

JUNIT ASSIGNMENT

Name: Shruti Bhosale

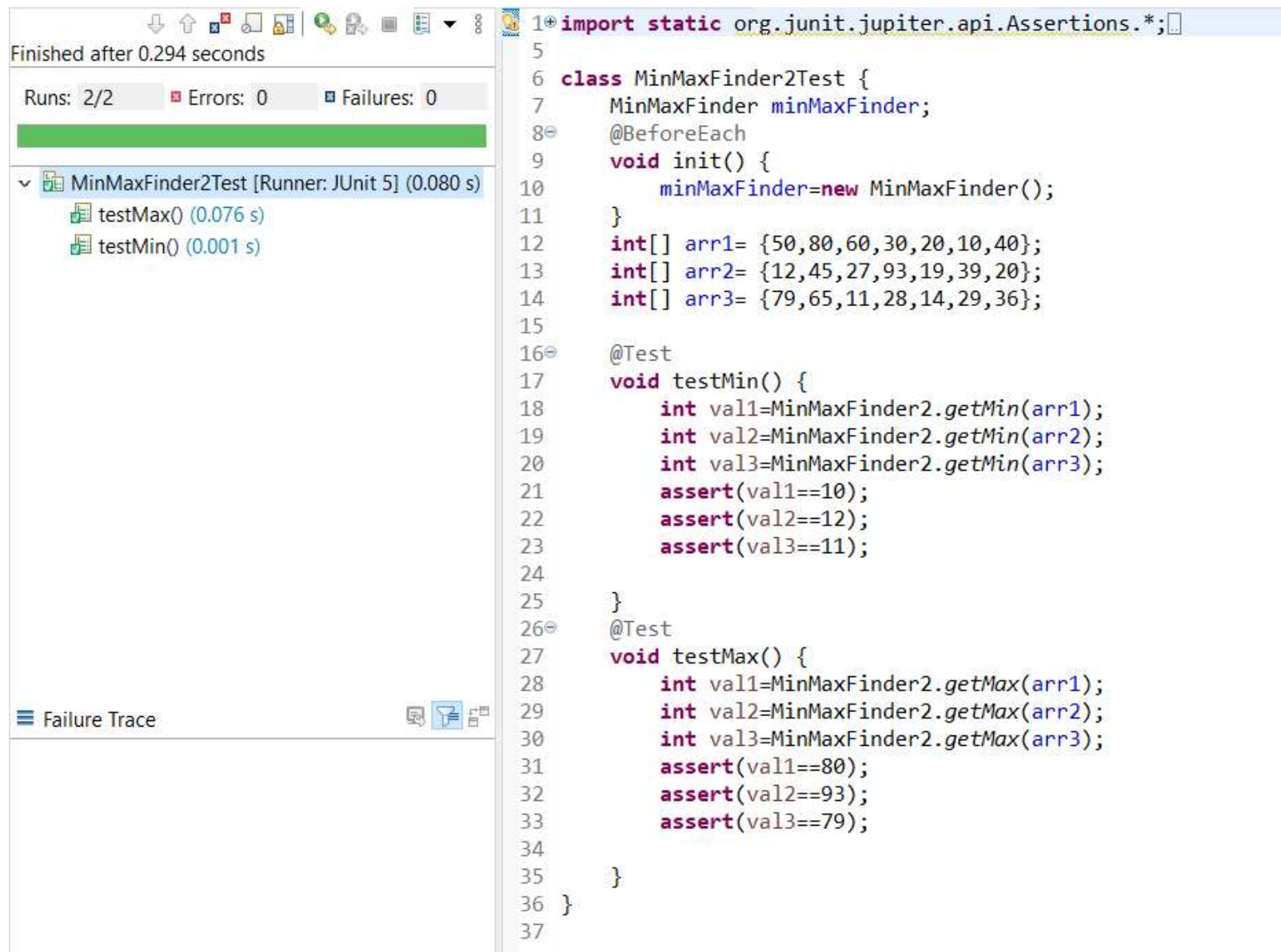
1) MinMaxFinder:

```
1 import java.util.Arrays;
2
3 public class MinMaxFinder{
4     public int[] findMinMax(int[] arr) {
5         Arrays.sort(arr);
6         int min=arr[0];
7         int max=arr[arr.length-1];
8         int[] result= {min,max};
9         return result;
10    }
11 }
12 }
```

