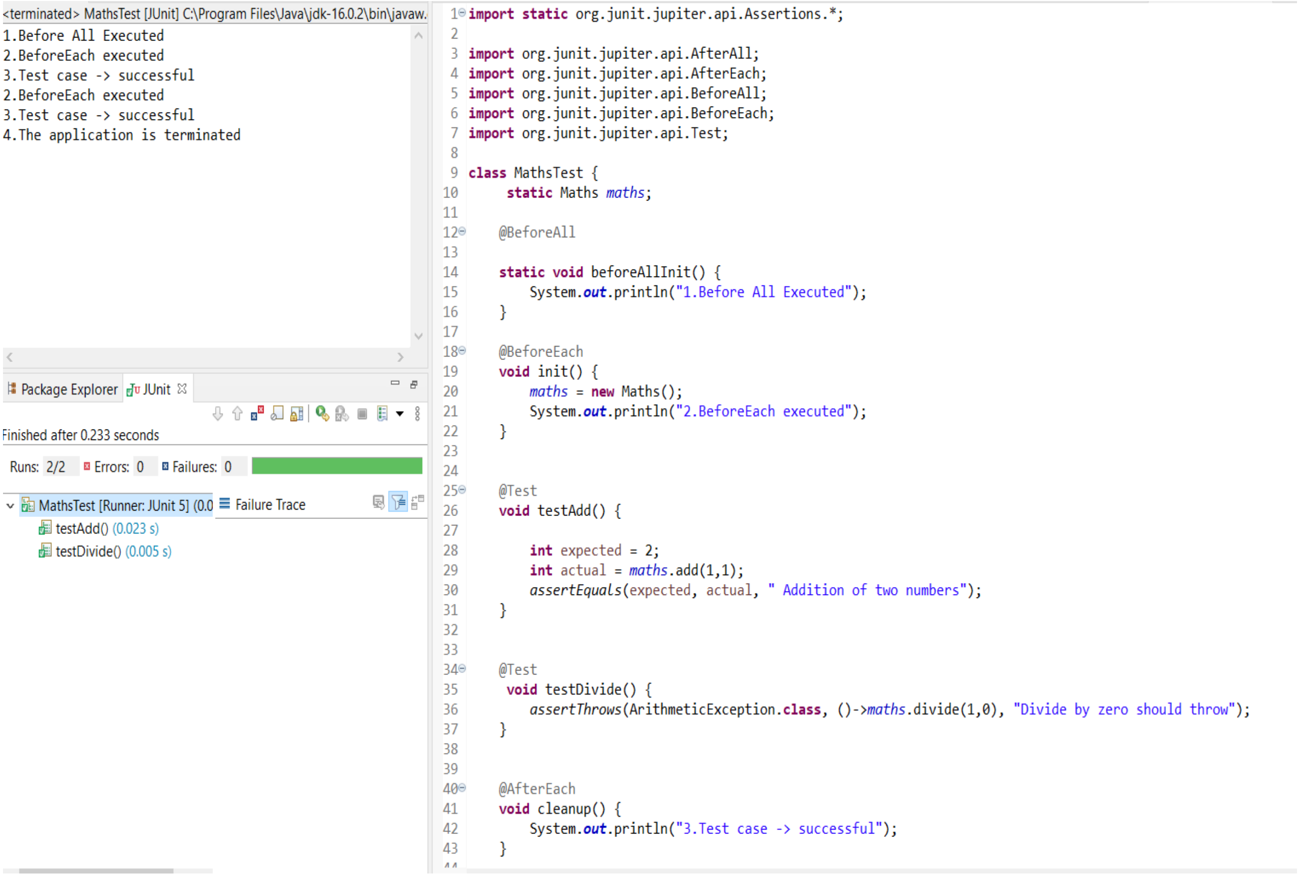
Junit Assignment

Q4



```
<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
```

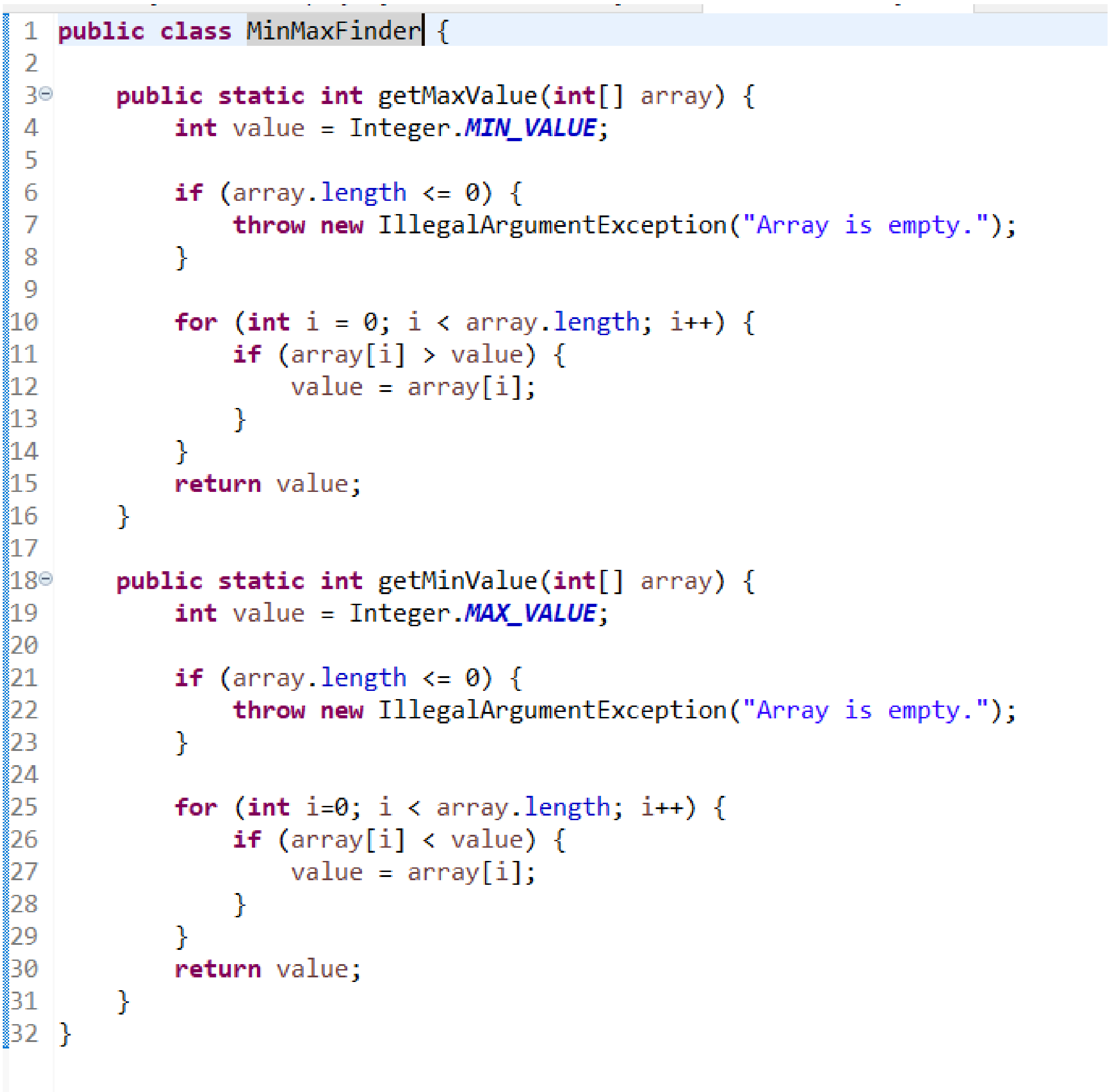
Finished after 0.233 seconds

Runs: 2/2 Errors: 0 Failures: 0

MathsTest [Runner: JUnit 5] (0.0
testAdd() (0.023 s)
testDivide() (0.005 s)

