

Test Cases

Code-Based Test Cases

1) Testing input and output, as well as string variables.

```
Scanner testObject = new Scanner(System.in);  
System.out.println("Enter an item.");  
String outputTest = testObject.nextLine();  
System.out.println("Entered item: " + outputTest);
```

2) Testing loops

```
For (int i = 0; i < 7; i++){  
System.out.println(i);  
}
```

3) Testing arithmetic operations and numeric variable assignment.

```
Int testVariable1 = 10;  
float testVariable2 = 15.5;  
System.out.println(testVariable 1 + testVariable2);  
System.out.println(testVariable 1 - testVariable2);
```

```
System.out.println(testVariable 1 * testVariable2);
```

```
System.out.println(testVariable 1 / testVariable2);
```

```
System.out.println(testVariable 1 % testVariable2);
```

4) Ensuring that the user cannot inject code into the game world, since both the terminal and game are written in Java.

a)

```
/weather thunder
```

b)

```
/GameMode creative
```

c)

```
/Time set day
```

5) Ensure that code with errors does not execute.

a) No quotation marks inside the print statement.

```
System.out.println(hello world)
```

b) Attempting to divide by zero.

```
Int testCase = 50 / 0;
```

```
System.out.println(testCase);
```

c) Attempting to print variables from a previous program.

First program:

```
Int testCase = 5;
```

```
System.out.println(testCase);
```

Run program. Close terminal, begin new program.

```
System.out.println(testCase);
```

Should return error.

6) Ensure the terminal behaves accordingly when it is run with no input.

Simply click “run” with nothing inside the terminal.

7) Ensure the terminal behaves accordingly when it is run with “unusual” characters.

```
System.out.println("€ ^&* ¢");
```

8) Create an array, and read from it.

```
Int arrayTest[] = {1, 2, 3};
```

```
System.out.println("Second integer array value: " + arrayTest[1] + "\narray  
length: " + arrayTest.length);
```

```
String[] arrayTestTwo = new String[]{"One", "Two", "Three"};
```

```
System.out.println("Second string array value: " + arrayTestTwo[1] + "\nString  
array length: " + arrayTestTwo.length);
```

In-game Manual Test Cases

Verify that the terminal correctly launches in-game.

Simply right clicking the terminal block and doing so in multiple locations.

Ensuring that terminal still opens and functions if multiple are in the game world, or in render distance.

Verify that users receive rewards when a problem is solved

Walk through some/all of the exercises and ensure that when a problem is solved, the user receives the designated reward.

Verify that the terminal is fully functional and reliable.

Ensure that the in-game terminal can handle enough code such that the longest exercise can be written and executed. IE: attempt to execute a program with 40-50 lines of code.

Stress test terminal

A) Create a program that calculates all prime numbers between 1 and 500. Attempt to execute the program multiple times in quick succession if possible.

```
for (int i = 2; i <= 500; i++) {  
    if (isPrime(i)) {  
        System.out.println(i);  
    }  
}
```

```

        }
    }
}

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

```

Source for some of the code, if required:

<https://www.baeldung.com/java-generate-prime-numbers>

In-Game Automated Test Cases

```
new TestCase(
```

"Test 1: Successful Compilation and Execution",

```
"public void playerMain() {\n" +  
    "    System.out.println(\"Hello, World!\");\n" +  
    "}",  
"Hello, World!\n"  
)
```

// Test 2: Compilation Error (Missing Semicolon)

```
new TestCase(  
    "Test 2: Compilation Error (Missing Semicolon)",  
    "public void playerMain() {\n" +  
        "    System.out.println(\"Missing semicolon\")\n" + // Missing  
semicolon  
        "}",  
    "Compilation failed:\n" +  
        "Error on line 3: ';' expected\n"  
)
```

// Test 3: Runtime Error (NullPointerException)

```
new TestCase(  
    "Test 3: Runtime Error (NullPointerException)",  
    "public void playerMain() {\n" +  
        "    String s = null;\n" +
```

```
        "    System.out.println(s.length());\n" + // This will cause  
NullPointerException
```

```
        "}",  
        "java.lang.NullPointerException"  
    ),
```

// Test 4: Testing Loops

```
new TestCase(  
    "Test 4: Testing Loops",  
    "public void playerMain() {\n" +  
        "    for (int i = 0; i < 7; i++) {\n" +  
        "        System.out.println(i);\n" +  
        "    }\n" +  
        "}",  
    "0\n1\n2\n3\n4\n5\n6\n"  
),
```

// Test 5: Testing Arithmetic Operations and Numeric Variable Assignment

```
new TestCase(  
    "Test 5: Testing Arithmetic Operations and Numeric Variable  
Assignment",  
    "public void playerMain() {\n" +  
        "    int testVariable1 = 10;\n" +
```

```
        " float testVariable2 = 15.5f;\n" + // Added 'f' suffix for float
literal
        " System.out.println(testVariable1 + testVariable2);\n" +
        " System.out.println(testVariable1 - testVariable2);\n" +
        " System.out.println(testVariable1 * testVariable2);\n" +
        " System.out.println(testVariable1 / testVariable2);\n" +
        " System.out.println(testVariable1 % testVariable2);\n" +
        "}",
        "25.5\n-5.5\n155.0\n0.6451613\n10.0\n"
    ),
```

// Test 6: Testing Arrays

```
new TestCase(
    "Test 6: Testing Arrays",
    "public void playerMain() {\n" +
        " int[] numbers = {10, 20, 30, 40, 50};\n" +
        " for (int number : numbers) {\n" +
        "     System.out.println(number);\n" +
        " }\n" +
        "}",
    "10\n20\n30\n40\n50\n"
),
```

// Test 7: Testing a While Loop


```

new TestCase(
    "Test 7: Testing a While Loop",
    "public void playerMain() {\n" +
        "    int count = 5;\n" +
        "    while (count > 0) {\n" +
        "        System.out.println(count);\n" +
        "        count--;\n" +
        "    }\n" +
        "}",
    "5\n4\n3\n2\n1\n"
),

```

// Test 8: Creating a Method and Calling It from playerMain

```

new TestCase(
    "Test 8: Creating a Method and Calling It from playerMain",
    "public void playerMain() {\n" +
        "    greet(\"Minecraft User\");\n" +
        "}\n\n" +
        "public void greet(String name) {\n" +
        "    System.out.println(\"Hello, \" + name + \"!\");\n" +
        "}",
    "Hello, Minecraft User!\n"
)

```