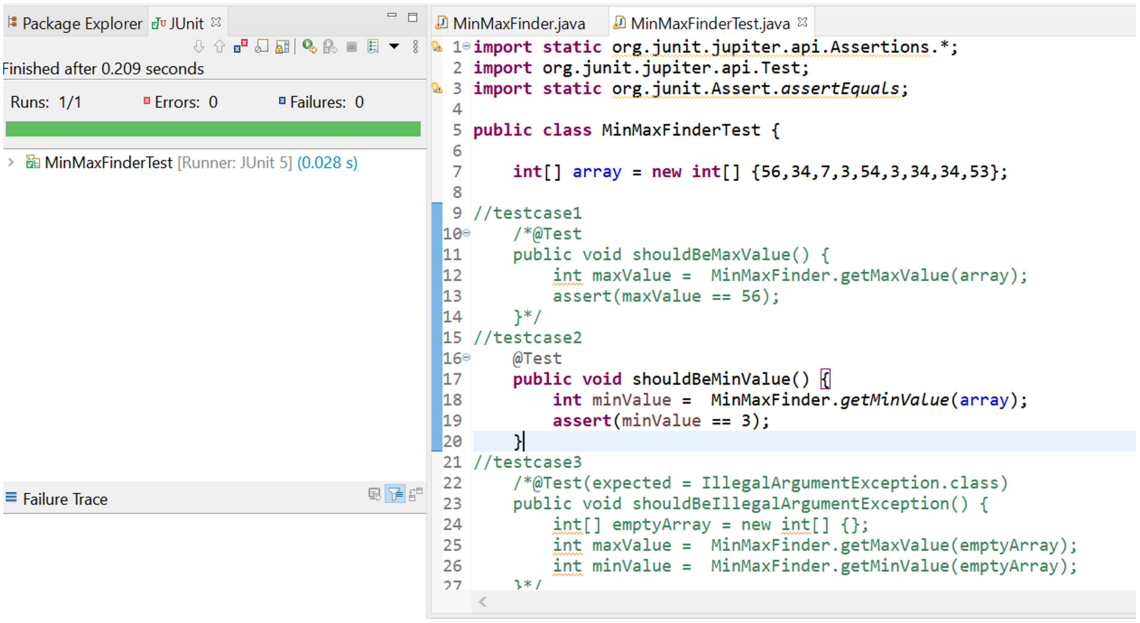
JUnit

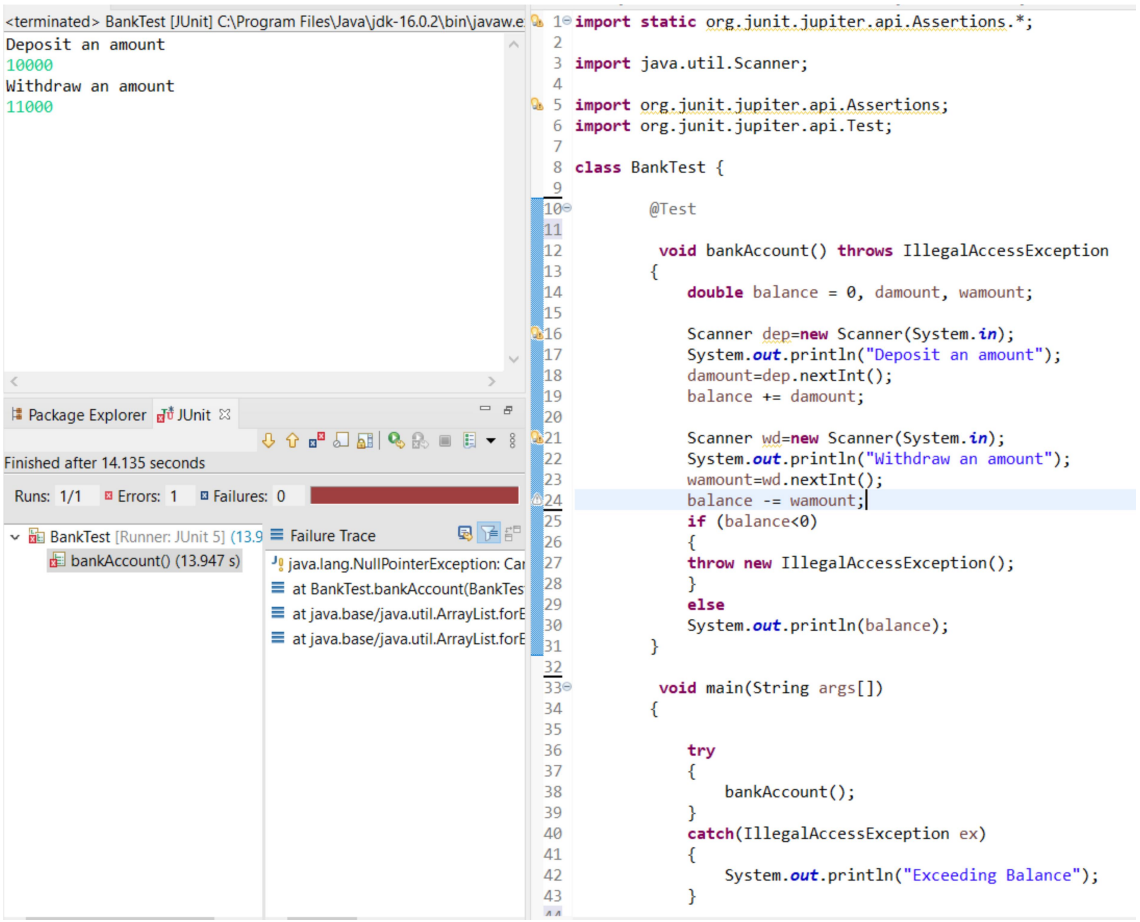
1

```java
import java.util.Scanner;

public class MinMaxFinder {

        public static int getMaxValue(int[] array) {
               int value = Integer.MIN_VALUE;

               if (array.length <= 0) {
                   throw new IllegalArgumentException("Array is empty.");
               }

               for (int i = 0; i < array.length; i++) {
                   if (array[i] > value) {
                       value = array[i];
                   }
               }
               return value;
           }

           public static int getMinValue(int[] array) {
               int value = Integer.MAX_VALUE;

               if (array.length <= 0) {
                   throw new IllegalArgumentException("Array is empty.");
               }

               for (int i=0; i < array.length; i++) {
                   if (array[i] < value) {
                       value = array[i];
                   }
               }
               return value;
           }
       }
```

