Workshop Title: The Absolute Beginner's Guide to JUnit in the Classroom

Presenters:

Stephen H. Edwards (Contact Person)
Virginia Tech, Dept. of Computer Science
114 McBryde Hall (0106)
Blacksburg, VA 24061
Phone: 1-540-231-5723
Fax: 1-540-231-6075
E-mail: edwards@cs.vt.edu

Manuel A. Pérez-Quiñones
Virginia Tech, Dept. of Computer Science
1125 Knowledge Works II (0106)
Blacksburg, VA 24061
Phone: 1-540-231-2646
Fax: 1-540-231-6075
E-mail: perez@cs.vt.edu

Abstract: Software testing has become popular in introductory courses, but many educators are unfamiliar with how to write software tests or how they might be used in the classroom. This workshop provides a practical introduction to JUnit for educators. JUnit is the Java testing framework that is most commonly used in the classroom. Participants will learn how to write and run JUnit test cases; how-to's for common classroom uses (as a behavioral addition to an assignment specification, as part of manual grading, as part of automated grading, as a student-written activity, etc.); and common solutions to tricky classroom problems (testing standard input/output, randomness, main programs, assignments with lots of design freedom, assertions, and code that calls `exit()`).

Intended audience: This workshop is intended for CS educators who want to write software tests or who want students to write tests for their own assignments. No prior experience with testing is required. Java fluency is expected. Basic knowledge of Eclipse is preferred, but not required. The examples presented will focus on CS1- and CS2-level Java assignments.

Presenter Biographies:
Stephen Edwards is the lead designer and project manager for Web-CAT—winner of the 2006 Premier Award, which recognizes high-quality, non-commercial courseware designed to enhance engineering education. He is also the author of LIFT, a library for writing JUnit-based tests for GUI applications, and a number of testing tools that make writing classroom-oriented software tests easier for students and for teachers.

Together with Manuel Pérez-Quiñones, Edwards has given several successful tutorials, workshops, and demonstrations on how to incorporate software testing into programming assignments. He has held workshops at SIGCSE, OOPSLA, and CCSC-E. Most recently, they presented a workshop at SIGCSE 2011 on testing GUI-based applications written using Java Swing or the ACM JTF library. Feedback was overwhelmingly positive, with all participants responding to the survey giving an overall rating of good or excellent, and with three quarters rating the workshop as excellent. 100% of evaluations agreed or strongly agreed regarding recommending "a workshop offered by these same presenters." The previous version of their Web-CAT workshop at SIGCSE 2007 received 100% "strongly agree" ratings on whether participants would recommend it to colleagues.

Materials provided: Participants will be provided with electronic access to all presentation materials and examples, all required jar-files to run the software, as well as tutorials on how to make use of each of the classroom strategies for leveraging software tests.

Rough Agenda:
The goal is to **give participants a solid introduction to writing JUnit tests**. The workshop will be discussion-oriented and driven primarily by live examples. As examples are demonstrated, participants can ask questions as they arise and answers or alternatives can be illustrated in real time. In addition, three hands-on "practice" activities will give participants a chance to work with live examples of the content. The workshop will include:

1. A quick introduction to concepts of unit testing and the various strategies commonly used to employ it in the classroom—public "acceptance tests" as part of an assignment specification, private "reference tests" run by course staff during manual grading, "reference tests" used for

automated grading, student-written tests for self-checking and/or grading, and student-written tests of pre-existing code for developing debugging/testing skills. (15 minutes).

2. An overview of JUnit and how it works, both on the command line and in typical IDEs (Eclipse and BlueJ); the difference between JUnit 3 and JUnit 4 (15 minutes).

3. Hands-on activity (and discussion): Practice writing JUnit tests for a classroom example (participants have two to choose from) (20 minutes).

4. An overview of common problems that instructors run into when writing software tests, and how they can be resolved—testing standard input/output behaviors, testing programs that use random numbers, testing `main()` programs instead of individual methods, testing code with assertions, and testing code that calls `exit()` (25 minutes).

5. Hands-on activity (and discussion): Practice writing JUnit tests for a classroom example that uses stdio or random numbers (participants have two to choose from. (20 minutes plus 15-minute break.)

6. Using reference tests for automated grading, and the issues it raises. Writing software tests for student programs that have little or no constraints on student code structure (i.e., "open-ended" design problems, where students come up with the design themselves). Practical strategies for giving feedback (30 minutes).

7. Hands-on activity (and discussion): Practice adding JUnit tests for an assignment. Participants pick the assignment (two to choose from) and pick the strategy they wish to employ (public tests to enhance the assignment spec, reference tests for grading, requiring students to write tests, etc.), and adapt the assignment and/or practice writing the corresponding tests. Participants exchange results and discuss. (30 minutes).

8. Wrap-up: open discussion of testing issues. Participants can discuss their own experiences, and ask questions about classroom experience or how specific assignment issues can best be tested or handled (10 minutes).

Audio/Visual and Computer requirements: Overhead projection for one computer for the presenter is the only AV requirement. We request that power strips be available in case participants want to use their own laptops.

Laptop Recommended: This is a hands-on workshop that is driven both by discussion and by practice on examples. Participants who do bring laptops will be able to follow along with the examples presented. In addition, Participants who do not have a laptop can benefit from the live demonstrations of code writing and program execution, and use the hands-on time for discussion (or partner with another participant).

Space and Enrollment restrictions: None.

Other critical information: None.