CS6474: Software Testing Laboratory
(Spring 2023)

Bishwajit Prasad Gond
222CS3113

Master of Technology
`222cs3113@nitrkl.ac.in`

Department of Computer Science & Engineering
NIT, Rourkela

March 18, 2023

Contents

1 JCute 1

1.1 Checking prime numbers, odd numbers, or even numbers. 1

1.2 Checking odd numbers, or even numbers. 3

1.3 Calculator based on floating-point 32-bit numbers. 5

1.4 Reverse an array. 7

1.5 Decimal to hexadecimal. 9

1.6 Grade students based on marks. 11

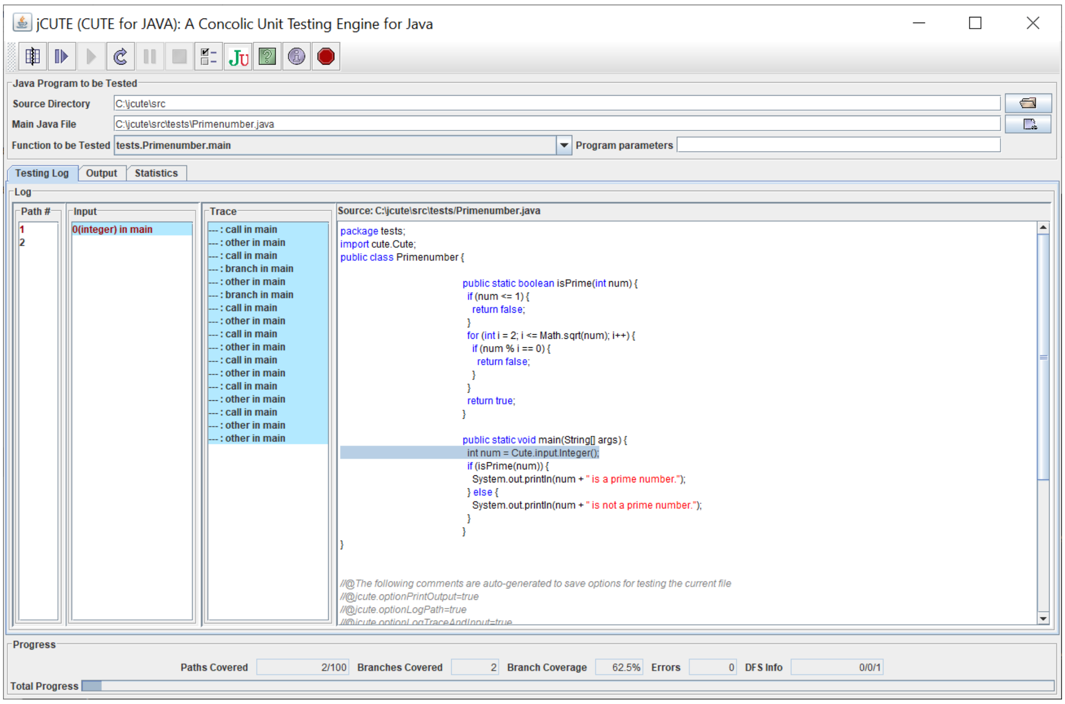
1 JCute

The Java Concolic Unit Testing Engine (jCUTE) automatically generates unit tests for Java programs. Concolic execution combines randomized concrete execution with symbolic execution and automatic constraint solving. Symbolic execution allows jCUTE to discern inputs that lead down different execution paths; randomized concrete execution helps it overcome limitations of the constraint solver, like the inability to analyze system calls or solve general systems of non-linear integer equations. Through this combination, jCUTE is able to generate test cases that execute many different execution paths in real Java programs.

jCUTE supports multi-threaded programs. It can discover race conditions and deadlocks through systematic schedule exploration.

Write code for scenarios given below using Java and apply concolic testing to improve branch coverage.

1.1 Checking prime numbers, odd numbers, or even numbers.


jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Java Program to be Tested
Testing Log
Output
Statistics
Log
Progress

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Primenumber.java

Function to be Tested: tests.Primenumber.main

Program parameters:

Path #: 1, 2

Input: 0(integer) in main

```
--- : call in main
--- : other in main
--- : call in main
--- : branch in main
--- : other in main
--- : branch in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : other in main
```

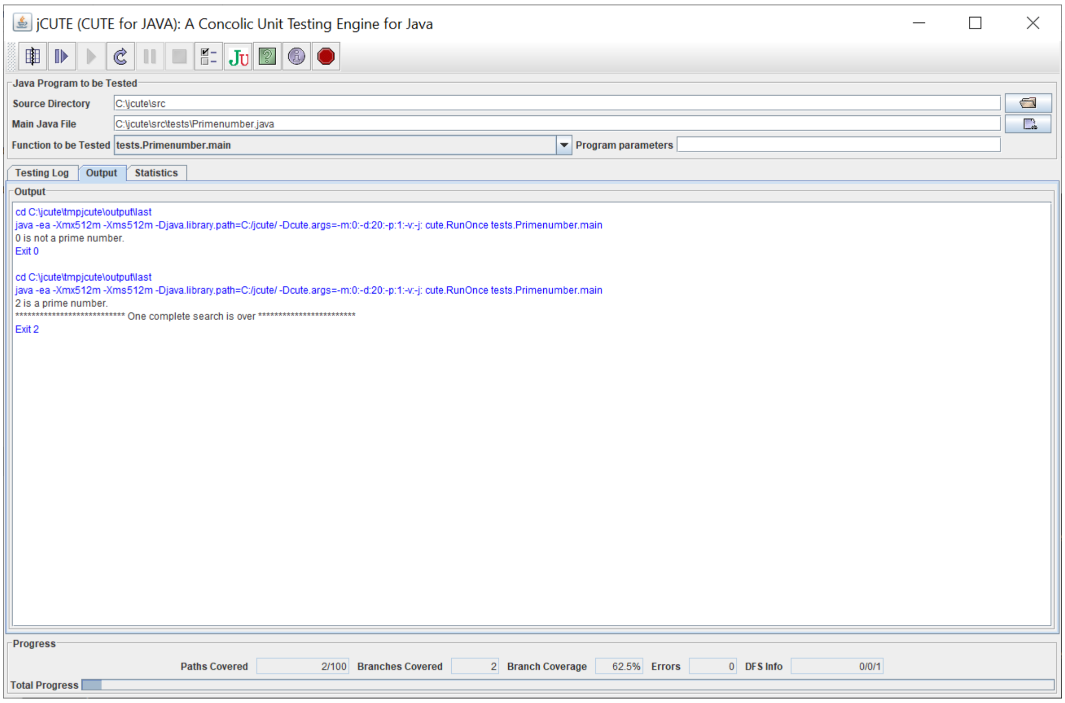
Source: C:\jcute\src\tests/Primenumber.java

```
package tests;
import cute.Cute;
public class Primenumber {

                              public static boolean isPrime(int num) {
                               if (num <= 1) {
                                return false;
                               }
                               for (int i = 2; i <= Math.sqrt(num); i++) {
                                if (num % i == 0) {
                                 return false;
                                }
                               }
                               return true;
                              }

                              public static void main(String[] args) {
                               int num = Cute.input.Integer();
                               if (isPrime(num)) {
                                System.out.println(num + " is a prime number.");
                               } else {
                                System.out.println(num + " is not a prime number.");
                               }
                              }
}


//@The following comments are auto-generated to save options for testing the current file
//@jcute.optionPrintOutput=true
//@jcute.optionLogPath=true
//@jcute.optionLogTraceAndInput=true
```

Paths Covered: 2/100 Branches Covered: 2 Branch Coverage: 62.5% Errors: 0 DFS Info: 0/0/1

Total Progress

Figure 1: Testing log


jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Java Program to be Tested
Testing Log
Output
Statistics
Output
Progress