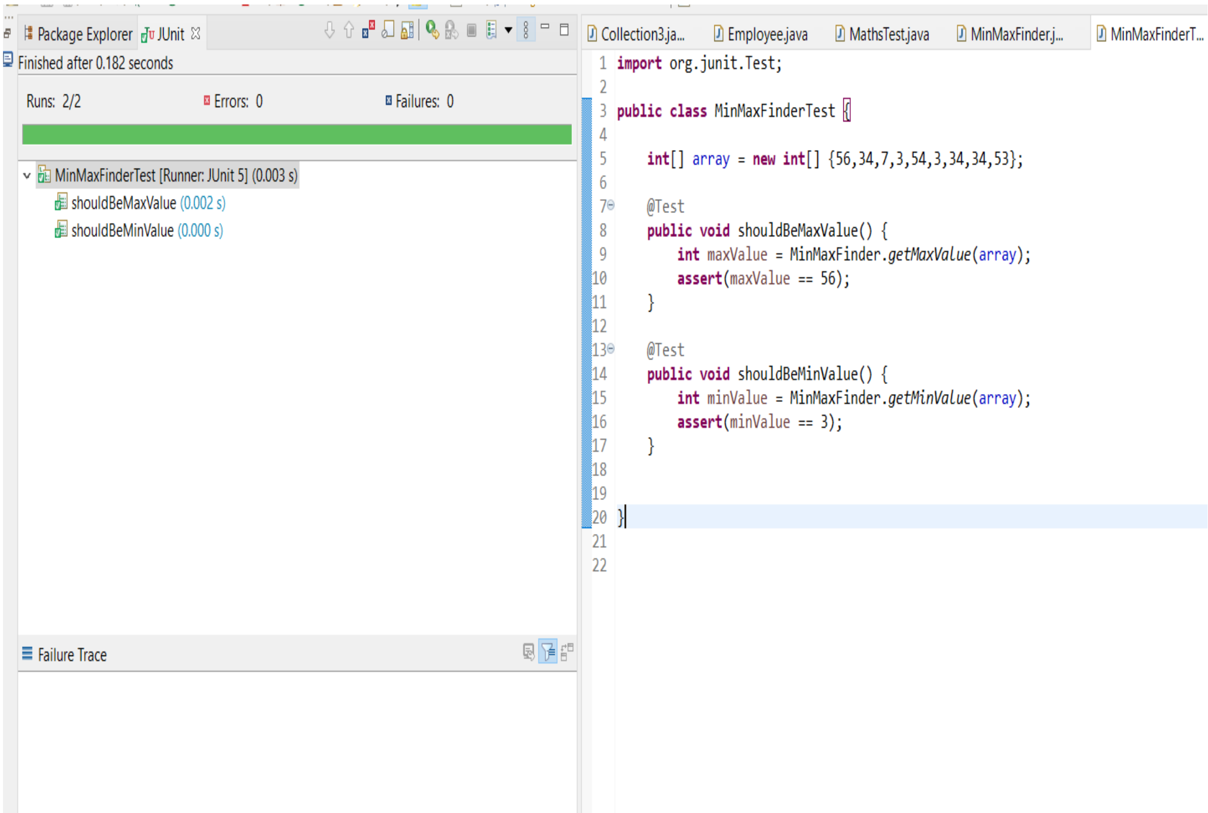
```java
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class MathsTest {
     static Maths maths;

    @BeforeAll

    static void beforeAllInit() {
        System.out.println("1.Before All Executed");
    }

    @BeforeEach
    void init() {
        maths = new Maths();
        System.out.println("2.BeforeEach executed");
    }


    @Test
    void testAdd() {

        int expected = 2;
        int actual = maths.add(1,1);
        assertEquals(expected, actual, " Addition of two numbers");
    }


    @Test
     void testDivide() {
        assertThrows(ArithmeticException.class, ()->maths.divide(1,0), "Divide by zero should throw");
    }


    @AfterEach
    void cleanup() {
        System.out.println("3.Test case -> successful");
    }
```