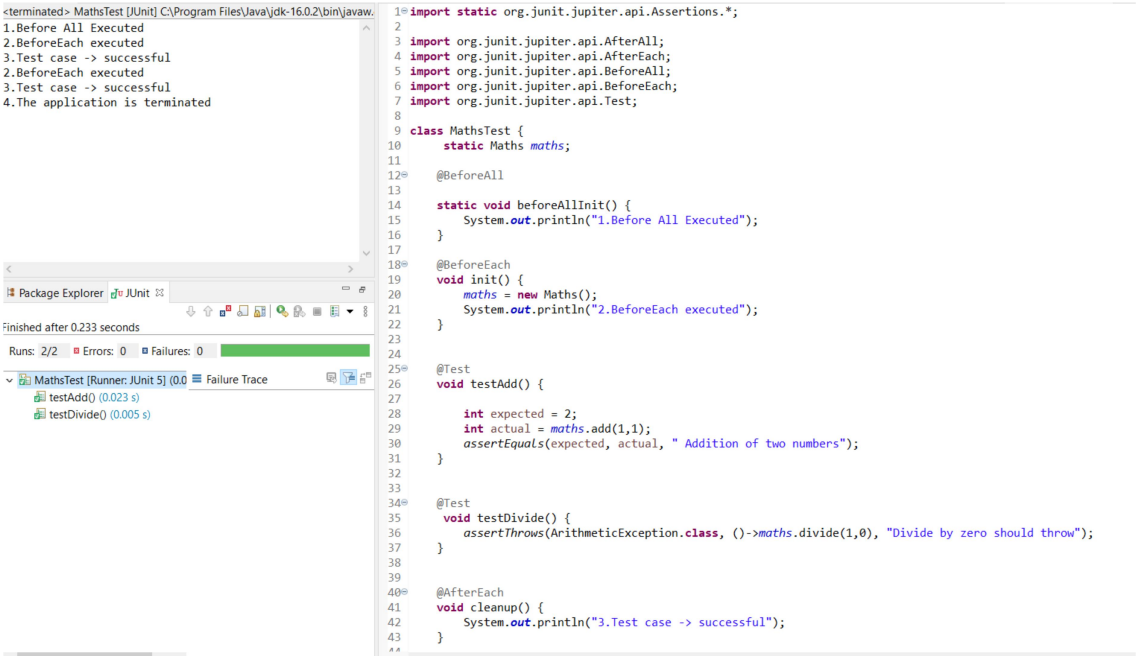



Package Explorer
JUnit
Finished after 0.209 seconds
Runs: 1/1
Errors: 0
Failures: 0
MinMaxFinderTest [Runner: JUnit 5] (0.028 s)
Failure Trace
MinMaxFinder.java
MinMaxFinderTest.java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import static org.junit.Assert.assertEquals;
public class MinMaxFinderTest {
int[] array = new int[] {56,34,7,3,54,3,34,34,53};
//testcase1
/*@Test
public void shouldBeMaxValue() {
int maxValue = MinMaxFinder.getMaxValue(array);
assert(maxValue == 56);
}*/
//testcase2
@Test
public void shouldBeMinValue() {
int minValue = MinMaxFinder.getMinValue(array);
assert(minValue == 3);
}
//testcase3
/*@Test(expected = IllegalArgumentException.class)
public void shouldBeIllegalArgumentException() {
int[] emptyArray = new int[] {};
int maxValue = MinMaxFinder.getMaxValue(emptyArray);
int minValue = MinMaxFinder.getMinValue(emptyArray);

3


<terminated> BankTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.e
Deposit an amount
10000
Withdraw an amount
11000
Package Explorer
JUnit
Finished after 14.135 seconds
Runs: 1/1
Errors: 1
Failures: 0
BankTest [Runner: JUnit 5] (13.9
bankAccount() (13.947 s)
Failure Trace
java.lang.NullPointerException: Cat
at BankTest.bankAccount(BankTes
at java.base/java.util.ArrayList.forE
at java.base/java.util.ArrayList.forE
import static org.junit.jupiter.api.Assertions.*;
import java.util.Scanner;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
class BankTest {
@Test
void bankAccount() throws IllegalAccessException
{
double balance = 0, damount, wamount;
Scanner dep=new Scanner(System.in);
System.out.println("Deposit an amount");
damount=dep.nextInt();
balance += damount;
Scanner wd=new Scanner(System.in);
System.out.println("Withdraw an amount");
wamount=wd.nextInt();
balance -= wamount;
if (balance<0)
{
throw new IllegalAccessException();
}
else
System.out.println(balance);
}
void main(String args[])
{
try
{
bankAccount();
}
catch(IllegalAccessException ex)
{
System.out.println("Exceeding Balance");
}

4


<terminated> MathsTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.
1.Before All Executed
2.BeforeEach executed
3.Test case -> successful
2.BeforeEach executed
3.Test case -> successful
4.The application is terminated
Package Explorer
JUnit
Finished after 0.233 seconds
Runs: 2/2
Errors: 0
Failures: 0
MathsTest [Runner: JUnit 5] (0.0
Failure Trace
testAdd() (0.023 s)
testDivide() (0.005 s)
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.AfterAll;
4 import org.junit.jupiter.api.AfterEach;
5 import org.junit.jupiter.api.BeforeAll;
6 import org.junit.jupiter.api.BeforeEach;
7 import org.junit.jupiter.api.Test;
8
9 class MathsTest {
10 static Maths maths;
11
12 @BeforeAll
13
14 static void beforeAllInit() {
15 System.out.println("1.Before All Executed");
16 }
17
18 @BeforeEach
19 void init() {
20 maths = new Maths();
21 System.out.println("2.BeforeEach executed");
22 }
23
24
25 @Test
26 void testAdd() {
27
28 int expected = 2;
29 int actual = maths.add(1,1);
30 assertEquals(expected, actual, " Addition of two numbers");
31 }
32
33
34 @Test
35 void testDivide() {
36 assertThrows(ArithmeticException.class, ()->maths.divide(1,0), "Divide by zero should throw");
37 }
38
39
40 @AfterEach
41 void cleanup() {
42 System.out.println("3.Test case -> successful");
43 }