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Primenumber.java

Function to be Tested: tests.Primenumber.main

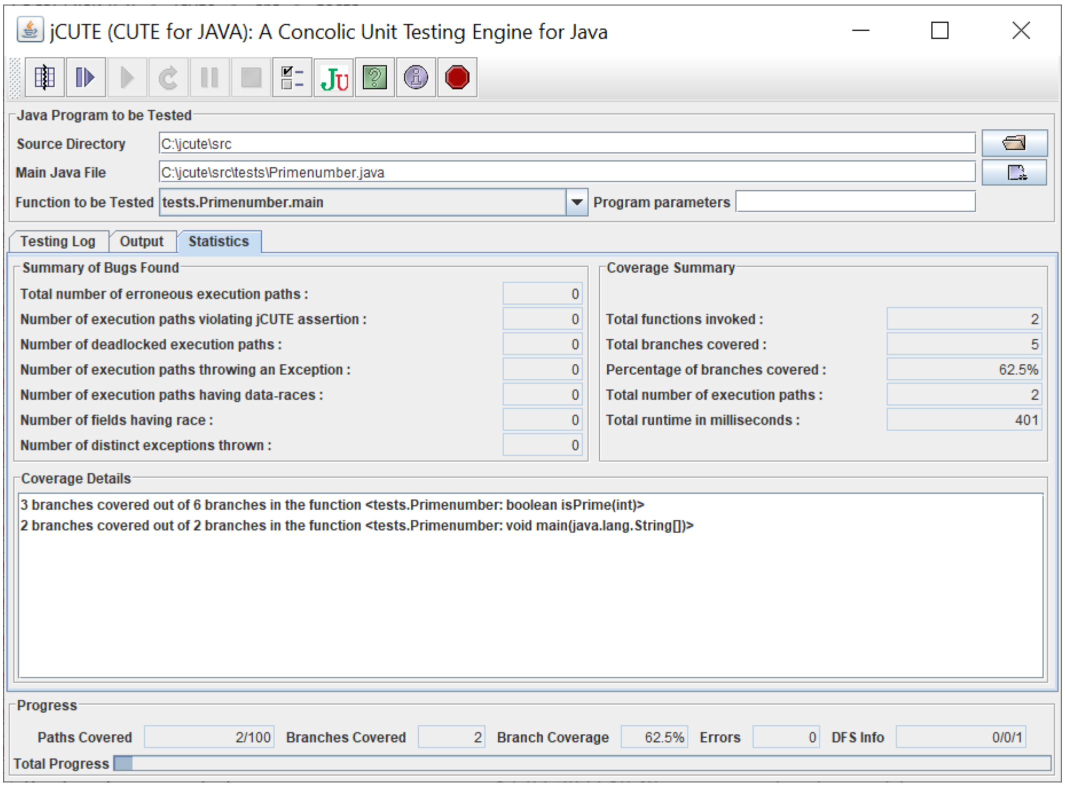
Program parameters:

```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:20:-p:1:-v:-j: cute.RunOnce tests.Primenumber.main
0 is not a prime number.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:20:-p:1:-v:-j: cute.RunOnce tests.Primenumber.main
2 is a prime number.
************************* One complete search is over ***********************
Exit 2
```

Paths Covered: 2/100 Branches Covered: 2 Branch Coverage: 62.5% Errors: 0 DFS Info: 0/0/1

Total Progress

Figure 2: Output



Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Primenumber.java

Function to be Tested: tests.Primenumber.main

Program parameters:

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Total functions invoked :	2
Total branches covered :	5
Percentage of branches covered :	62.5%
Total number of execution paths :	2
Total runtime in milliseconds :	401

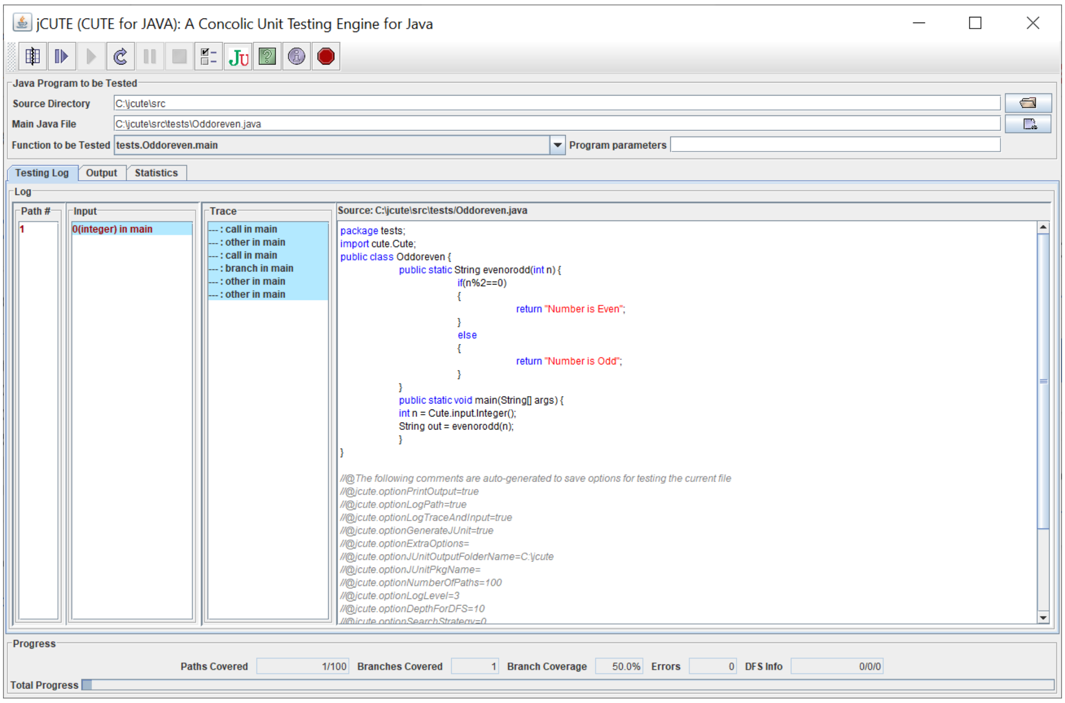
```
3 branches covered out of 6 branches in the function <tests.Primenumber: boolean isPrime(int)>
2 branches covered out of 2 branches in the function <tests.Primenumber: void main(java.lang.String[])>
```

Paths Covered: 2/100 Branches Covered: 2 Branch Coverage: 62.5% Errors: 0 DFS Info: 0/0/1

Total Progress

Figure 3: Statistics Screenshot

1.2 Checking odd numbers, or even numbers.


jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Testing Log
Output
Statistics

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Oddoreven.java

Function to be Tested: tests.Oddoreven.main

Program parameters:

Log

Path #	Input	Trace
1	0(integer) in main	--- : call in main --- : other in main --- : call in main --- : branch in main --- : other in main --- : other in main

Source: C:\jcute\src\tests/Oddoreven.java

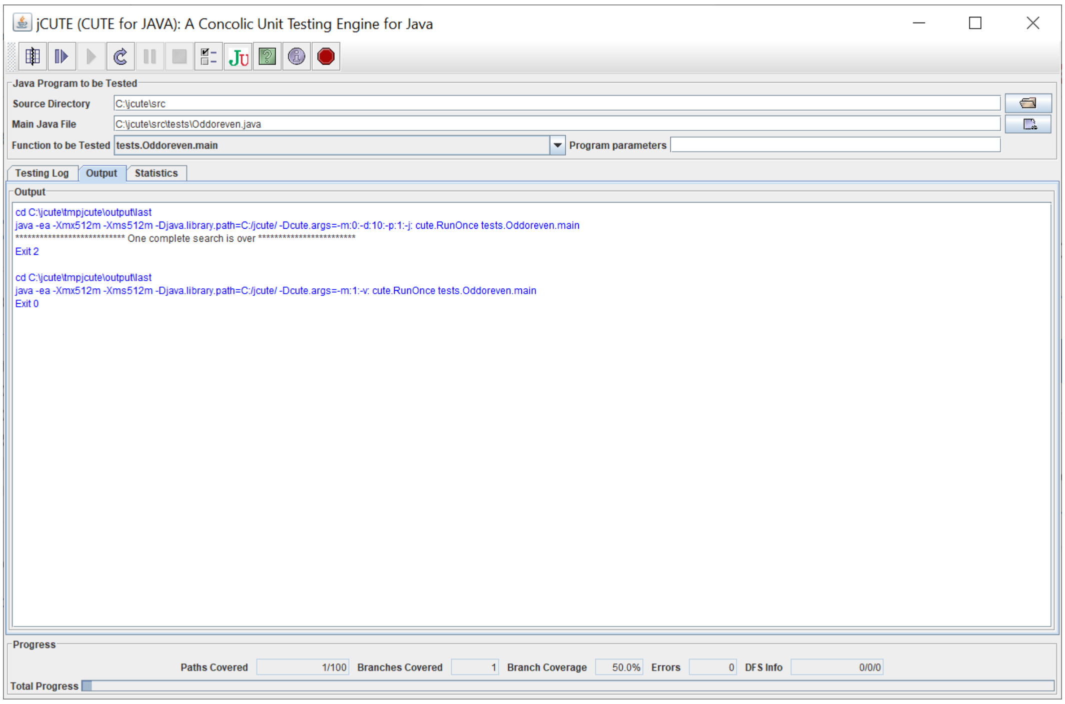
```
package tests;
import cute.Cute;
public class Oddoreven {
        public static String evenorodd(int n) {
                if(n%2==0)
                {
                        return "Number is Even";
                }
                else
                {
                        return "Number is Odd";
                }
        }
        public static void main(String[] args) {
        int n = Cute.input.Integer();
        String out = evenorodd(n);
        }
}

//@The following comments are auto-generated to save options for testing the current file
//@jcute.optionPrintOutput=true
//@jcute.optionLogPath=true
//@jcute.optionLogTraceAndInput=true
//@jcute.optionGenerateJUnit=true
//@jcute.optionExtraOptions=
//@jcute.optionJUnitOutputFolderName=C:\jcute
//@jcute.optionJUnitPkgName=
//@jcute.optionNumberOfPaths=100
//@jcute.optionLogLevel=3
//@jcute.optionDepthForDFS=10
//@jcute.optionSearchStrategy=0
```

