iTutor: Assisting Students Learning Java Programming  
Bellanov Apilli, Dr. Tao Xie  
bsapilli@ncsu.edu, xie@csc.ncsu.edu

Problem

- Insufficient or inadequate tutoring services at introductory levels of programming courses
- Students are forced to absorb a lot of technical information in a short period of time
- Few number of tutors allotted for courses with a significantly high amount of students

Proposed Solution:

- An approach and a tool, iTutor
- Generates regression tests that expose behavioral differences between a student’s and tutor’s program
- Provides feedback on how and why a student’s solution differs from that of a tutor

Student & Tutor Solutions

- iTutor accepts a **student** and **tutor** solution to a given programming problem
- **Interface** information is derived from the tutor solution and serves as a third input
  - Student solution is required to follow the interface derived from the tutor solution
  - iTutor can only test methods present in both solutions

Solution Analysis

- iTutor initially accepts **student**, **tutor**, and **interface** inputs:
  - Student & tutor solutions are class files written in Java bytecode
  - Bytecode are formal instructions that the Java Virtual Machine executes
- Solutions are **decompiled** and class information is stored in 3 instance variables that map to each solution (I: interface, T: tutor, S: student)
  - Student solution is required to follow the interface derived from the tutor solution
- iTutor can only test methods present in both solutions
- iTutor extracts method information from the instance variables
  - Extracted information is encoded and stored in a file for further analysis

Method Sequences

- A **method sequence** is a series of method calls that make up a path a program could possibly take, for instance, a simple Stack:
  - Sequence 1: push(1) -> push(1) -> pop() -> push(1) -> pop() -> pop()
  - Sequence 2: pop() -> push(1) -> pop() -> push(1) -> pop() -> pop()
  - Although the sequences are similar in the calls made, they are two different method sequences due to the order in which the calls are made
- Upon each execution of iTutor, different sets method sequences are generated
  - Solutions can be implemented differently but the general behavior should be the same
  - iTutor invokes method sequences on both the student and tutor solutions
  - Return values are compared for methods that are not void
  - Observer methods monitor instance variables as void methods are executed

JCute

- JCUET (Concolic Unit Testing Engine for Java) is a tool used to systematically test Java programs
  - Uses concolic execution techniques to explore all execution paths of a program with data inputs
  - Generates appropriate inputs to direct program to alternate paths
  - Chooses appropriate values in conditional statements, loops, parameters to functions, etc.
  - Automatically generates optimal JUnit tests cases
- JCUTE is responsible for choosing the appropriate values for variables in each solution
  - Can achieve optimal path coverage, branch coverage and detect erroneous execution
  - Can record and replay an execution of a program

Anticipated Contributions & Future Work

- iTutor is currently implemented as a plugin for the Eclipse Application
  - Initially, we plan to provide feedback in the form of specific scenarios (behavior differs between solutions)
  - We plan to expand iTutor to adapt to natural language processing techniques in order to give more constructive feedback
- We believe iTutor can help students in introductory levels of programming better understand their courses
  - The tool is always available wherever an Eclipse application is installed
  - Serves as an alternative to tutors when they are unavailable