WeBWorK in CS

Teaching Software Quality by Encouraging Contributions 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 Wildenberg
Cornell College, Iowa, USA

Cornell College

PACE UNIVERSITY

Outline

- Pedagogical Context
- Systems for Automated Assessment of Programming Assignments
- WeBWorK
- WeBWorK-JAG
- Process of Contributing to WeBWorK
- Results and lessons learned
- Conclusions and Future Work

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

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

WeBWorK in CS

Existing Systems

- Boss *www.dcs.warwick.ac.uk/boss*
- CodeLab *www.turingscraft.com*
- CourseMarker *www.cs.nott.ac.uk/CourseMarker*
- DevSquare *www.devsquare.com*
- Gradiance *www.gradiance.com*
- JavaBat *www.javabat.net*

- MyCodeMate *www.mycodemate.com*

- OWL *owl.course.com*

- Viope *www.viope.com*

- WebCAT

http://zing.ncsl.nist.gov/WebTools/WebCAT/overview.html

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

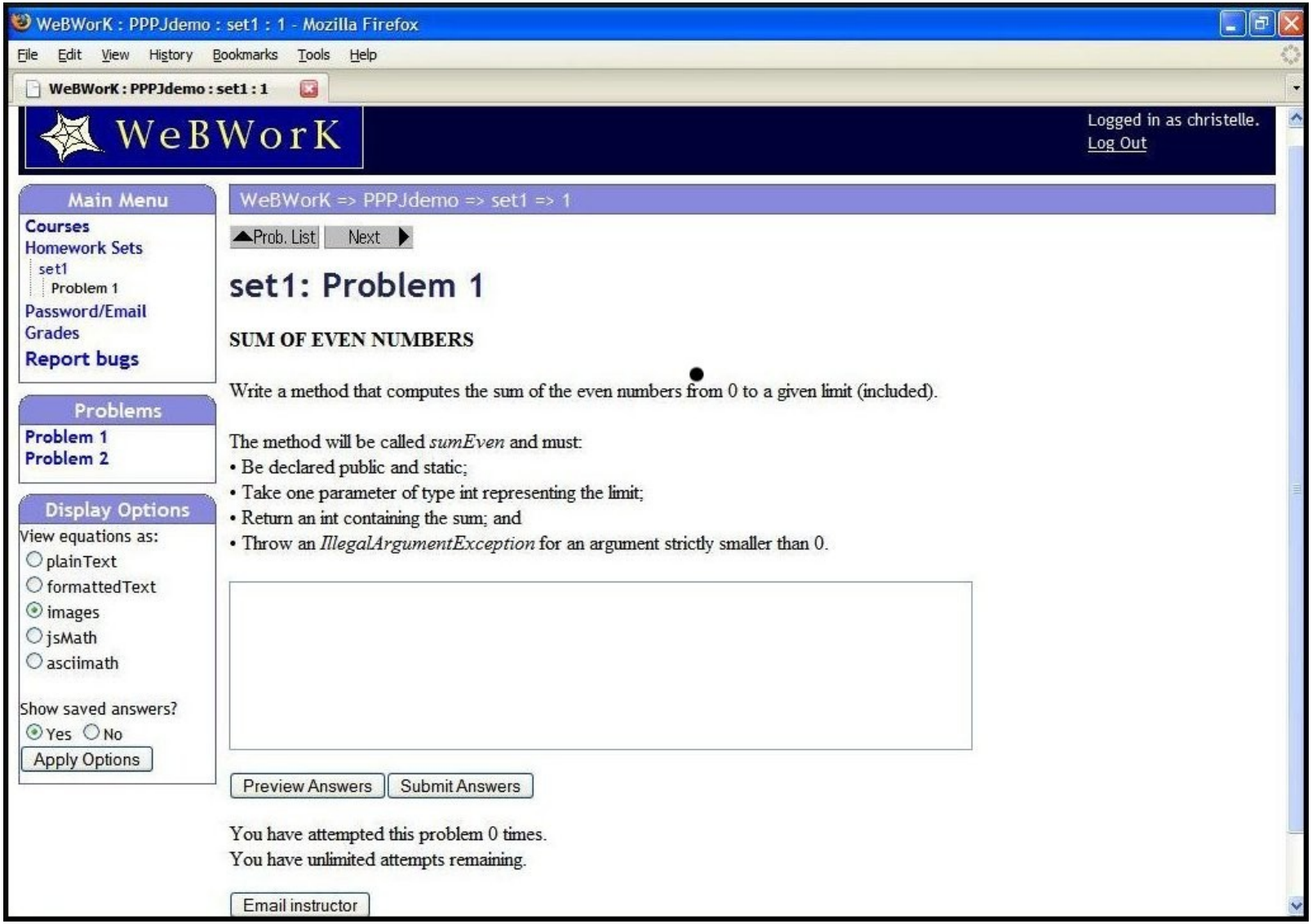
WeBWorK

- 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 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