Progress

Paths Covered: 1/100 Branches Covered: 1 Branch Coverage: 50.0% Errors: 0 DFS Info: 0/0/0

Total Progress

Figure 4: Testing log


jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Testing Log
Output
Statistics

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Oddoreven.java

Function to be Tested: tests.Oddoreven.main

Program parameters:

Output

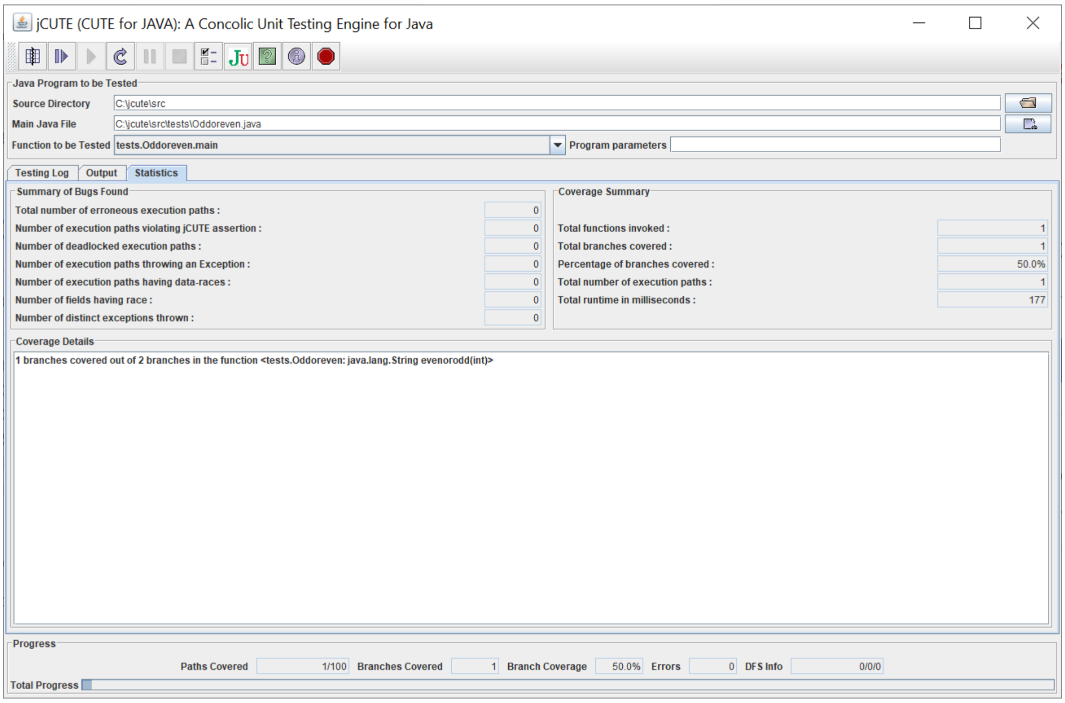
```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:10:-p:1:-j: cute.RunOnce tests.Oddoreven.main
*************************** One complete search is over ************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.Oddoreven.main
Exit 0
```

Progress

Paths Covered: 1/100 Branches Covered: 1 Branch Coverage: 50.0% Errors: 0 DFS Info: 0/0/0

Total Progress

Figure 5: Output



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Testing Log
Output
Statistics

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\Oddoreven.java

Function to be Tested: tests.Oddoreven.main

Program parameters:

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	1
Total branches covered :	1
Percentage of branches covered :	50.0%
Total number of execution paths :	1
Total runtime in milliseconds :	177

Coverage Details

```
1 branches covered out of 2 branches in the function <tests.Oddoreven: java.lang.String evenorodd(int)>
```

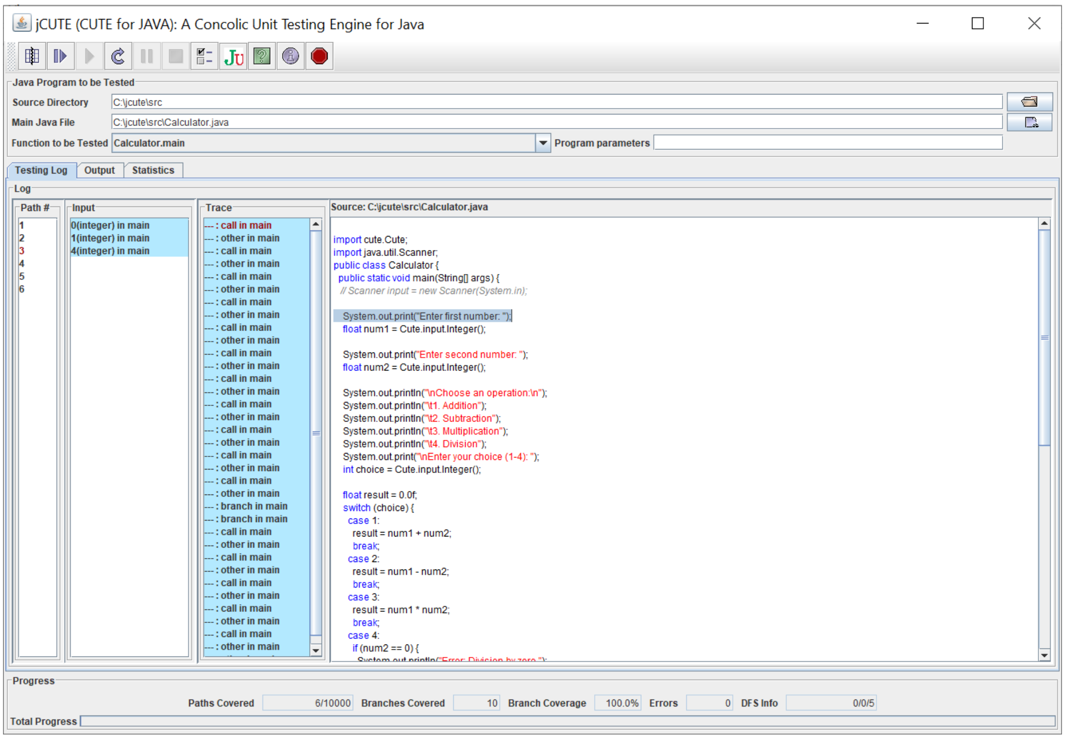
Progress

Paths Covered: 1/100 Branches Covered: 1 Branch Coverage: 50.0% Errors: 0 DFS Info: 0/0/0

Total Progress

Figure 6: Statistics Screenshot

1.3 Calculator based on floating-point 32-bit numbers.



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Java Program to be Tested
Testing Log
Output
Statistics
Log
Path #
Input
Trace
Progress

