Instrumentation

SWE 261P

Execution Tracing and Profiling

- Gathering dynamic information about programs
  - Execution tracing
  - Execution profiling
  - Execution coverage
- Instrumenting for tracing, profiling, coverage
  - Preprocessing
  - Online processing
  - Postprocessing

Execution Tracing

Execution Tracing records, as the program executes with a test case (an input to the program), some sequence of events that occur. For example:

- the sequence of statements executed when a program is run with a test case
- the sequence of program states associated with statements executed when a program is run with a test case

Execution Profiling

Execution Profiling records, as the program executes with a test case, some quantity that an event occurs. For example:

- the number of times a statement is executed when a program is run with test case
- the number of times a variable is changed when the program is run with a test case
- the time spent in a method

Execution Coverage

Execution Coverage records, as the program executes with a test case, whether an event occurs. For example:

- whether a statement is executed when a program is run with test case
- whether a branch is taken when the program is run with a test case
- whether a variable is changed when the program is run with a test case

slow: trace, since it needs to print every sequence
not slow: profile, set a data structure, if execute one sentence, plus 1
more easy: set all sentence are zero, if execute, it becomes 1, return true

Instrumentation

Instrumentation is the process of adding code to a program (called probes) such that when the program is executed, it records information about its execution

The program with the probes is called an instrumented program

Instrumentation

Types of instrumentation

more fast, optimize

- Preprocessing — use of instrumenter (i.e., probes inserted into original program to produce modified program)

recode something just VM knows

- Online processing — usually VM monitoring

- (subtype, or metatype) Postprocessing — postprocessing one of the other types (e.g. trace to coverage)

Instrumentation Tools

- Modifies program at the
  - Source code level, before compilation
  - Bytecode/machine code, during or after compilation
- ***Should*** preserve the behavior of the program (but may not succeed)

```
in multi-thread,probe may slow down, causing block
```

Some Instrumenters

- GCC/Gcov
- Cobertura
- Jacoco
- EclEmma
- Clover
- Java JVM TI
- ...

TriangleType Exercise

- Install a code coverage plugin for your IDE. For Eclipse, go the the Marketplace and install EclEmma. For IntelliJ, make sure the Code Coverage plugin is installed (I think it is by default).
- In Canvas (under Files), there is a Java Jar file called TriangleType.jar. Download this file to your computer. This jar file contains a class called TriangleType, which contains a public static method called triangleType(). This method takes three integers as input and outputs an integer. The three input variables each describe the lengths of each side of a triangle. The lengths of the sides of the triangle should be less than or equal to 1000. The output of this method will be one of 5 possible values: 1 for a scalene triangle, 2 for an isosceles triangle, 3 for an equilateral triangle, 4 for values that do not describe a triangle, and 5 for values that are out of bounds.
- Using Eclipse, create a new project. Under the Build Properties for the project, add the jar file as an external library. Create a new "JUnit Test Case." The new JUnit Test Case Class should be named TriangleTypeTest. The "Class Under Test" is TriangleType. You will be creating test cases for the method TriangleType.triangleType().