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 class MinMaxFinderTest {
4     MinMaxFinder minMaxFinder;
5     @BeforeEach
6     void init() {
7         minMaxFinder=new MinMaxFinder();
8     }
9     int[] arr1= {50,80,60,30,20,10,40};
10    @Test
11    void testforEqual() {
12        int[] expected= {10,80};
13        int[] actual=minMaxFinder.findMinMax(arr1);
14        assertEquals(expected,actual);
15    }
16
17    @Test
18    void testforNotEqual() {
19        int[] expected= {10,50};
20        int[] actual=minMaxFinder.findMinMax(arr1);
21        Assertions.assertNotEquals(expected, actual);
22    }
23
24    @Test
25    void test3() {
26        boolean res=true;
27        int[] expected=new int[] {10,80};
28        int[] actual=minMaxFinder.findMinMax(arr1);
29        if(actual!=null) {
30            res=false;
31        }
32        Assertions.assertFalse(res);
33    }
34 }
35 }
```

2) MinMaxFinder2:

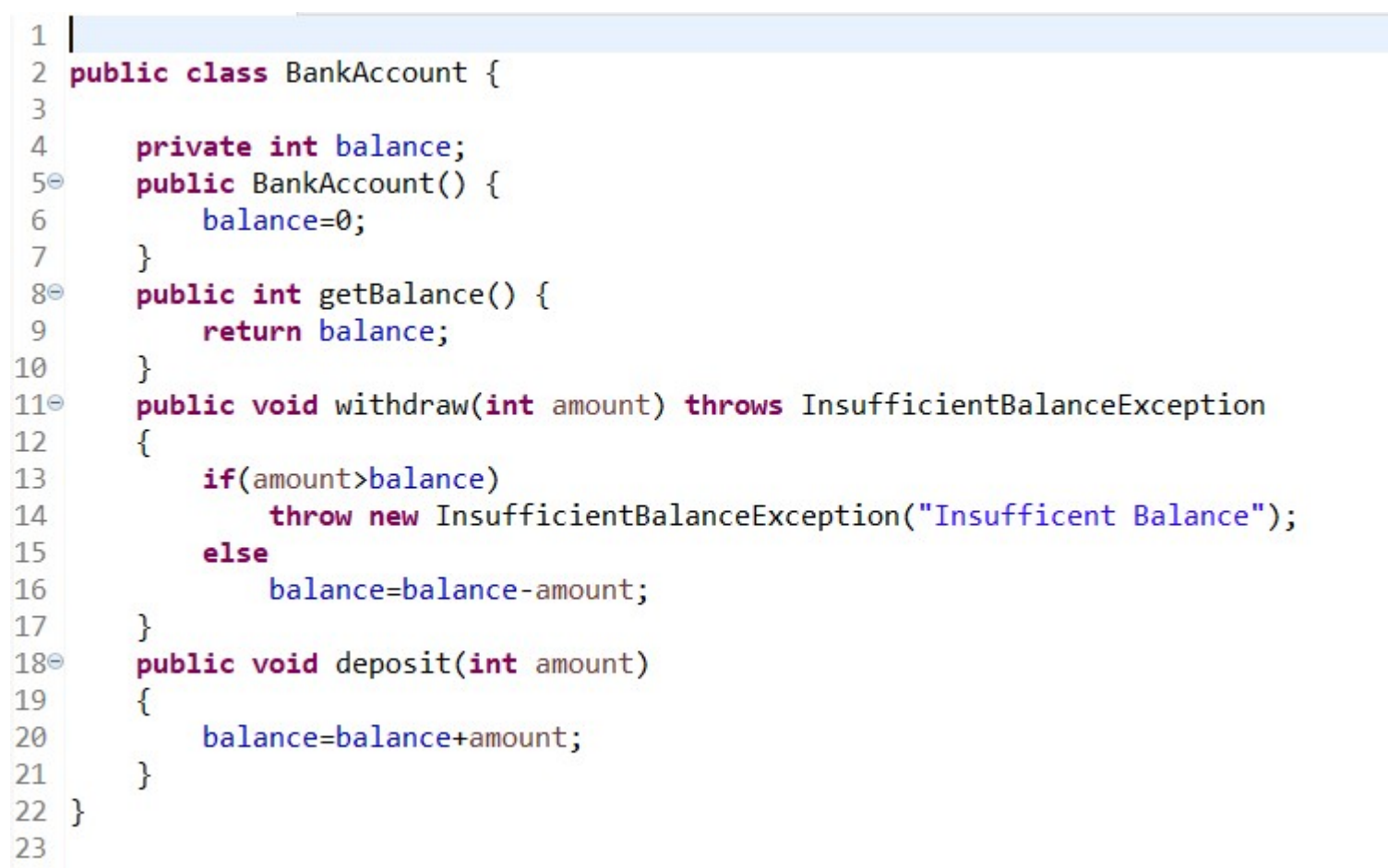
```
1 import java.util.Arrays;
2
3 public class MinMaxFinder2{
4
5     public static int getMin(int[] arr) {
6         Arrays.sort(arr);
7         int min=arr[0];
8         return min;
9     }
10
11     public static int getMax(int[] arr) {
12         Arrays.sort(arr);
13         int max=arr[arr.length-1];
14         return max;
15     }
16 }
17 }
```



The screenshot shows an IDE with two panels. The left panel displays test results for `MinMaxFinder2Test` [Runner: JUnit 5] (0.080 s). It shows two tests: `testMax()` (0.076 s) and `testMin()` (0.001 s). The right panel shows the source code for `MinMaxFinder2Test`. The code imports `org.junit.jupiter.api.Assertions.*` and defines a class `MinMaxFinder2Test` with a `@BeforeEach` `init()` method that initializes `minMaxFinder`. It also has two `@Test` methods: `testMin()` and `testMax()`, each with three assertions.

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3
4
5
6 class MinMaxFinder2Test {
7     MinMaxFinder minMaxFinder;
8     @BeforeEach
9     void init() {
10         minMaxFinder=new MinMaxFinder();
11     }
12     int[] arr1= {50,80,60,30,20,10,40};
13     int[] arr2= {12,45,27,93,19,39,20};
14     int[] arr3= {79,65,11,28,14,29,36};
15
16     @Test
17     void testMin() {
18         int val1=MinMaxFinder2.getMin(arr1);
19         int val2=MinMaxFinder2.getMin(arr2);
20         int val3=MinMaxFinder2.getMin(arr3);
21         assert(val1==10);
22         assert(val2==12);
23         assert(val3==11);
24     }
25
26     @Test
27     void testMax() {
28         int val1=MinMaxFinder2.getMax(arr1);
29         int val2=MinMaxFinder2.getMax(arr2);
30         int val3=MinMaxFinder2.getMax(arr3);
31         assert(val1==80);
32         assert(val2==93);
33         assert(val3==79);
34     }
35 }
36
37
```