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\Calculator.java

Function to be Tested: Calculator.main

Program parameters:

Path #:

```
1
2
3
4
5
6
```

Input:

```
0(integer) in main
1(integer) in main
4(integer) in main
```

Trace:

```
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : branch in main
--- : branch in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
```

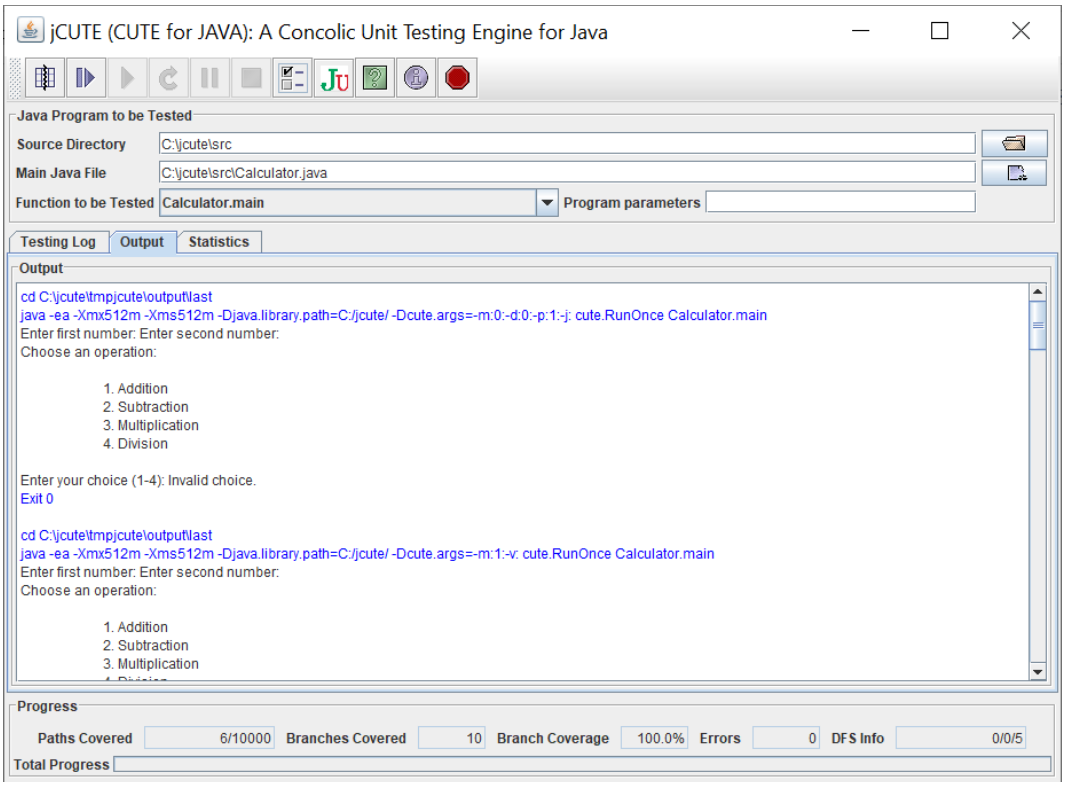
Source: C:\jcute\src\Calculator.java

```java
import cute.Cute;
import java.util.Scanner;
public class Calculator {
  public static void main(String[] args) {
   // Scanner input = new Scanner(System.in);

    System.out.print("Enter first number: ");
    float num1 = Cute.input.Integer();

    System.out.print("Enter second number: ");
    float num2 = Cute.input.Integer();

    System.out.println("\nChoose an operation:\n");
    System.out.println("\t1. Addition");
    System.out.println("\t2. Subtraction");
    System.out.println("\t3. Multiplication");
    System.out.println("\t4. Division");
    System.out.print("\nEnter your choice (1-4): ");
    int choice = Cute.input.Integer();

    float result = 0.0f;
    switch (choice) {
      case 1:
        result = num1 + num2;
        break;
      case 2:
        result = num1 - num2;
        break;
      case 3:
        result = num1 * num2;
        break;
      case 4:
        if (num2 == 0) {
          System.out.println("Error: Division by zero.");
```

Paths Covered: 6/10000 Branches Covered: 10 Branch Coverage: 100.0% Errors: 0 DFS Info: 0/0/5

Total Progress

Figure 7: Testing log



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Java Program to be Tested
Testing Log
Output
Statistics
Output
Progress

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\Calculator.java

Function to be Tested: Calculator.main

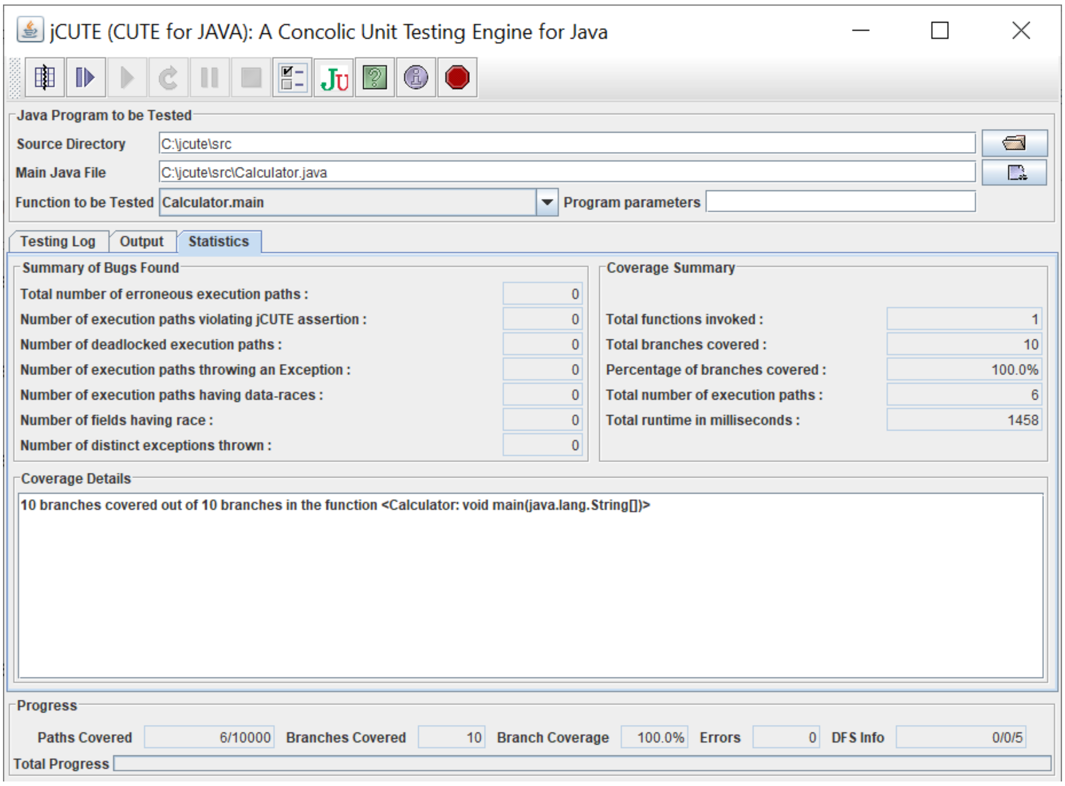
Program parameters:

```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce Calculator.main
Enter first number: Enter second number:
Choose an operation:

                1. Addition
                2. Subtraction
                3. Multiplication
                4. Division

Enter your choice (1-4): Invalid choice.
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce Calculator.main
Enter first number: Enter second number:
Choose an operation:

                1. Addition
                2. Subtraction
                3. Multiplication
                4. Division
```

Paths Covered: 6/10000 Branches Covered: 10 Branch Coverage: 100.0% Errors: 0 DFS Info: 0/0/5

Total Progress

Figure 8: Output



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Testing Log
Output
Statistics

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\Calculator.java

Function to be Tested: Calculator.main

Program parameters:

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	1
Total branches covered :	10
Percentage of branches covered :	100.0%
Total number of execution paths :	6
Total runtime in milliseconds :	1458

Coverage Details

10 branches covered out of 10 branches in the function <Calculator: void main(java.lang.String[])>