Q1



```java
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;
    }
}
```

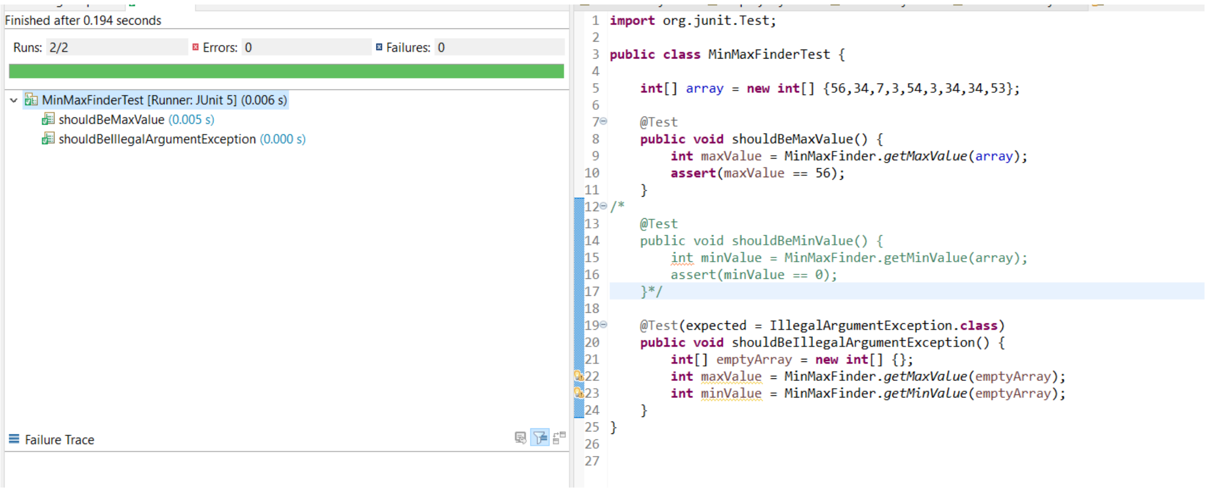
Test Case 1



```
Finished after 0.182 seconds

Runs:  2/2        Errors:  0        Failures:  0

MinMaxFinderTest [Runner: JUnit 5] (0.003 s)
    shouldBeMaxValue (0.002 s)
    shouldBeMinValue (0.000 s)
```

```java
import org.junit.Test;

public class MinMaxFinderTest {

    int[] array = new int[] {56,34,7,3,54,3,34,34,53};

    @Test
    public void shouldBeMaxValue() {
        int maxValue = MinMaxFinder.getMaxValue(array);
        assert(maxValue == 56);
    }

    @Test
    public void shouldBeMinValue() {
        int minValue = MinMaxFinder.getMinValue(array);
        assert(minValue == 3);
    }


}