3)Bank Account:



The screenshot shows an IDE with a single panel displaying the source code for `BankAccount`. The code defines a class `BankAccount` with a private `balance` attribute. It has a constructor `BankAccount()` that initializes `balance` to 0. It also has two public methods: `getBalance()` which returns the current balance, and `withdraw(int amount)` which throws an `InsufficientBalanceException` if the amount is greater than the balance, and `deposit(int amount)` which adds the amount to the balance.

```
1
2 public class BankAccount {
3
4     private int balance;
5     public BankAccount() {
6         balance=0;
7     }
8     public int getBalance() {
9         return balance;
10    }
11    public void withdraw(int amount) throws InsufficientBalanceException
12    {
13        if(amount>balance)
14            throw new InsufficientBalanceException("Insufficent Balance");
15        else
16            balance=balance-amount;
17    }
18    public void deposit(int amount)
19    {
20        balance=balance+amount;
21    }
22 }
23
```



```

1 import static org.junit.jupiter.api.Assertions.*;
2
3 public class BankTest {
4     BankAccount customer;
5
6     @BeforeClass
7     public static void test1() {
8         System.out.println("Test Data setup for all Test");
9         Assume.assumeTrue(false);
10    }
11    @AfterClass
12    public static void test2() {
13        System.out.println("Test data cleaned");
14    }
15    @Before
16    public void InitilizeCustomer() {
17        System.out.println("@ Before Get Called");
18        customer=new BankAccount();
19    }
20 }
21
22

