CSc 227 — Program Design and Development
(McCann)

Sample First Midterm Exam — The Answers

Background and Advice

Former students have suggested that their performance on the first midterm exam of a class would have been better had sample exam questions been available, the idea being that they would have come into that first exam with less anxiety. If having such a preparation aid helps put students in the proper frame of mind to take the first exam, I'm willing to provide a sample exam.

I suggest that you treat this as if it were the actual exam. That is, do not look at it until you are ready to do it, and do it in the same setting as we use for midterms: A time period of 75 minutes with no book, no notes, and no calculator. Only when you are finished (or out of time!) should you download and look at the sample answers, and compare them to your own.

Please remember: The following is a *sample* set of questions. On the actual exam, you will see different questions, on a different subset of topics, phrased and structured in different ways. If you are expecting the actual exam to be only a slight variation of this sample, you will be disappointed.

Finally: **This is the only sample exam I will provide this semester.** After seeing this (and the real exam), you'll have a very good idea of how future exams will be structured.

1. (10 points) Consider the following sequence of Java statements, augmented with imaginary line numbers for convenience:

```
[0]    StringBuilder message = new StringBuilder("227 is instructive");
[1]    message.insert(0,"C SC ");    // each gap is one space
[2]    System.out.println(message);
[3]    message.delete(14,15);
[4]    message.replace(15,21,"ens");
[5]    System.out.println(message);
```

`insert(i,s)` inserts the characters of s into the object starting at index i (and the existing characters are moved down to make room), `delete(i,j)` removes the characters from index i through index j-1, and `replace(i,j,s)` is `delete` plus the insertion of s.

(a) What is the output of the statement at line 2?

```
C SC 227 is instructive
```

(b) What is the output of the statement at line 5?

```
C SC 227 is intense
```

(c) Append additional statements to this code that further modify the content of `message` so that, if the result of your modification were displayed, the output will be:

```
CSc 227 is making me tense!
```

Here's one way to do it (the meaning of `append()` should be clear):

```
message.delete(1,2);
message.replace(2,3,"c");
message.replace(11,13,"making me ");
message.append("!");
```

2. (12 points) *Really* short answer questions.

   (a) Parameters to Java methods are pass-by-`__?__`. value

   (b) Which parts of a method header are also found in a method signature?

   The method's name and formal parameters

   (c) Name exactly three attributes of a Java variable.

   Possibilities include: name, address, type, scope, ...

   (d) When multiplying an int value and a float value, what will be the type of the result? float

   (e) What are the result type **and** result of the evaluation of this expression: `(short) 5.6 * 2`

   Result = 10, Result type = int

3. (8 points) Consider this method header from the `StringBuilder` class:

```
public StringBuilder insert(int offset, char[] str)
```

   (a) Briefly explain the purposes of each of the first three words of this header.

   `public` – Indicates that the method is accessible outside of the class in which it is declared.

   `StringBuilder` – The method will return a reference to an object instantiated from this class.

   `insert` – The name of the method.

   (b) The keyword `static` is missing from this header. What does its absence tell us?

   Without 'static', the method is an instance method, accessible only through a String-Builder object rather than the StringBuilder class.

4. (3 points) What is the result of the evaluation of this familiar expression, when all variables are of type `int`, $c = 10, y = 66, monthCode = 5$, and $d = 31$: $\left(\left\lfloor \frac{5y}{4} \right\rfloor + monthCode + d - 2(c \,\%\, 4) + 7\right) \,\%\, 7$

$\left(\left\lfloor \frac{5 \cdot 66}{4} \right\rfloor + 5 + 31 - 2(10 \,\%\, 4) + 7\right) \,\%\, 7$
$= (110 + 36 - 4 + 7)\%7$
$= 149\%7 = 2$

5. (4 points) Two characteristics of an operator are precedence and associativity. Name a Java operator that has right to left associativity, and explain why it can't have left to right associativity.