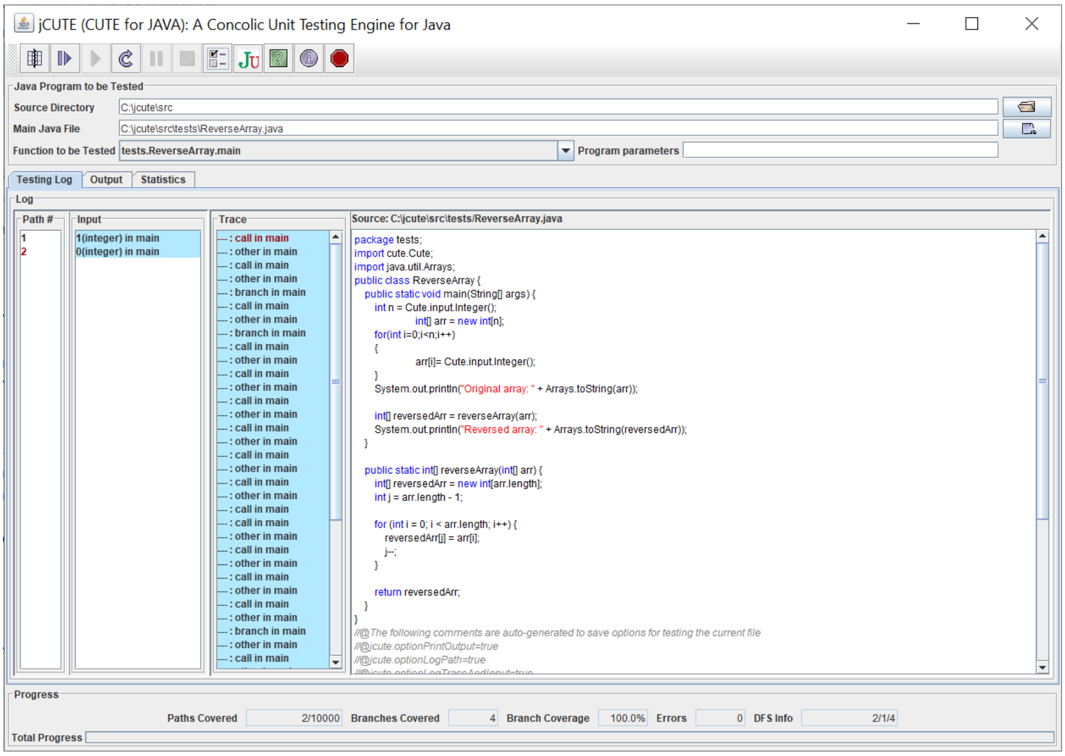
Progress

Paths Covered	6/10000	Branches Covered	10	Branch Coverage	100.0%	Errors	0	DFS Info	0/0/5

Total Progress

Figure 9: Statistics Screenshot

1.4 Reverse an array.



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\ReverseArray.java

Function to be Tested: tests.ReverseArray.main

Program parameters:

Log

Path #	Input
1	1(integer) in main
2	0(integer) in main

Trace

```
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : branch in main
--- : call in main
--- : other in main
--- : branch in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : branch in main
--- : other in main
--- : call in main
```

Source: C:\jcute\src\tests/ReverseArray.java

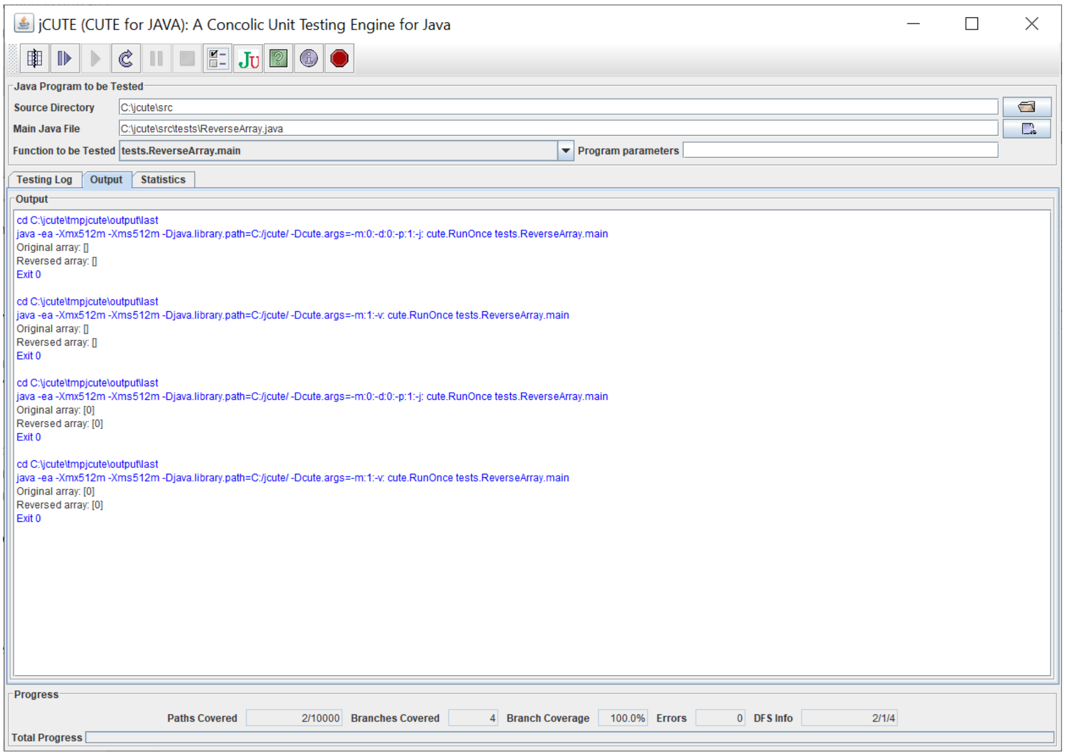
```java
package tests;
import cute.Cute;
import java.util.Arrays;
public class ReverseArray {
  public static void main(String[] args) {
    int n = Cute.input.Integer();
            int[] arr = new int[n];
    for(int i=0;i<n;i++)
    {
            arr[i]= Cute.input.Integer();
    }
    System.out.println("Original array: " + Arrays.toString(arr));

    int[] reversedArr = reverseArray(arr);
    System.out.println("Reversed array: " + Arrays.toString(reversedArr));
  }

  public static int[] reverseArray(int[] arr) {
    int[] reversedArr = new int[arr.length];
    int j = arr.length - 1;

    for (int i = 0; i < arr.length; i++) {
      reversedArr[j] = arr[i];
      j--;
    }

    return reversedArr;
  }
}
//@The following comments are auto-generated to save options for testing the current file
//@jcute.optionPrintOutput=true
//@jcute.optionLogPath=true
//@jcute.optionLogTraceAndInput=true
```

Progress

Paths Covered: 2/10000 Branches Covered: 4 Branch Coverage: 100.0% Errors: 0 DFS Info: 2/1/4

Total Progress

Figure 10: Testing log



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\ReverseArray.java

Function to be Tested: tests.ReverseArray.main

Program parameters:

