# SE1011 Exam 1 Feedback

**Dr. Yoder, Fall 2014, MSOE**

## Problem 1

① The long is cast to a float, not the other way around. In the assignment statement `float f = l;` the number on the right, "l" needs to be converted to the type of the variable that it is being assigned to, in this case a float.

② The problem had an error in the problem statement. It should have read `int i = (int)d;`. If you interpreted the problem this way, I gave you full credit. If you didn't, I also gave you full credit if you correctly discussed that a double cannot be cast to an int.

③ -2 Although there can be loss of less significant digits when casting a long to a float, this is a legal implicit cast because it is from an integer type to a floating-point (decimal-part) type.

④ -0.5 Explanation seems to be going in the right direction, but is missing key information. For example, the explanation should specify that explicit casting is always legal.

⑤ -2 Explanation focuses on the wrong line of the example – the line where the variable is assigned, ignoring the line with the cast.

⑥ -2 Part b: No explanation given (with correct answer)

## Problem 4

① -2 This answer swaps the role of the Java Virtual Machine with the Java Compiler, or says that the Java Virtual Machine is responsible for reading the .java source file. (-1 if ② (an error) is also made).

② -2 This answer clearly swaps the roles of the .java and .class files. A .java file contains ***source*** code – plain-text code written by the programmer.

④ -0 Explanation seems to be going in the right direction, but is missing key information. For example, the answer should distinguish between a ***plain text source*** of a program and a ***compiled byte-code*** version of the same program.

## Problem 6

① -0.5 The program specified to accept an integer from the user. When counting, using integers is important because exact comparisons with floating-point numbers can be tricky. For example, this program never terminates:

```
double d = 0;
while(d != 0.3) {
   d += 0.1;
}
```

(Example adapted from http://stackoverflow.com/questions/249467/what-is-a-simple-example-of-floating-point-rounding-error)

② -0.5 Use `in.nextInt()` to get an integer. The parentheses are required, because nextInt is a method (that is, a behavior of the object "in" points to), and all methods require parenthesis in Java. in.next() gives a string holding the next word the user types.

③ -2 Wrong number of "going ups" printed – one extra (or perhaps one less).

⑤ -4 No loop! You should print "going_up" multiple times, which requires a loop. You may want to come up with an independent exercise to demonstrate that you understand the material on this problem.

## Problem 7

① -2 Need to ask for input every time through the loop. Otherwise, user can only compute one tax value, and just keeps seeing it over and over again until he or she quits.

② -7 "while" should be used to create a loop so that the program allows the user to enter multiple salaries. Using "if" will not loop.

③ -2 "exit" and "continue" are the conditions to decide whether to go through the loop body (either the first time, or again), ***based on what the user input***. I mark this this way if neither your code nor your fives steps show that you know this.

## Problem 8

① -1 "good" and "bad" reversed. Oops!

② When using two if statements in a row, it is often (but not always) better to use a single if statement with an else clause.

For example,

```
if (x>5) {
  //do A
}
if (x<=5) {
  //do B
}
```

is better written

```
if (x>5) {
  //do A
}
else {
  //do B
}
```

This makes it clear that you only want one of the two blocks to execute, and avoids something inside A changing x so that both A & B execute. For some programs, you want both A and B to execute. Suppose you were writing a program to decide how many items to pack into a box, and and you wrote if statements to pack the largest boxes first (if needed). In that case, you might want to pack the big boxes and then pack the medium boxes with the modified value of itemsLeft. In this case, you would use two if statements

```
if(itemsLeft >= 10) {
  itemsLeft = itemsLeft % 10;
}
if(itemsLeft >=5) {
  itemsLeft = itemsLeft  % 5;
}
```
(This example is somewhat contrived – you could just compute the modulus without the ifs and it would still work.)

-0 – Code works correctly, but has poor style.

-2 – Code will handle some cases incorrectly. e.g. above, if the second condition was x<5, the x==5 case would not be handled.

-2 – Code handles many cases incorrectly. For example, if the second condition was x>1, and x had the value 2, the code would do both A and B when it should only do B.

③ -2 – && instead of || (or || instead of &&) causes wrong range to be used.

④ -2 Nested ifs instead of combining conditions with &&, results in some cases not being handled.

⑤ -1 Assignment missing.