CPSC 124-02, Fall 2021
Information on the First Test

The first test for this course takes place in class on Friday, September 24. It will cover material from classes and labs as well as from assigned readings. The test will be four pages long. It can include definitions and short answer questions; longer essay-type questions; questions that ask you to write code segments or possibly a complete program; and questions that ask you to determine the purpose or the output of a given code segment. For definitions, you might be asked about basic terms such as *CPU*, *main memory*, *machine language*, *binary number*, *high-level language*, *compiler*, *algorithm*, *pseudocode*, *infinite loop*, *literal*, *variable*, *type*, *subroutine*, *control structure*, *syntax*, *semantics*, *style*, and so on.

The test covers parts but not all of Chapters 1, 2, and 3 from the textbook. From Chapter 1, we covered only some basic concepts, from Sections 1.1 and 1.3. (We also covered all of the concepts in Section 1.4 later.) We covered large parts of Chapter 2. We omitted 2.3.4, 2.3.5, 2.4,4, 2.4.5, and 2.4.6. We went through the details of expressions in Section 2.5 quickly, and you should pay attention only to the operators that are in the list below. We did not do anything with Section 2.6. In Chapter 3, we covered almost all of Sections 3.1 through 3.6, but remember that you will not be asked to write `do..while` or *switch* statements.

You are not responsible for every detail of the reading. For example, you are not responsible for memorizing every subroutine defined in *Math*, *TextIO*, and *String*; see the lists, below, of subroutines for which you are responsible. Similarly, see the list below for graphics subroutines that you should be able to use. You will not be asked about specific examples that are covered in the text, such as the *3N+1* problem or interest rate computations. There will be no questions on Linux or the command-line environment, except possibly for the *javac* and *java* commands and the general process of editing, compiling, and running a program.

Here is a list of some of the things that you should know about:

CPU
main memory (RAM) and how it is organized
computer programs
machine language
high-level programming language
compiler
Java Virtual Machine
how to compile and run a Java program: The *javac* and *java* commands

style rules and why they are important
style rules for variable names and class names
formatting and indentation of programs
comments and why they are used
comments using //
comments using /* ... */
the basic structure of a Java program (**You should have it memorized by now!!!**):

```
    public class <program-name> {
        public static void main(String[] args) {
           <statements>
        }
    }
```

syntax and semantics
syntax errors (found by compiler) vs. semantic errors (seen at run time)

identifiers
compound names (with ".") such as System.out.print or str.equals
variables
data types
Java's primitive types: int, double, char, boolean
the String type (NOT a primitive type!)
literals

Java literals of various types (numeric, char, String, boolean)
special characters in char and String literals, such as '\n' and '\t'
the basic idea of binary numbers (strings of 0's and 1's)
assignment statements
type compatibility rules for assignment statements
declaring variables
subroutines, and "calling" a subroutine such as System.out.print or g.fillRect
parameters in subroutines, such as the x in System.out.println(x)
functions: subroutines that "return a value"

important functions in the Math class:
      Math.random(), Math.sqrt(x), Math.pow(x,y)

important functions in any String str:
      str.length(), str.charAt(i), str.indexOf(ch), str.equals(s)

comparing Strings with str.equals(str2) instead of str == str2
System.out.print(x) and System.out.println(x)
formatted output with System.out.printf(formatString, v1 v2, ...)
important format specifiers for use in the formatString of printf:
      for example: %s   %d   %10d   %1.2f
getting input from the user
TextIO; important TextIO input functions:
      TextIO.getln(), TextIO.getlnInt(), TextIO.getlnDouble()
expressions
operators
precedence of operators and using parentheses to control order of evaluation

important operators:
      + - * / %   == < > != <= >=   && || !   ++ --   += -=

type-casting, especially using (int) and (double) for type-casting numbers
making random integers: (int)(m * Math.random() + n)
how the / and % operators work for integers; using n % d to test if d evenly divides n
using the + operator with a String

statements
control structures
the empty statement: ;
block statement: { }
while statement: while (*condition*) { *statements* }
for statement: for (*initialize* ; *condition* ; *update*) { *statements* }
simple if statement: if (*condition*) { *statements* }
if..else statement: if (*condition*) { *statements* } else { *statements* }
using if..else if... for multiway branching
using while(true) loops
using **break** in a loop
processing the characters in a String using a for loop
nested for loops
infinite loop
algorithm (the most basic object of study in computer science!)
pseudocode
bugs and debugging

graphics subroutines in JavaFX
pixels
the xy coordinate system that is used for drawing

important subroutines in a graphics context object g:
      g.setStroke(color), g.strokeRect(x,y,w,h), g.strokeOval(x,y,w,y), g.strokeLine(x1,y1,x2,y2)
      g.setFill(color), g.fillRect(x,y,w,h), g.fillOval(x,y,w,y)
      g.fillText(str,x,y)

basic color values: Color.RED, Color.GREEN, Color.BLUE, Color.WHITE, Color.BLACK, etc.

using drawing commands in control structures such as for loops