CIS 110 — Introduction To Computer Programming

October 5th, 2011 — Exam 1

Review problems

Scores:

1	
2	
3	
4	
5	
6	
Total (100 max)	

CIS 110 Exam 1 Instructions

- You have 50 minutes to finish this exam. Time will begin when called by a proctor and end precisely 50 minutes after that time. If you continue writing after the time is called, you will receive a zero for the exam.
- This exam is *closed-book, closed-notes, and closed-computational devices* except for a one page sheet (8.5" by 11") of double-sided notes.
- When writing code, the only abbreviations you may use are for `System.out.println` and `System.out.print` as follows:

  $$\text{System.out.println} \longrightarrow \text{S.O.PLN}$$
  $$\text{System.out.print} \longrightarrow \text{S.O.P}$$

  Otherwise all code must be written out as normal.
- Please do not separate the pages of the exam. If a page becomes loose, please make sure to write your name on it so that we don't lose it, and use the provided staplers to reattach the sheet when you turn in your exam.
- If you require extra paper, please use the backs of the exam pages or the extra sheet paper provided at the end of the exam. Clearly indicate on question page where the graders can find the remainder of your work (e.g., "back of page" or "on extra sheet").
- If you have any questions, please raise your hand and an exam proctor will come to answer them.
- When you turn in your exam, you will be required to show ID. If you forgot to bring your ID, please talk to an exam proctor immediately.

Good luck, have fun!

CIS 110 Exam 1 Cheat Sheet

```
/** 1. For syntax, look at the format of the code presented in the questions
 *     (unless otherwise stated, it is all syntactically correct code). */

// class declarations       // method declarations
public class <name> {       public static <type> <name>(<params>) {
    <methods>                   <statements>
}                           }

/** 2. Useful methods for String objects */

charAt(index)           // Returns the character at the given (zero-based) index.
endsWith(text)          // Returns true if the string ends with the given text.
indexOf(character)      // Returns the (zero-based) index of the given character.
length()                // Returns the length of the string.
startsWith(text)        // Returns true if the string starts with the given text.
substring(start, stop)  // Returns the characters from the start index to just
                        //   before the stop index
toLowerCase()           // Returns a new string with all lower case characters.
toUpperCase()           // Returns a new string with all upper case characters.

/** 3. Useful methods for Graphics objects */

drawLine(x1, y1, x2, y2)        // Draws a line between (x1, y1) and (x2, y2)
drawOval(x, y, width, height)   // Draws the outline of the largest oval that
                                //   fits in the specified rectangle.
drawRect(x, y, width, height)   // Draws the outline of the specified rectangle.
drawString(msg, x, y)           // Draws the given text with its lower-left
                                //   corner at (x, y)
fillOval(x, y, width, height)   // Fills the largest oval that fits in the
                                //   specified rectangle.
fillRect(x, y, width, height)   // Fills the outline of the given rectangle.
setColor(color)                 // Sets the graphics context to use the color.
```

Notes about this exam:

- Exam #1 covers all content covered so far in class from lectures, labs, homeworks, and the text (chapters 1–3 including 3G).
- The purpose of this exam is to *test your algorithmic thinking skills* rather than have you regurgitate facts about computer programming. To prepare for this exam, you should practice those skills by reviewing homeworks and doing practice problems discussed in section or found in the text.
- The exam is *closed book*, *closed notes*, and *closed electronic devices*. You may bring a single 8.5×11" sheet of double-sided notes to help jog your memory.
- In addition to your note sheet, we will include a sheet of commonly-used APIs so that you do not have to write them down. This cheat sheet appeared on the previous page.
- This practice exam introduces you to the types of problems that will be on the real exam. That way you can spend more time solving problems instead of trying to understand what the problem asks of you. That being said, while my goal is not to stray far from this format, the format of the actual exam may change slightly if necessary.
- I strongly recommend that you attempt this practice exam without looking at the answers. Afterwards, check your work, and then use the usual channels (e.g., Piazza, your TA, or myself) to resolve any lingering questions that you may have.

