Extending and Contributing to an Open Source Web-based System for the Assessment of Programming

Dr. Christelle Scharff
Dr. Olly Gotel
Pace University, New York, USA

Dr. Andrew Wildenber
Cornell College, Iowa, USA
WeBWorK in CS

Outline

• Pedagogical Context
• Systems for Automated Assessment of Programming Assignments
• WeBWorK
• WeBWorK-JAG
• Findings and Lessons Learned from the Use of WeBWorK for Homeworks
• Students’ Contributions to WeBWorK
• Conclusions and Future Work
WeBWork in CS

Pedagogical Context

• Programming is the first skill a computer science major is expected to master
• Programming fundamentals are taught in CS1 and CS2 courses [CC2005]
  – Fundamental programming constructs, algorithms and problem solving, elementary data structures, recursion, event-driven programming
• Open source for early exposure to collaborative and community-driven development
• Test-driven development for emphasizing the criticality of formulating requirements in a testable manner and laying the basis for quality coding
• Peer-review for giving students an awareness of the value of getting an independent person to examine code and detect errors before release and use by others
WebWork in CS

Systems for Automated Assessment of Programming Assignments

• Web-based systems to encourage practice (with feedback), and improve and reinforce students’ understanding of concepts

• Types of questions
  – True / false, short answer, multiple-choice, programming

• Grading programs
  – Correctness + quality + authenticity
Existing Systems

- **Boss**  www.dcs.warwick.ac.uk/boss
- **CodeLab**  www.turingscraft.com
- **CourseMarker**  www.cs.nott.ac.uk/CourseMarker
- **Gradiance**  www.gradiance.com
- **JavaBat**  www.javabat.net
- **MyCodeMate**  www.mycodemate.com
- **OWL**  owl.course.com
- **Viope**  www.viope.com
WeBWorK in CS

WeBWorK

- webwork.rochester.edu
- Project funded by NSF
- Free, open-source and web-based
- Automated problem delivery and grading
- Initial development and applications in the fields of mathematics and physics
- Currently in use at more than 50 colleges and universities
WeBWorK in CS

WeBWorK

• Problems are written in the Problem Generating macro language (PG)
  – Text, HTML, Latex, Perl

• Underlying engine dedicated to dealing with mathematical formulae
  – \( x+1 = (x^2-1)/(x-1) = x+\sin(x)^2+\cos(x)^2 \)

• Individualized and parameterized versions of problems
WeBWorK in CS

WeBWorK for Programming Fundamentals

- atlantis.seidenberg.pace.edu/webwork2/demo
- True / false, short answer and multiple choice problems for Java, Python and SML
- Extension of WeBWorK for use in programming fundamentals
- Evaluation of Java program fragments by interfacing WeBWorK with JUnit [www.junit.org]
  - WeBWorK-JAG = WeBWorK + JUnit
set1: Problem 2

(1 pt)
VARARGS

Check all the true statements about varargs.

- A. Varargs permits us to pass in many arguments to a method when the number of arguments is not known at compile time
- B. A method can have multiple varargs as parameters
- C. Varargs must appear as the last parameter of a method
- D. Varargs must appear as the first parameter of a method
- E. Varargs are not allowed in Java 1.4.
- F. Varargs are treated as arrays by the compiler

Preview Answers  Submit Answer

You have attempted this problem 0 times.
You have unlimited attempts remaining.

Email instructor

Page generated at 10:35am on May 21, 2007
WebWork.Â© 2000-2006 The WebWork Project
WebWork in CS

set1: Problem 2

The above answer is correct.

(1 pt)
VARARGS

Check all the true statements about varargs.

- A. Varargs permits us to pass in many arguments to a method when the number of arguments is not known at compile time
- B. A method can have multiple varargs as parameters
- C. Varargs must appear as the last parameter of a method
- D. Varargs must appear as the first parameter of a method
- E. Varargs are not allowed in Java 1.4.
- F. Varargs are treated as arrays by the compiler

Your score was recorded.
You have attempted this problem 2 times.
You received a score of 100% for this attempt.
Your overall recorded score is 100%
set1: Problem 2

The above answer is NOT correct.