```

```

1
2 public class InsufficientBalanceException extends Exception {
3
4     public InsufficientBalanceException() {
5         super();
6         // TODO Auto-generated constructor stub
7     }
8
9     public InsufficientBalanceException(String message, Throwable cause, boolean enableSuppression,
10        boolean writableStackTrace) {
11         super(message, cause, enableSuppression, writableStackTrace);
12         // TODO Auto-generated constructor stub
13     }
14
15     public InsufficientBalanceException(String message, Throwable cause) {
16         super(message, cause);
17         // TODO Auto-generated constructor stub
18     }
19
20     public InsufficientBalanceException(String message) {
21         super(message);
22         // TODO Auto-generated constructor stub
23     }
24
25     public InsufficientBalanceException(Throwable cause) {
26         super(cause);
27         // TODO Auto-generated constructor stub
28     }
29 }
30
31

```

Finished after 0.194 seconds

Runs: 2/2 Errors: 0 Failures: 0

- ExceptionTest [Runner: JUnit 5] (0.027 s)
 - ValidateWithdrawnExceptionTest (0.013 s)
 - ValidateWithdrawExceptionMessageTest (0.013 s)

```

1 import static org.junit.jupiter.api.Assertions.*;
2
3 public class ExceptionTest {
4
5     @Test(expected=InsufficientBalanceException.class)
6     public void ValidateWithdrawnExceptionTest() throws InsufficientBalanceException
7     {
8
9         BankAccount customer = new BankAccount();
10        customer.withdraw(1000);
11    }
12    @SuppressWarnings("deprecation")
13    @Rule
14    public ExpectedException thrown=ExpectedException.none();
15
16    @Test
17    public void ValidateWithdrawExceptionMessageTest() throws InsufficientBalanceException
18    {
19        thrown.expect(InsufficientBalanceException.class);
20        thrown.expectMessage("Insufficient Balance");
21        BankAccount customer = new BankAccount();
22        customer.withdraw(1000);
23    }
24 }
25

```

4)Life Cycle:

```
1
2 public class Math {
3     public int add (int a , int b) {
4         return a+b;
5     }
6
7     public Object divide(int a, int b) {
8
9         return a/b;
10    }
11 }
```

```
1 import static org.junit.jupiter.api.Assertions.*;
2 import org.junit.jupiter.api.AfterAll;
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeAll;
5 import org.junit.jupiter.api.BeforeEach;
6 import org.junit.jupiter.api.Test;
7 import org.junit.jupiter.api.Test;
8 class MathTest {
9     static Math math;
10    @BeforeAll
11    static void beforeAllInit() {
12        System.out.println("1.BeforeAll Executed");
13    }
14    @BeforeEach
15    void init() {
16        math = new Math();
17        System.out.println("2.BeforeEach Executed");
18    }
19    @Test
20    void testAdd() {
21
22        int expected = 2;
23        int actual = math.add(1,1);
24        assertEquals(expected, actual, " Addition of two numbers");
25    }
26    @Test
27    void testDivide() {
28        assertThrows(ArithmeticException.class, ()->math.divide(1,0), "Divide by zero should throw");
29    }
30    @AfterEach
31    void cleanup() {
32        System.out.println("3.Test case -> successful");
33    }
34    @AfterAll
35    static void afterAllfunc() {
36        System.out.println("4.The application is terminated");
37    }
38 }
```

Output:

```
Console Problems Debug Shell
<terminated> MathTest (1) [JUnit] C:\Program Files\Java\jdk-
1.Before All Executed
2.BeforeEach executed
3.Test case -> successful
2.BeforeEach executed
3.Test case -> successful
4.The application is terminated

Finished after 0.366 seconds
Runs: 2/2 Errors: 0 Failures: 0
MathTest [Runner: JUnit 5] (0.003 s)
  testAdd() (0.000 s)
  testDivide() (0.003 s)
```