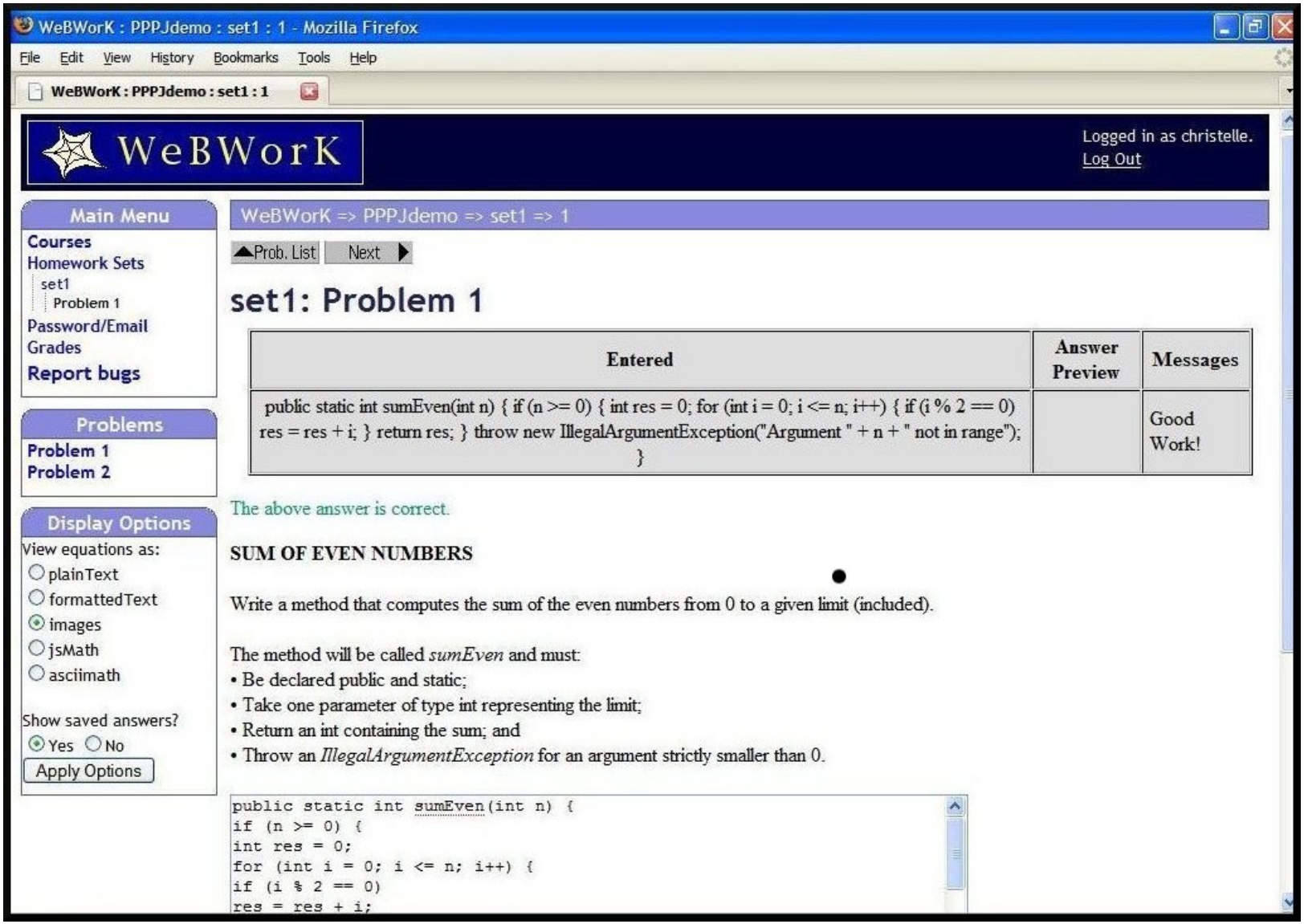
WeBWorK in CS



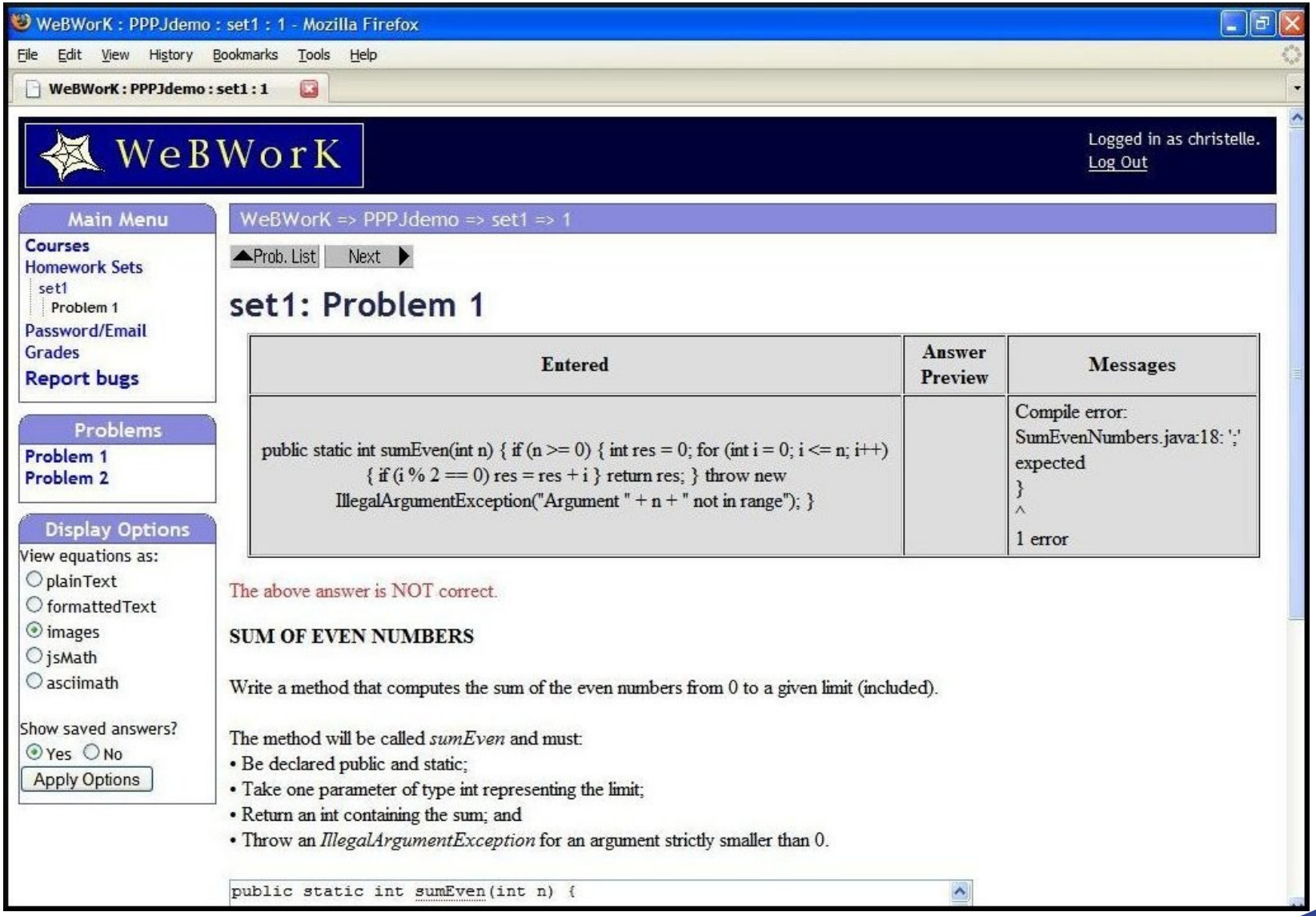
Cornell College

PACE UNIVERSITY

WeBWorK in CS



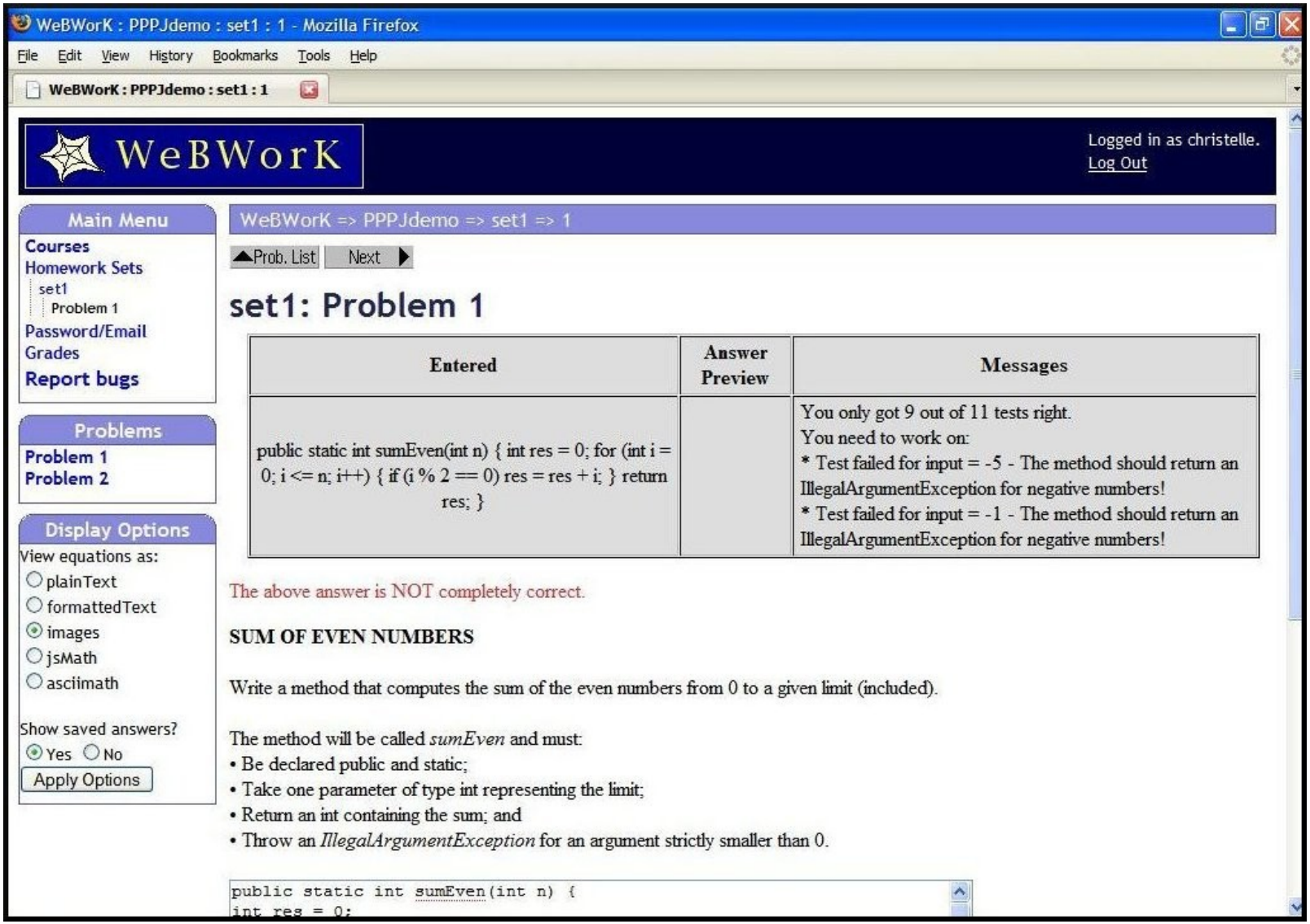
WeBWorK in CS



Cornell College

PACE UNIVERSITY

WeBWorK in CS



Cornell College

PACE UNIVERSITY

Process of Contributing to WeBWorK

1.Formulation of the problem (Requirements);

2.Peer-review of the problem formulation (Requirements Inspection and Validation -- SQA);

3.Design of unit tests (Requirements Refinement and Testing);

4.Peer-review of the unit tests (Test Case Inspection and Verification -- SQA);

5.Integration of the problem with its test cases into the Web-based system (Deployment); and

6.Testing of the problem and feedback by users (User Acceptance Testing -- SQA).

Templates for Problem Formulation

- **Description.** A short description of the method to be written.
- **Method name.** The name of the method.
- **Method signature description.** A description of the method signature in terms of:
- Modifier, i.e. either static, public, protected, private or package;
- Type of the method, i.e. either static or instance;