(1 pt)
VARARGS

Check all the true statements about varargs:

☐ A. Varargs permits us to pass in many arguments to a method when the number of arguments is not known at compile time
☐ B. A method can have multiple varargs as parameters
☐ C. Varargs must appear as the last parameter of a method
☐ D. Varargs must appear as the first parameter of a method
☐ E. Varargs are not allowed in Java 1.4.
☐ F. Varargs are treated as arrays by the compiler

Your score was recorded.
You have attempted this problem 1 time.
You received a score of 0% for this attempt.
Your overall recorded score is 0%
set1: Problem 1

SUM OF EVEN NUMBERS

Write a method that computes the sum of the even numbers from 0 to a given limit (included).

The method will be called `sumEven` and must:
- Be declared public and static;
- Take one parameter of type `int` representing the limit;
- Return an `int` containing the sum; and
- Throw an `IllegalArgumentException` for an argument strictly smaller than 0.
WeBWorK in CS

set1: Problem 1

public static int sumEven(int n) { if (n >= 0) { int res = 0; for (int i = 0; i <= n; i++) { if (i % 2 == 0) res = res + i; } return res; } throw new IllegalArgumentException("Argument "+ n + ", not in range"); }

The above answer is correct.

SUM OF EVEN NUMBERS

Write a method that computes the sum of the even numbers from 0 to a given limit (included).

The method will be called sumEven and must:
- Be declared public and static;
- Take one parameter of type int representing the limit;
- Return an int containing the sum; and
- Throw an IllegalArgumentException for an argument strictly smaller than 0.

public static int sumEven(int n) {
    if (n >= 0) {
        int res = 0;
        for (int i = 0; i <= n; i++) {
            if (i % 2 == 0)
                res = res + i;
        }
        return res;
    } else
        throw new IllegalArgumentException("Argument "+ n + ", not in range");
}
set1: Problem 1

public static int sumEven(int n) {
    if (n >= 0) {
        int res = 0;
        for (int i = 0; i <= n; i++)
            if (i % 2 == 0) res = res + i;
        throw new
        IllegalArgument Exception("Argument " + n + " not in range");
    }
}

The above answer is NOT correct.

SUM OF EVEN NUMBERS

Write a method that computes the sum of the even numbers from 0 to a given limit (included).

The method will be called sumEven and must:
- Be declared public and static;
- Take one parameter of type int representing the limit;
- Return an int containing the sum; and
- Throw an IllegalArgumentException for an argument strictly smaller than 0.

public static int sumEven(int n) {
WeBWorK in CS

set1: Problem 1

public static int sumEven(int n) { int res = 0; for (int i = 0; i <= n; i++) { if (i % 2 == 0) res = res + i; } return res; }

The above answer is NOT completely correct.

**SUM OF EVEN NUMBERS**

Write a method that computes the sum of the even numbers from 0 to a given limit (included).

The method will be called `sumEven` and must:
- Be declared public and static;
- Take one parameter of type `int` representing the limit;
- Return an `int` containing the sum; and
- Throw an `IllegalArgumentException` for an argument strictly smaller than 0.

```java
public static int sumEven(int n) {
    int res = 0;
    return res;
}
```
DOCUMENT();
loadMacros(
    "PG.pl",
    "PGbasicmacros.pl",
    "PGchoicemacros.pl",
    "PGanswermacros.pl",
    "PGauxiliaryFunctions.pl",
    "javaAnswerEvaluators.pl"
);
BEGIN_TEXT
# Specification of the problem
\{ANS_BOX(1,1,30);\}
END_TEXT
ANS(java_cmp("directoryname","classname"));
ENDDOCUMENT();
public class Factorial {

    public static int myfactorial(int n) {
        if (n <= 12 && n > 0) {
            return n * myfactorial(n - 1);
        } else if (n == 0) {
            return 1;
        }
        throw new IllegalArgumentException("Argument "+n+" not in range");
    }