Output

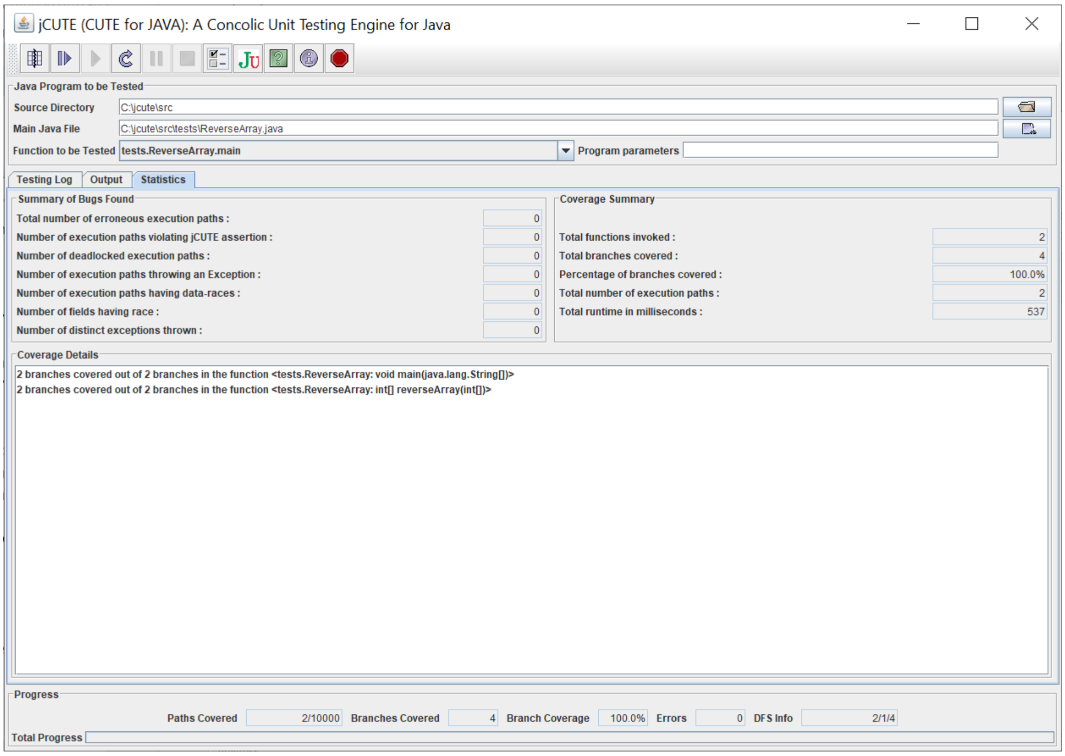
```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.ReverseArray.main
Original array: []
Reversed array: []
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.ReverseArray.main
Original array: []
Reversed array: []
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.ReverseArray.main
Original array: [0]
Reversed array: [0]
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.ReverseArray.main
Original array: [0]
Reversed array: [0]
Exit 0
```

Progress

Paths Covered: 2/10000 Branches Covered: 4 Branch Coverage: 100.0% Errors: 0 DFS Info: 2/1/4

Total Progress

Figure 11: Output


jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java
Testing Log
Output
Statistics

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\ReverseArray.java

Function to be Tested: tests.ReverseArray.main

Program parameters:

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	2
Total branches covered :	4
Percentage of branches covered :	100.0%
Total number of execution paths :	2
Total runtime in milliseconds :	537

Coverage Details

```
2 branches covered out of 2 branches in the function <tests.ReverseArray: void main(java.lang.String[])>
2 branches covered out of 2 branches in the function <tests.ReverseArray: int[] reverseArray(int[])>
```

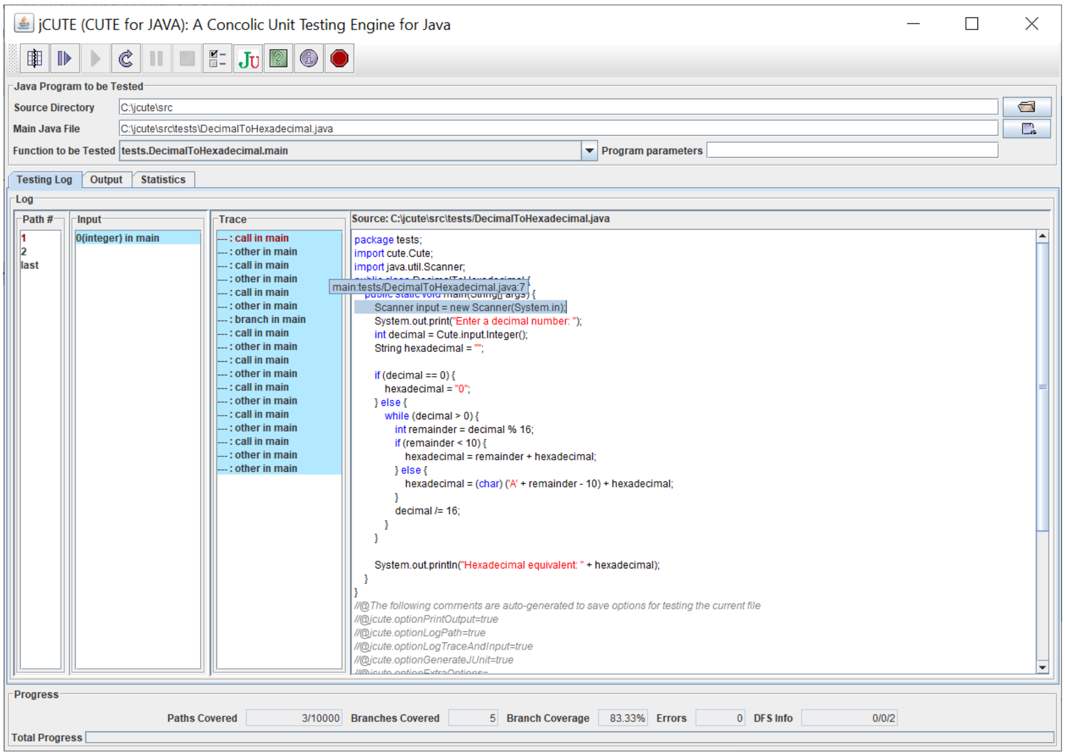
Progress

Paths Covered: 2/10000 Branches Covered: 4 Branch Coverage: 100.0% Errors: 0 DFS Info: 2/1/4

Total Progress

Figure 12: Statistics Screenshot

1.5 Decimal to hexadecimal.



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\DecimalToHexadecimal.java

Function to be Tested: tests.DecimalToHexadecimal.main

Program parameters:

Testing Log | Output | Statistics

Log

Path #	Input	Trace
1	0(integer) in main	--- : call in main
2		--- : other in main
last		--- : call in main
		--- : other in main
		--- : call in main
		--- : other in main
		--- : branch in main
		--- : call in main
		--- : other in main
		--- : call in main
		--- : other in main
		--- : call in main
		--- : other in main
		--- : call in main
		--- : other in main
		--- : call in main
		--- : other in main
		--- : other in main

Source: C:\jcute\src\tests/DecimalToHexadecimal.java

main:tests/DecimalToHexadecimal.java:7

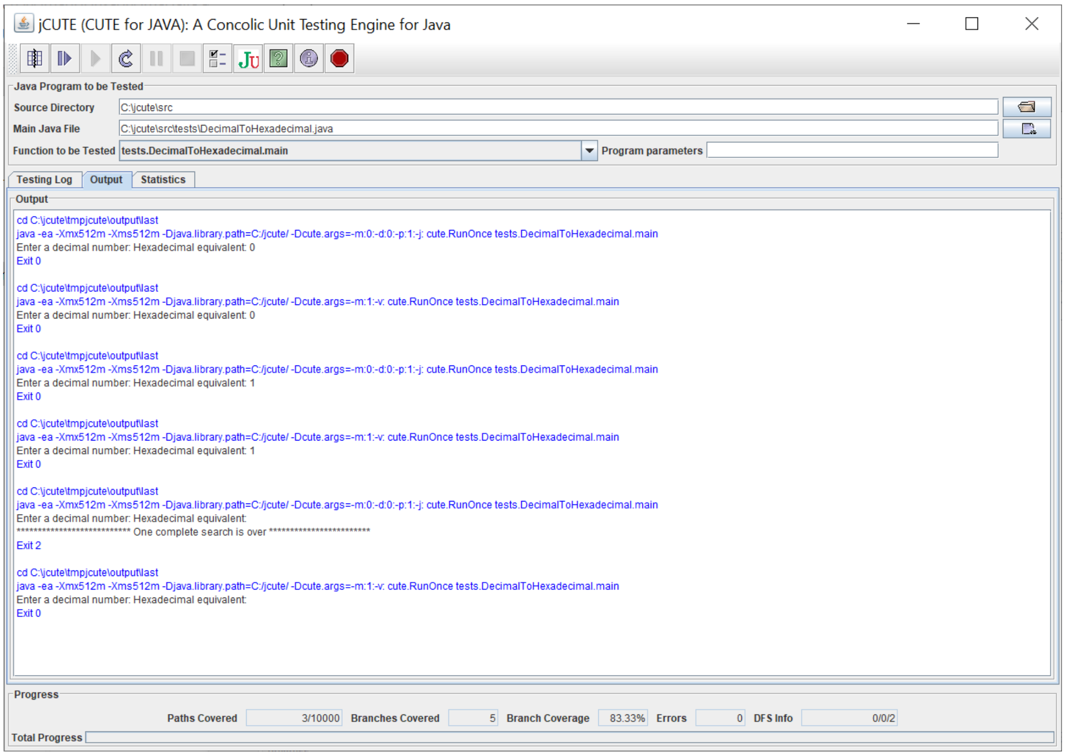
```
package tests;
import cute.Cute;
import java.util.Scanner;
public class DecimalToHexadecimal {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a decimal number: ");
    int decimal = Cute.input.Integer();
    String hexadecimal = "";

    if (decimal == 0) {
      hexadecimal = "0";
    } else {
      while (decimal > 0) {
        int remainder = decimal % 16;
        if (remainder < 10) {
          hexadecimal = remainder + hexadecimal;
        } else {
          hexadecimal = (char) ('A' + remainder - 10) + hexadecimal;
        }
        decimal /= 16;
      }
    }

    System.out.println("Hexadecimal equivalent: " + hexadecimal);
  }
}
//@The following comments are auto-generated to save options for testing the current file
//@jcute.optionPrintOutput=true
//@jcute.optionLogPath=true
//@jcute.optionLogTraceAndInput=true
//@jcute.optionGenerateJUnit=true
//@jcute.optionExtraOptions=
```