- Number and type of parameters; and
- Return type.
- **Exceptions.** A description of the exceptions to be thrown along with the cases in which they are thrown.
- **Code.** Code provided to support writing the method.
- **Notes.** Particular restrictions concerning the method to be written.

Example – Problem Formulation

- Write a method that computes the factorial of a given number.
- The method will be called *factorial* and must:
  - Be *public* and *static*;
  - Take an *int* as a parameter;
  - Return the factorial of that *int* as an *int*; and
  - Throws an *IllegalArgumentException* for a negative input or an input that would not return a Java *int*.

Example – Expected Solution

```
public class Factorial {

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

}
```

Example - JUnit Code

```
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.assertTrue("Signature problems", 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");
}
```

Example – Integration - PG Code

```
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();
```

Example – Integration - Java Class

```
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

}
```

Contributing to WeBWorK – Students' Results

- CS1-level contributed questions
- When specifying the question students had the user in mind (not so much when writing the tests)
- Poor formulation of the questions (scope not clear, modifiers not stated, requirement for the use of a specific algorithm)
- Eclipse facilitated writing unit test cases
- 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

Contributing to WeBWorK – Lessons Learned

- Writing JUnit tests is programming
- Crucial role of QA to catch problems
- Students are proud to contribute to an open-source system that other students will have access to
- Did we manage to augment our library of programming questions? Yes and No
- Adaptivity of our teaching model outside of the WeBWorK context

Conclusions and Future Work

- 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
- Repeat the experiment and go through the complete process
- Create a community of contributors to monitor quality, share work and extend the WeBWorK library

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 (user acceptance testing)