    // Factorial method to be entered by the user
    replaceme

}
import java.lang.reflect.*;
import junit.framework.*;

public class FactorialJUnitTest extends TestCase {
    private boolean existsFactorial, isStatic, returnType, paramType;

    // FactorialJUnitTest, setUp, tearDown
    public void testMethodSignature() {
        Assert.assertEquals("Signature problems", true, existsFactorial && isStatic && returnType && paramType);
    }

    public void testFactorial3() {
        try {
            assertEquals(6, Factorial.factorial(3));
        } catch (Exception e) {
            fail("Fail - n = 3");
        }
    }

    public void testFactorial-4() {
        try {
            Factorial.factorial(-4);
            fail("Fail - n = -4");
        } catch (Exception e) {
            if (e instanceof IllegalArgumentException)
                assertTrue(true);
            else
                fail("Fail - n = -4");
        }
    }
}
Use of WeBWorK – Students’ Results

• Use of WeBWorK in two graded homeworks to review topics on loops, arrays and recursion
  – HW1: Multiple-choice and short answer type questions
  – HW2: WeBWorK-JAG questions

• Students’ results:
  – HW1: High number of attempts for questions on recursion and with a large number of possible answers
  – HW2: Average number of attempts per WeBWorK-JAG question was 20
Use of WeBWorK – Students’ Feedback

- Specification and presentation of the questions
  - Long questions restrict visibility and rely on short-term memory
  - WeBWorK-JAG question specifications require a large amount of setup information that confuses students

- IDE-like environment
  - Entering code into a text box area does not provide the efficient features of the Eclipse IDE editor
Use of WeBWorK – Students’ Feedback

• Lack of feedback in multiple-choice and short answer questions
  – Especially for problems composed of numerous questions where the wrong answers are not pinpointed

• Lack of feedback in WeBWorK-JAG questions
  – Additional feedback information for compilation errors and red/green failure/success indication were proposed as extensions to WeBWorK-JAG
  – Students are interested in a very fine level of granularity in WeBWorK-JAG failure messages (number of successful and failed tests, exact/similar data test, hints)
Contributing to WeBWorK

- Students' contributions to WeBWorK
  - Multiple-choice and short answer questions on Java 1.5 features
  - WeBWorK-JAG questions
- Contributed questions were peer-reviewed and tested by students, and then integrated in the WeBWorK library of problems by the instructors
WeBWorK in CS

Contributing to WeBWorK – Students’ Results

• WeBWorK-JAG questions
  – Poor formulation of the questions (scope not clear, modifiers not stated, requirement for the use of a specific algorithm)
  – Well-written test cases for method signatures
  – Not exhaustive test cases for the method (null object, equivalence class identification, exceptions, invalid inputs, no white box testing)
  – Coarse or inexistent feedback for failure test cases
  – Crucial role of QA to catch problems
WeBWorK in CS

Contributing to WeBWorK – Students’ Feedback

• When specifying the question students had the user in mind (not so much when writing the tests)
• Students became familiar with the use of the Reflection API
• Writing WeBWorK-JAG questions forced students to think about testing first
• Students improved their testing skills (including regression testing)
Conclusions and Future Work

• Development of the WeBWorK-JAG extension
• Development of a novel pedagogy encouraging students to contribute their own questions to the system WeBWorK library and introducing them to crucial practices of software engineering

• Add more support in WeBWorK for programming problems (IDE-like environment, presentation of Java code in questions)
• Add more granular and visual feedback on performance for students and instructors

• Need of an open-source web-based assessment system for programming assignments
• Create a community of contributors to monitor quality, share work and extend the WeBWorK library
WeBWorK in CS

Acknowledgements

• NSF CCLI AI Grants “Collaborative Research: Adapting and Extending WeBWorK for Use in the Computer Science Curriculum” #0511385 and #0511391

• Students:
  – The 19 students of CS2
  – Jacqueline Baldwin, Nathan Baur (JUnit extension)
  – Sophal Chiv (inputing problems and help desk)
  – Eileen Crupi, Tabitha Estrellado (inputing problems)
  – Allyson Ortiz, Veronica Portas (existing systems)
  – Yue Ma (testing)