Examples include the assignment operators and unary negation.

Take unary negation as the example. Without R-to-L associativity, in the statement `--a` we would start by trying to negate the second unary negation operator, which makes no sense.

6. (6 points) We know that Java uses the leftmost (high-order) bit of its binary representation of integers to stand for the sign of the integer value. Consider a language that uses 4-bit integers. The bit pattern '0000' represents the integer value 0, '0001' is +1, etc. Which integer values do the patterns '0111', '1000' and '1111' represent?

$0111_2 = 7_{10}$, $1000_2 = -8_{10}$, and $1111_2 = -1_{10}$.

7. (5 points) What is the output of this section of Java code, and what are the final values of the variables?

```
int x=0, y;
y = ++x - 1;
System.out.println(y==x--);
```

Output	x	y
false	0	0

8. (2 points) Consider this segment of Java code:

```
Character c1, c2;
c1 = new Character('A');
c2 = c1;
```

If we compare `c1` and `c2` with the '==' operator, what is being compared?

The references (addresses) held by the two reference variables, NOT the characters held within the Character wrapper objects.

9. (3 points) In general, what is the purpose of a Java class?

A class serves as an object factory, defining state and operation information that is associated with new objects instantiated from it. Alternatively, a class can serve as a collection of related general-purpose methods (e.g., the `Math` class).

10. (3 points) Define: **Instantiable**.

Instantiable classes are those with publicly accessible constructors; that is, classes from which objects can be created.

11. (8 points) Write a `do-while` loop that places the characters containing in the String `str` into a 1-D array named `aray`, in reverse order. Start by providing the declaration of the array of characters, but don't allocate the array until you need to. Do not execute the loop if the string contains no characters.

```
+--------------------------------------------------------------
|  char [] aray = null;      // don't allocate array storage yet
|
|  if (str.length() >= 1)
|      aray = new char[str.length()];   // ok, now allocate array
|      int i = str.length()-1;          // start at back of string
|      int j = 0;                       // start at front of array
|      do {
|          aray[j] = str.charAt(i);
|          i--;  j++;
|      } while (i >= 0);
|  }
+--------------------------------------------------------------
```

12. (4 points) Java isn't completely object-oriented. Give one example, and explain how your example shows that Java isn't a true OO language.

> Any of the base types (`int`, `double`, etc.) are good examples. These variables do not contain references to objects containing values, they contain the values. They also do not have methods.

13. (3 points) Explain the purpose of the keyword '`this`' in Java.

> Within an instance method, 'this' is a reference to the object on whose behalf the method is being executed. When an object's method is being executed, and that method would like to refer to that object's own instance variable, for example, we can prefix "this." to the variable's name. See example T01n18.java for a demonstration.

14. (12 points) Write a new public void method for `WordHelper` called `shuffle`. Let's assume that a WordHelper object's instance data is a String object named 'word.' `shuffle` treats the string as if it were a deck of cards, performing one interleaving of the characters, and setting the object's word to be the newly rearranged word. To perform the interleaving, divide (or just simulate dividing, your choice) the string into two halves. Alternating between the halves, add one character at a time to the resulting string. If word's length is not even, place the last character at the front of the shuffled string. This 11-character example should help you understand what needs to be done:

```
ABCDEFGHIJK  -->  ABCDE FGHIJ K  -->  AFBGCHDIEJ K  -->  KAFBGCHDIEJ
```

```
+--------------------------------------------------------------
|  public void shuffle ()
|  {
|      int len = word.length();
|      int first = 0, second = len/2;       // works for odds and evens
|      String result = new String();
|
|      for (int i=0; i < len/2; i++) {
|          result += word.charAt(first);
|          result += word.charAt(second);
|          first++;  second++;
|      }
|
|      if (len % 2 == 1) {                  // prepend the last if un-paired
|          result = word.charAt(len-1) + result;
|      }
|
|     word = result;
|  }
+--------------------------------------------------------------
```