Software Testing

Chris Brown
North Carolina State University
Department of Computer Science

Learning Objectives

- Understand the importance of testing and discover types of testing for programs

- Apply testing principles and perform basic unit testing practices with the JUnit framework for Java code.

Testing

- Testing is the process of finding software faults

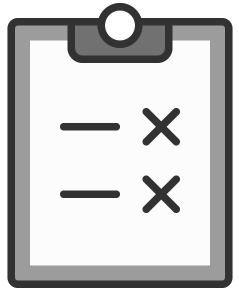
  - **Fault:** “an incorrect step, process, or data definition in a program”

Example: Program to determine if a given number is odd or even.

Why Test Software?

- Test cases uncover failure by finding where a program's *actual behavior* deviates from the *expected behavior*.
- Testing increases confidence that your program works correctly and meets the requirements.

  - No such thing as perfect code!
- Test Early and Test Often!

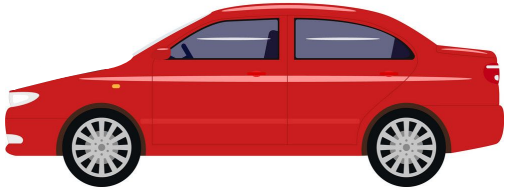


Benefits

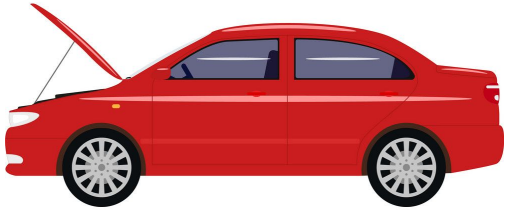
- Improves code quality,
- Catching bugs earlier reduces debugging time and costs,
- Provides documentation,
- Enhances software maintenance,
- Increases ability to add more features iteratively (agile),
- and more!

Types of Testing

1. **Black Box Testing:** ignores the internals of the program and focuses on functionality



2. **White Box Testing:** uses code to guide testing



White Box Testing

- Also known as unit testing
  - **unit:** individual functions or methods
- Test all of your program's methods (except main!)
- Create testing a separate Java test class
  - Usually <ClassName>Test.java in a test/ folder

`src/EvenOrOdd.java → test/EvenOrOddTest.java`

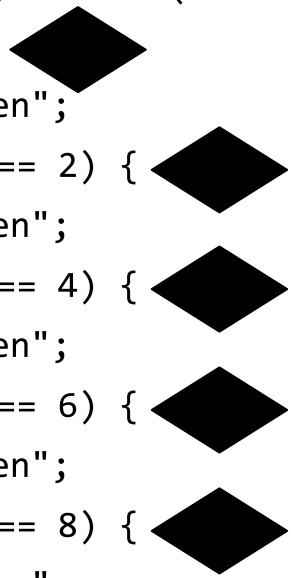
Cyclomatic Complexity

- Measure of a method's complexity: number of independent paths in the basis set of method.
- ***Minimum*** number of test cases to write for a method
    - **Code Coverage:** percentage of lines executed in tests

Cyclomatic Complexity = #decisions + 1

- decisions: if, else if, loop control, boolean operators,...

```
public static String evenOdd(int num) {
     if (num == 0) {
          return "Even";
     } else if (num == 2) {
          return "Even";
     } else if (num == 4) {
          return "Even";
     } else if (num == 6) {
          return "Even";
     } else if (num == 8) {
          return "Even";
     } else {
          return "Odd";
     }
}
```



6

```
public static String evenOdd(int num) {
    if (num == 0 || num == 2 || num == 4 || num == 6 || num == 8) {
        return "Even";
    }
    return "Odd";
}
```

6

```
public static void evenOddToNum(int num) {
    for (int i = 1; i <= num; i++) {
        if (i % 2 != 0) {
            System.out.print(i + " ");
        }
}
```

3

JUnit

- Unit testing framework for Java
- External library
- JUnit must be downloaded and referenced when compiling and executing

  1. hamcrest-core-1.3.jar
  2. junit-4.12.jar

  - go.ncsu.edu/chris-teaching-demo

JUnit Conventions

- Test cases can be broken out into individual methods.
- Naming convention:
  `test<MethodName><DescriptionOfTest>`

```
/**
 * Determines if a number is even or odd
 * @param num number input by user
 * @return String if value is even or odd
 */
public static String evenOdd(int num) {
```

0?	`testEvenOddZero`
1?	`testEvenOddOddNumber`
2?	`testEvenOddEvenNumber`
50?	`testEvenOddFifty`
negative number?	`testEvenOddNegative`
9.5?	`testEvenOddDecimal`

- @Test is used to identify each test method in your test class.

Assert Statements

- Assert statements “check” your expected and actual results
- If the expected results do not match the actual results, the test will FAIL
- A test case must have ***at least 1*** assert statement, otherwise the test case isn’t actually testing anything!

Assert Statements

Statement	Description
assertTrue(message, value)	asserts that the value passed as a parameter is boolean true. If it is not true, the test case will fail with the given message.
assertFalse(message, value)	asserts that the value passed as a parameter is boolean false. If it is not false, the test case will fail with the given message.
assertEquals(message, expectedValue, actualValue)	asserts that expectedValue equals the actualValue. If the two values are not equal, the test case will fail with the given message.

Assert Statements

```
public static String evenOdd(int num) {
    if (num == 0 || num == 2 || num == 4 || num == 6 || num == 8) {
        return "Even";
    } else {
        return "Odd";
    }
}
```

1. assertEquals("The expected output is Even", "Even", evenOdd(6)); // PASS
2. assertEquals("The expected output is Odd", "Odd", evenOdd(1)); // PASS
3. assertEquals("The expected output is Even", "Even", evenOdd(10)); // FAIL
4. assertTrue("The expected output is Odd", "Odd".equals(evenOdd(99)); // PASS
5. assertFalse("The expected output is Even", evenOdd(8).equals("Even")); // FAIL

Project Structure

project/ (top-level project directory)

- **src** (where your .java files should be stored)
- **test** (where your test .java files should be stored)
- **lib** (where your Junit (and other external libraries) archive files should be stored)
- **bin** (where your compiled .class files should be stored)

Compilation

1. Compiling Source Code:

```
$ javac -d bin src/<ClassName>.java
```

2. Compiling Test Code:

```
$ javac -d bin -cp bin:lib/* test/<ClassName>Test.java
$ javac -d bin -cp "bin;lib\*" test/<ClassName>Test.java
```

Execution

1. Executing Source Code:

```
$ java -cp bin <ClassName>
```

2. Test Code:

```
$ java -cp bin:lib/* org.junit.runner.JUnitCore <ClassName>Test
$ java -cp "bin;lib\*" org.junit.runner.JUnitCore <ClassName>Test
```

Remainder of Class

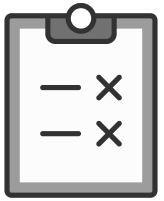
- Live Coding Example: Testing EvenOrOdd.java
- Group or HW Assignment

  - Fix EvenOrOdd and write EvenOrOddTest.java

- Prerequisites:

  - IDE or text editor
  - Java 11.0.7
  - hamcrest-core-1.3.jar and junit-4.12.jar
  - Project Structure (src, bin, test, and lib)
  - go.ncsu.edu/chris-teaching-demo

Wrap Up



- Testing is crucial for developing reliable systems and provides many benefits to development teams
- White box (know the code) and black box (ignore code) are two types of testing
- Cyclomatic complexity measures the decisions in your code
- JUnit: automated unit testing framework for Java