Expression Evaluating Machines

1. (10 points) Evaluate these Java expressions to their final values.

   (a) `10 * 3 + 4 / 2` $\longrightarrow$

   (b) `3 / (double) 2 + 5 / 7` $\longrightarrow$

   (c) `12 % 8 - 3 % 4` $\longrightarrow$

   (d) `"super" + 3 * 3 + "duper" + (8 - 5)` $\longrightarrow$

   (e)
   ```
   String s = "hello world!";
   s.substring(2, 3) + ":  " + (s.length() + 1)
   ```
   $\longrightarrow$

Trace You A Loop

2. (10 points) Trace the execution of the following loops by filling in the provided tables. Each row of the table corresponds to the state of the program after executing the line marked `HERE`. You may skip any cases of the variables that do not reach execution of the line marked `HERE`.

(a)
```
for (int i = 0; i < 8; i += 2) {
    for (int j = 0; j < i / 2; j++) {
        System.out.println("*"); // HERE
    }
}
```

i	j	Output (each line)

(b)
```
for (int i = -1; i < 1; i++) {
    for (int j = 2; j > i; j--) {
        for (int k = 0; k < 1; k++) {
            System.out.println((i + j) * k);  // HERE
        }
    }
}
```

i	j	k	Output (each line)

Method Mystery

3. (15 points) Write the output of this program in the space below.

```
public static void f(int x, String y, int z) {
    System.out.println(y + " ; " + z + " ; " + x);
}

public static int g(int n) {
    n--;
    n *= n + 2;
    return n;
}

public static void main(String[] args) {
    int x = 0;
    String y = "forty-two";
    int z = 1;
    f(z, y, x);
    f(x, "zero", z);
    f(z, y, z);
    f(x, "hello", g(5));
    f(g(1), "bye", g(g(2)));
}
```

Patterns, Patterns, Patterns

4. (20 points) Consider the following ASCII picture:

```
      *-
    +=*--
  +=+=*---
+=+=+=*----
```

(a) What is the form of each line (i.e., what is the pattern of characters on each line)?

_____

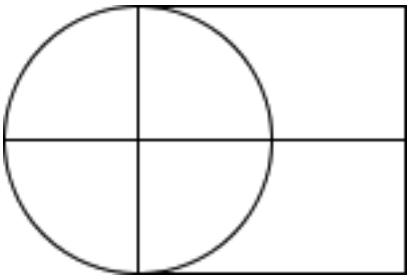
(b) For each chunk of characters you identify, give a formula for the number of chunks on each line in terms of the row number `i`. You may choose to number the first row either 0 or 1.

(c) Finally, write a method, `drawShape()` that draws the above diagram in terms of the formulae that you derived.

Parameterization

5. (20 points)

(a) Write a method, `drawFigure`, that takes a `Graphics` object as input and draws the following figure to that `Graphics` object.



The figure consists of three parts:

- A *circle* of radius 50. Its anchor point is (0, 0).
- A *square* of width 100. Its anchor point is (50, 0).
- A *horizontal line* of length 150. Its left endpoint is at (0, 50).

(b) Parameterize `drawFigure` further so that you can draw the figure anywhere on the screen. The figure should remain the same size, but the caller of `drawFigure` should be able to pass in an (x, y) pair to move the anchor of the figure (which is its upper left-hand corner). You may produce the new `drawFigure` method below or simply change your answer to the previous part to become parameterized.

Gonna Fly Now

6. (25 points) Write a method `sillyPrint` that takes a parameter n and writes out the numbers from 1 to n with the following format:

```
][1][][2][][][3][ ... ][n][
```

Where the numbers are surrounded by square brackets in a pattern. For reference, here is the execution of `sillyPrint(7)`.

```
][1][][2][][][3][][][][4][][][][][5][][][][][][6][][][][][][][7][
```

Hint: Decompose the problem, writing a separate, well-named method, for each subproblem. And then combine all those methods in the definition of `sillyPrint` to arrive at the final result.