Progress

Paths Covered 3/10000 Branches Covered 5 Branch Coverage 83.33% Errors 0 DFS Info 0/0/2

Total Progress

Figure 13: Testing log



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\DecimalToHexadecimal.java

Function to be Tested: tests.DecimalToHexadecimal.main

Program parameters:

Testing Log | Output | Statistics

Output

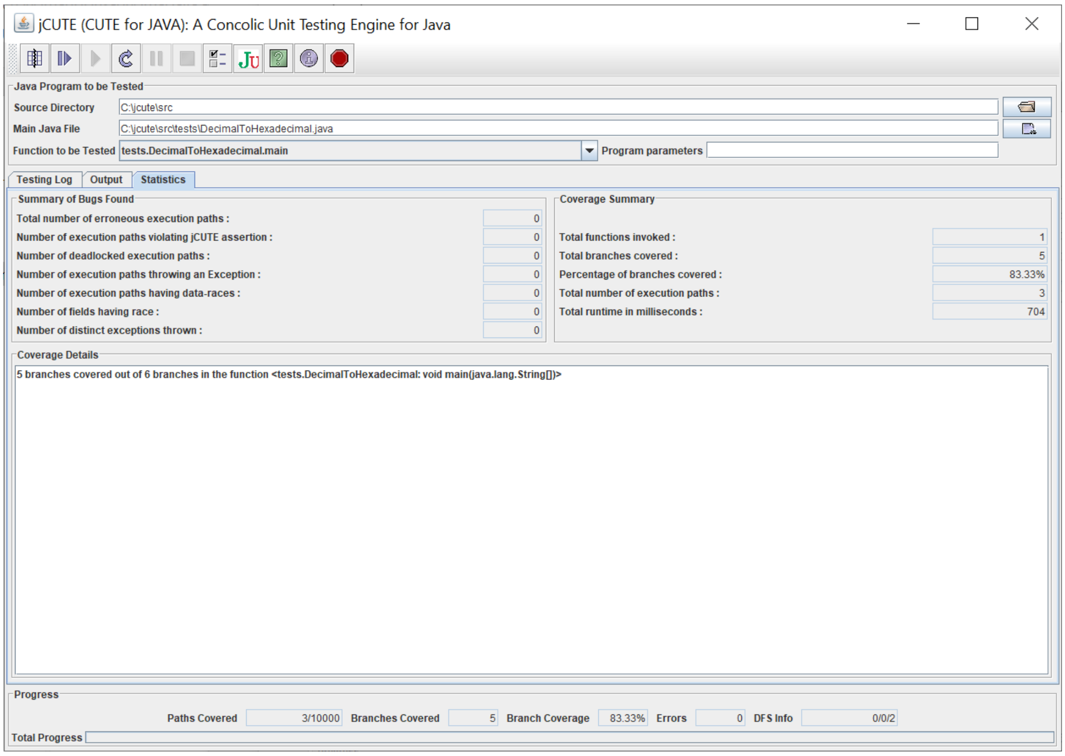
```
cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent: 0
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent: 0
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent: 1
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent: 1
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent:
*************************** One complete search is over ************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.DecimalToHexadecimal.main
Enter a decimal number: Hexadecimal equivalent:
Exit 0
```

Progress

Paths Covered 3/10000 Branches Covered 5 Branch Coverage 83.33% Errors 0 DFS Info 0/0/2

Total Progress

Figure 14: Output



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory	C:\jcute\src
Main Java File	C:\jcute\src\tests\DecimalToHexadecimal.java
Function to be Tested	tests.DecimalToHexadecimal.main
Program parameters	

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	1
Total branches covered :	5
Percentage of branches covered :	83.33%
Total number of execution paths :	3
Total runtime in milliseconds :	704

Coverage Details

```
5 branches covered out of 6 branches in the function <tests.DecimalToHexadecimal: void main(java.lang.String[])>
```

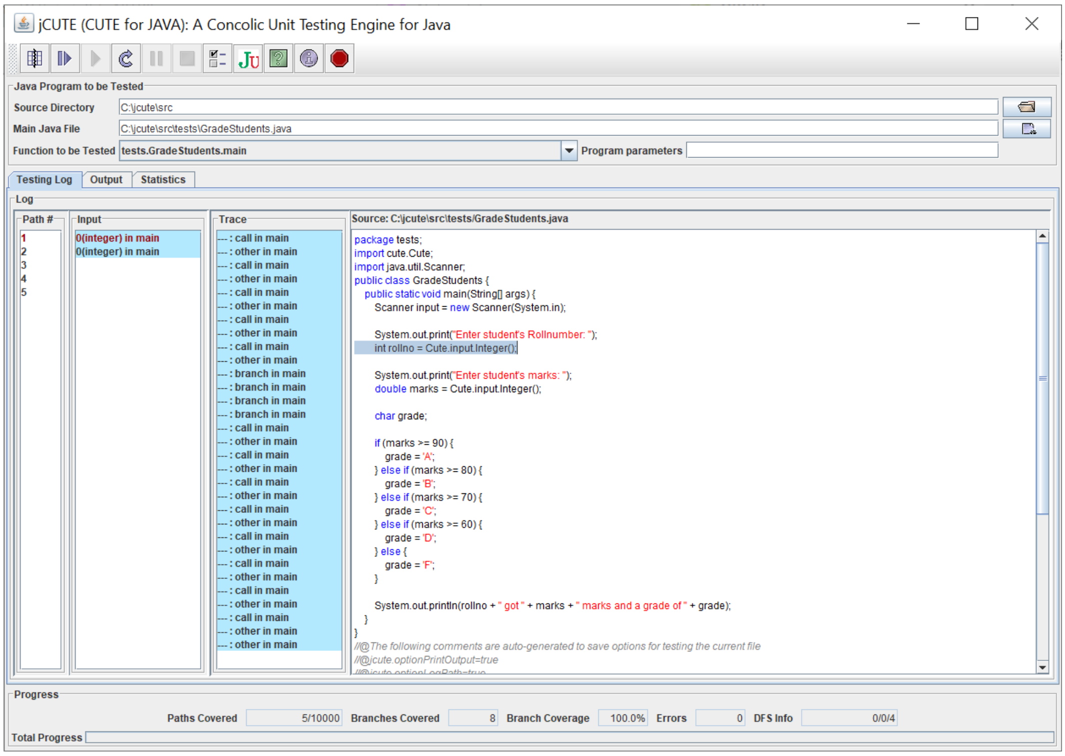
Progress

Paths Covered	3/10000	Branches Covered	5	Branch Coverage	83.33%	Errors	0	DFS Info	0/0/2

Total Progress

Figure 15: Statistics Screenshot

1.6 Grade students based on marks.



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\GradeStudents.java

Function to be Tested: tests.GradeStudents.main

Program parameters:

Path #	Input
1	0(integer) in main
2	0(integer) in main
3	
4	
5	

```
Trace
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : branch in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : call in main
--- : other in main
--- : other in main
```