```

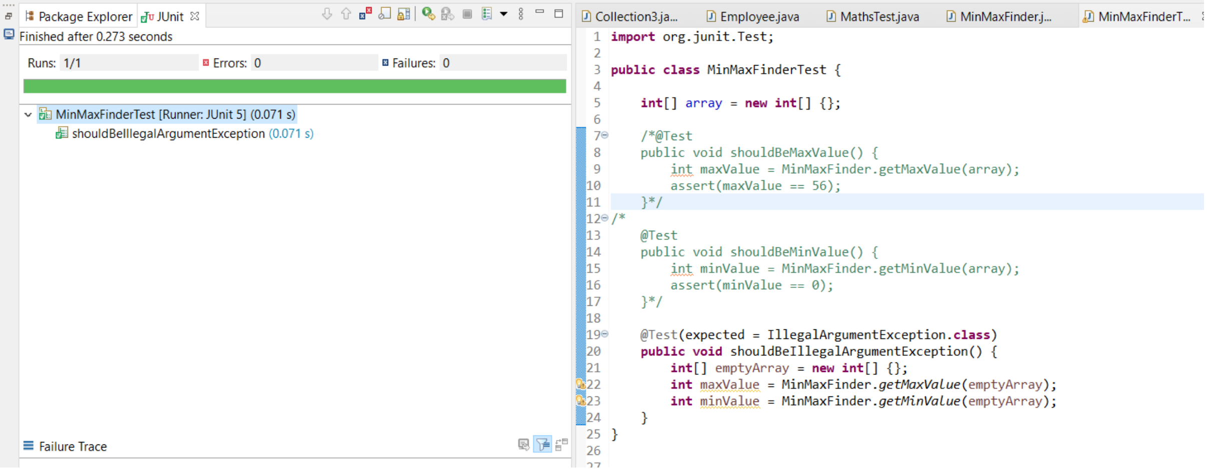
Test Case 2



```
Finished after 0.194 seconds

Runs:  2/2        Errors:  0        Failures:  0

MinMaxFinderTest [Runner: JUnit 5] (0.006 s)
    shouldBeMaxValue (0.005 s)
    shouldBeIllegalArgumentException (0.000 s)
```

```java
import org.junit.Test;

public class MinMaxFinderTest {

    int[] array = new int[] {56,34,7,3,54,3,34,34,53};

    @Test
    public void shouldBeMaxValue() {
        int maxValue = MinMaxFinder.getMaxValue(array);
        assert(maxValue == 56);
    }
/*
    @Test
    public void shouldBeMinValue() {
        int minValue = MinMaxFinder.getMinValue(array);
        assert(minValue == 0);
    }*/

    @Test(expected = IllegalArgumentException.class)
    public void shouldBeIllegalArgumentException() {
        int[] emptyArray = new int[] {};
        int maxValue = MinMaxFinder.getMaxValue(emptyArray);
        int minValue = MinMaxFinder.getMinValue(emptyArray);
    }
}


