Assessment Rubric

Semester

Instructor/Course/Section/Number of Students

Program Outcome/Competency Level Expected

BSCS.OC8/Beginning Level

The ability to write programs that demonstrate comprehensive knowledge of object-oriented programming

Supported by: CS46A

Instructions

For each task below:

1. Describe one or more problems of average difficulty that tested the student's ability to do the task. This might be an exam question, a lab, or an assignment. If the task was covered in lecture, but not in any problem, write "covered but not tested" in the space below. If the task was not covered at all, write "not covered" in the space below.

2. For each problem above, indicate the percentage of passing students who successfully solved the problem. Passing means the student received a C- or better in your course. Success means that the student received full or near-full credit.

Here you go. My editorial comments are in italics.

Tasks

Task 1: Building complex control structures

Write a Java program that employs nested control structures, for example, a program that counts the number of entries in a two dimensional array having certain properties.

Problem: Final Exam problem #8

29% (8/28)

This was a nontrivial problem. When I saw this rubric towards the end of the semester, I said “holy cow, I better run a lab training students to do well on this kind of problems.” Watching the students, I realized that one lab isn't nearly enough. We would need to make an organized effort teaching students how to solve algorithmic problems. This goes
well beyond being able to work with loops and arrays. Students would need to learn how to generate ideas, how to evaluate them, and how to translate them into working code.

**Task 2: Array Processing**

Write a Java program that modifies an array, for example, a program that sorts an array.

Problem: Final Exam problem #7

46% (13/28)

This problem wasn't quite as hard as #8, but quite a bit harder than the kind of array manipulation problems that we usually do in CS1 (i.e. linear traversal, only trivial modifications). The rubric and the course are not in synch.

**Task 3: Implementing a class**

Follow a methodology to implement a class from a specification. For example, given a test harness for a class, determine the signatures of the public methods, write javadoc comments for the public methods, determine the needed fields, implement the methods, and run the test harness.

Problem: Final Exam problem #1

82% (23/28)

This would have been nearly 100% if I had counted the number of people who actually took the exam. Several people didn't take the final and still managed to eke out a passing grade.

**Task 4: I/O Processing**

Write a Java program that employs input/output, for example, a program that extracts information from a text file.

Problem: Final Exam problem #6

25% (7/28)

Unfortunately, this problem had an algorithmic twist—printing the last two lines was definitely beyond the problem solving abilities of most students.
Task 5: Processing polymorphic collections

Write a Java program that processes heterogeneous objects and that requires an understanding of dynamic method invocation. For example, write a program that totals objects that implement a Buyable interface.

Problem: Final Exam problem #2

18% (5/28)

This didn't get much class time—one lab. Blame the furloughs.

We've got to pick our battle here. Right now, CS46A is a shallow introduction into lots of topics. If we want that, then we should be satisfied with students who can solve easy programming tasks and memorize basic concepts. If we want them to show a deep understanding and solve nontrivial problems, then we should move topics to CS46B (such as I/O and inheritance) and really teach them problem-solving strategies.

On the one hand, I like the idea of teaching problem solving. But then again, this is not something that any textbook covers in a systematic way—maybe because no author ever tried, but also quite possibly because it would overwhelm a significant number of beginning students.

The easy way out is to reword this rubric so that we are satisfied with simple loops and simple array processing. We should also get a more realistic task for demonstrating an understanding inheritance, such as forming a subclass that overrides a method from a superclass.