Source: C:\jcute\src\tests/GradeStudents.java

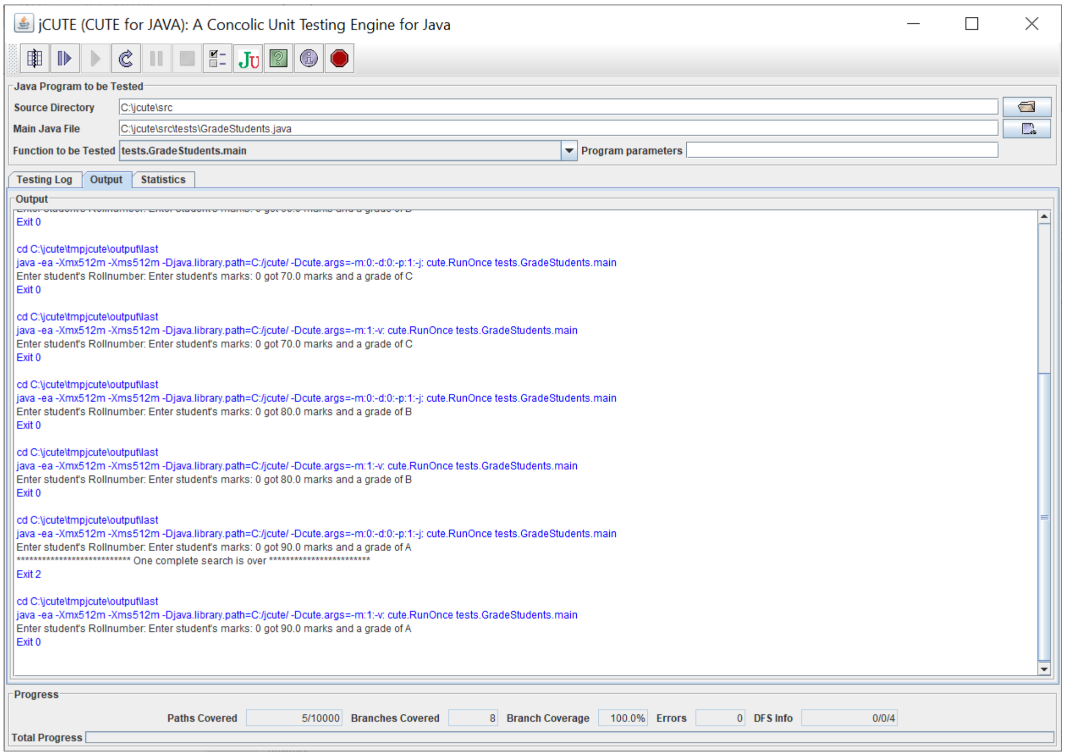
```java
package tests;
import cute.Cute;
import java.util.Scanner;
public class GradeStudents {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter student's Rollnumber: ");
    int rollno = Cute.input.Integer();

    System.out.print("Enter student's marks: ");
    double marks = Cute.input.Integer();

    char grade;

    if (marks >= 90) {
      grade = 'A';
    } else if (marks >= 80) {
      grade = 'B';
    } else if (marks >= 70) {
      grade = 'C';
    } else if (marks >= 60) {
      grade = 'D';
    } else {
      grade = 'F';
    }

    System.out.println(rollno + " got " + marks + " marks and a grade of " + grade);
  }
}
//@The following comments are auto-generated to save options for testing the current file
//@jcute.optionPrintOutput=true
//@jcute.optionLogPath=true
```

Progress

Paths Covered: 5/10000 Branches Covered: 8 Branch Coverage: 100.0% Errors: 0 DFS Info: 0/0/4

Total Progress

Figure 16: Testing log



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\GradeStudents.java

Function to be Tested: tests.GradeStudents.main

Program parameters:

Output

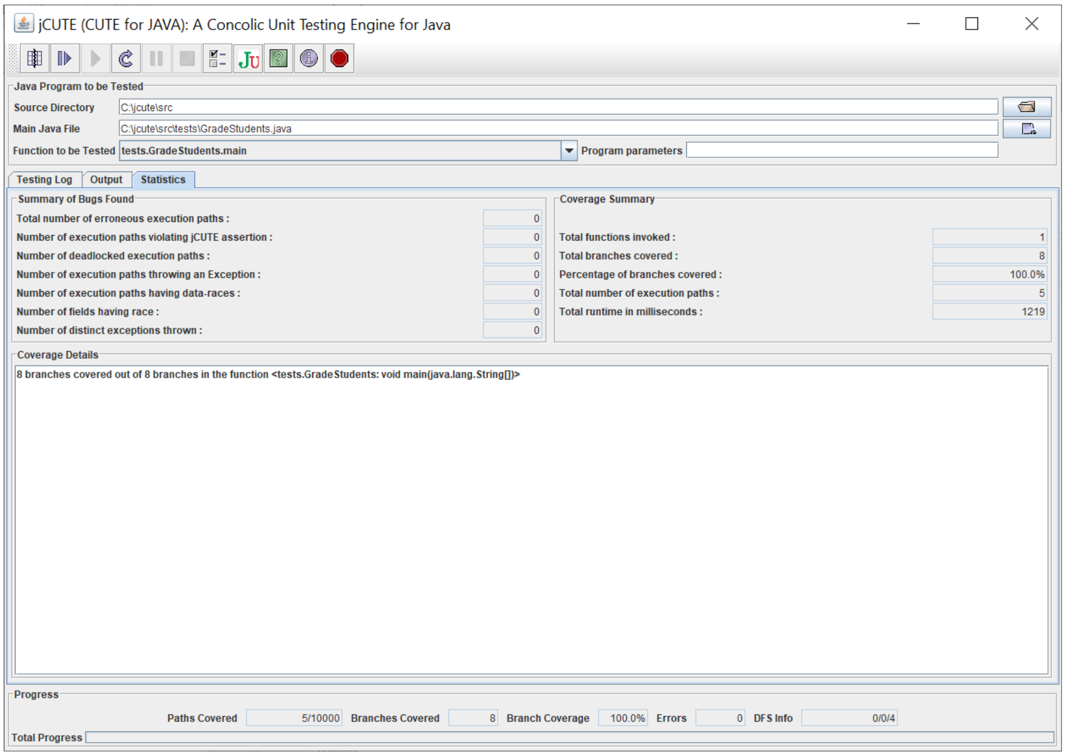
```
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 70.0 marks and a grade of C
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 70.0 marks and a grade of C
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 80.0 marks and a grade of B
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 80.0 marks and a grade of B
Exit 0

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:0:-d:0:-p:1:-j: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 90.0 marks and a grade of A
*************************** One complete search is over ************************
Exit 2

cd C:\jcute\tmpjcute\output\last
java -ea -Xmx512m -Xms512m -Djava.library.path=C:/jcute/ -Dcute.args=-m:1:-v: cute.RunOnce tests.GradeStudents.main
Enter student's Rollnumber: Enter student's marks: 0 got 90.0 marks and a grade of A
Exit 0
```

Progress

Paths Covered: 5/10000 Branches Covered: 8 Branch Coverage: 100.0% Errors: 0 DFS Info: 0/0/4

Total Progress

Figure 17: Output



jCUTE (CUTE for JAVA): A Concolic Unit Testing Engine for Java

Java Program to be Tested

Source Directory: C:\jcute\src

Main Java File: C:\jcute\src\tests\GradeStudents.java

Function to be Tested: tests.GradeStudents.main

Program parameters:

Summary of Bugs Found

Total number of erroneous execution paths :	0
Number of execution paths violating jCUTE assertion :	0
Number of deadlocked execution paths :	0
Number of execution paths throwing an Exception :	0
Number of execution paths having data-races :	0
Number of fields having race :	0
Number of distinct exceptions thrown :	0

Coverage Summary

Total functions invoked :	1
Total branches covered :	8
Percentage of branches covered :	100.0%
Total number of execution paths :	5
Total runtime in milliseconds :	1219

Coverage Details

```
8 branches covered out of 8 branches in the function <tests.GradeStudents: void main(java.lang.String[])>
```

Progress

Paths Covered: 5/10000   Branches Covered: 8   Branch Coverage: 100.0%   Errors: 0   DFS Info: 0/0/4

Total Progress

Figure 18: Statistics Screenshot