```

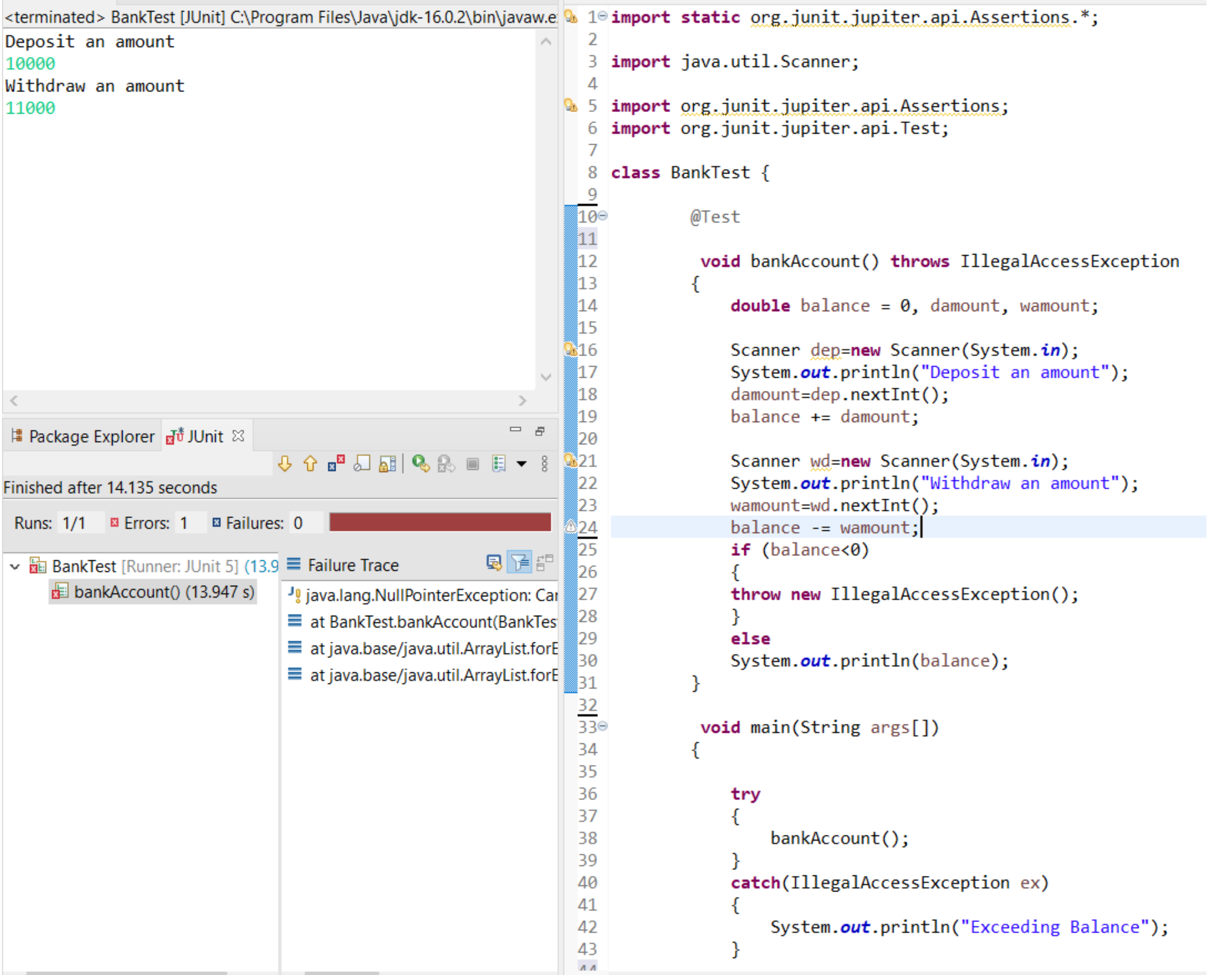
Test Case 3



```
Finished after 0.273 seconds
Runs: 1/1    Errors: 0    Failures: 0
MinMaxFinderTest [Runner: JUnit 5] (0.071 s)
    shouldBeIllegalArgumentException (0.071 s)
```

```java
import org.junit.Test;

public class MinMaxFinderTest {

    int[] array = new int[] {};

    /*@Test
    public void shouldBeMaxValue() {
        int maxValue = MinMaxFinder.getMaxValue(array);
        assert(maxValue == 56);
    }*/
/*
    @Test
    public void shouldBeMinValue() {
        int minValue = MinMaxFinder.getMinValue(array);
        assert(minValue == 0);
    }*/

    @Test(expected = IllegalArgumentException.class)
    public void shouldBeIllegalArgumentException() {
        int[] emptyArray = new int[] {};
        int maxValue = MinMaxFinder.getMaxValue(emptyArray);
        int minValue = MinMaxFinder.getMinValue(emptyArray);
    }
}
```

Q3



```
<terminated> BankTest [JUnit] C:\Program Files\Java\jdk-16.0.2\bin\javaw.e
Deposit an amount
10000
Withdraw an amount
11000
```

```
Finished after 14.135 seconds
Runs: 1/1    Errors: 1    Failures: 0
BankTest [Runner: JUnit 5] (13.9
    bankAccount() (13.947 s)
java.lang.NullPointerException: Car
at BankTest.bankAccount(BankTes
at java.base/java.util.ArrayList.forE
at java.base/java.util.ArrayList.forE
```

```java
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");
            }
```