CSCI 136 Written Exam #0 **Name: _____**
Fundamentals of Computer Science II
Spring 2014

This exam consists of 5 problems on the following 8 pages.

You may use your single-sided handwritten 8 ½ x 11 note sheet during the exam. You may use a simple handheld calculator. No computers, mobile devices, cell phones, or other communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work**.

Problem	Points	Score
1	10	
2	10	
3	12	
4	12	
5	12	
Total	56	

1. Loops, input, output (10 points). Consider the following program:

```
public class Prob1
{
   public static void main(String [] args)
   {
      double m = Double.POSITIVE_INFINITY;
      int    b = 0;
      int    i = 0;

      for (String s : args)
      {
         double val = Double.parseDouble(s);
         if (val < m)
         {
            b = i;
            m = val;
         }
         i++;
      }
      System.out.printf("%d %s\n", b, args[b]);
   }
}
```

Below are five example executions of the program. Give the output produced by the program. If the given input would cause a runtime exception, write "error".

Command line	Output
% java Prob1 1.0 2.12 0.516	
% java Prob1	
% java Prob1 -1 -2 -0.5	
% java Prob1 the cat sat	
% java Prob1 3.14 3.14 42.0	

2. Multiple choice (10 points). Circle the ***one best*** answer:

a) You want to store a list of floating-point values. You want the list to automatically grow when necessary. Which of the following lines of code meets your requirements and also compiles?

```
 I.  ArrayList<double> data = new ArrayList<double>();
 II. ArrayList<Double> data = new ArrayList<Double>();
III. ArrayList<Double> data = new ArrayList<Double>;
 IV. Double [] data = new Double();
 V.  double data = {};
```

b) In object-oriented programming, which of the following best describes polymorphism:

I. Allows objects of different, but related types, to live in the same collection (e.g. in an array).
II. Allows objects to compare themselves against each other (e.g. to sort in ascending order).
III. Allows objects to reduce memory demands by providing an explicit destructor method.
IV. Allows objects to hide the internal details of a class's implementation from clients.
V. Allows objects to handle unexpected input (e.g. a non-numeric string when an integer is expected).

c) Which of the following is the best reason for designing a data type using Java generics:

I. You want to hide the internal details of the data type from clients.
II. You want to handle unexpected input.
III. You want the data type to store a collection of any reference data type a client might require.
IV. You want to be able to sort a list of instances of the data type.
V. You want to override an implemented method in a parent class.

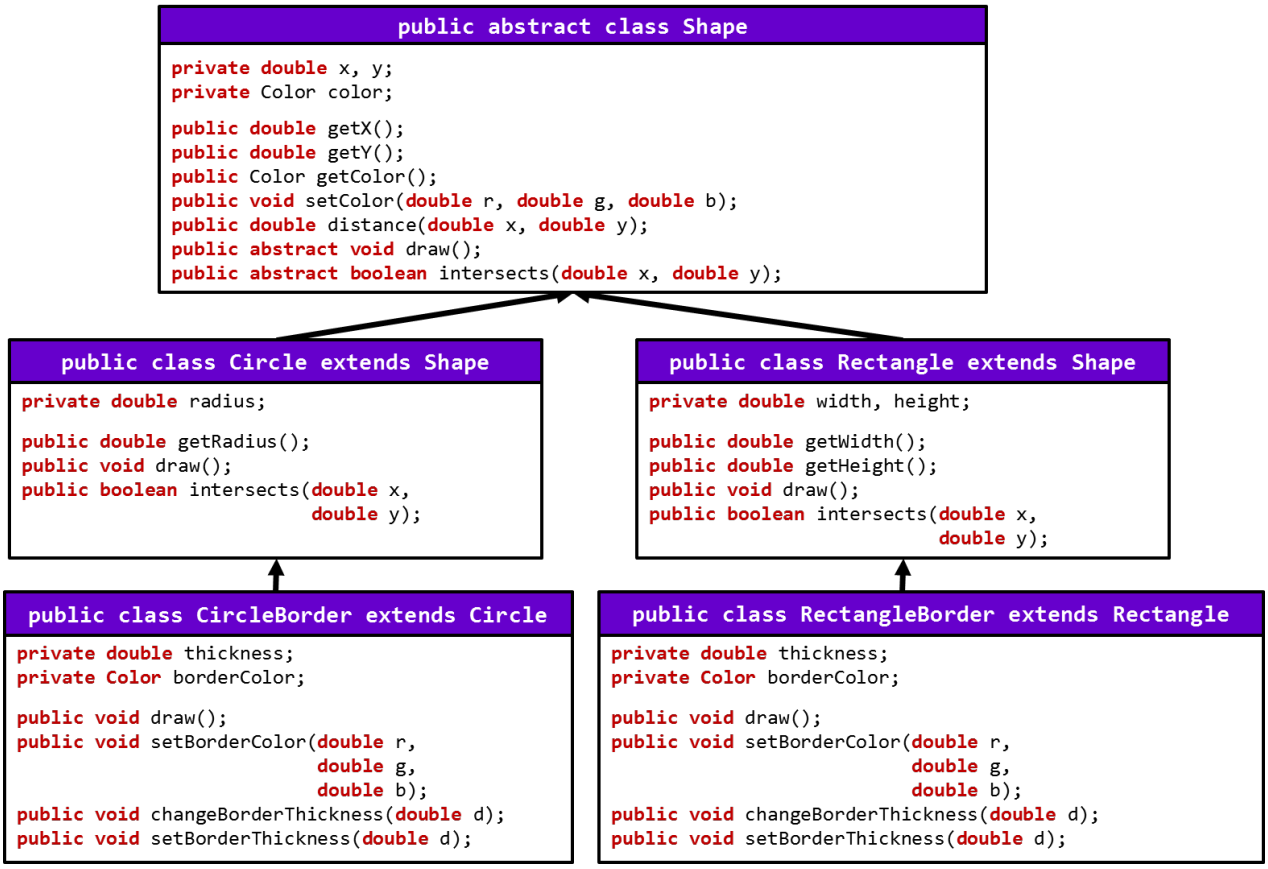
f) All the following are ***reliable*** conditions testing if the text contained in the `String` variable `password` is exactly equal to the text contained in the `String` variable `secret` ***EXCEPT***:

```
 I.   (password.equals(secret))
 II.  (secret.equals(password))
III.  (secret == password)
 IV.  (password.compareTo(secret) == 0)
 V.   (secret.compareTo(password) == (1/5))
```

g) You have a class called `Widget`. You have a variable storing a collection of `Widget` objects using a List abstract data type (ADT). You want to sort the variables using `Collections.sort()`. What must you add to the line "`public class Widget`" to help make this happen?

```
 I.   extends Comparable
 II.  extends CompareTo<Widget>
III.  implements Sortable
 IV.  implements Comparable<Widget>
 V.   implements compareTo(Widget other)
```

3. Objects and inheritance (12 points). Consider the following hierarchy of Java classes:



a) List the names of all ***primitive instance variables*** occurring in ***any*** class in the hierarchy.

b) List the names of all ***reference instance variables*** occurring in ***any*** class in the hierarchy.

c) List the names of all ***class or classes*** in the hierarchy that you could ***instantiate*** using the `new` operator.

d) List the names of all ***methods*** occurring in ***any*** class that ***override*** an implemented method in a parent.

e) Describe two advantages provided by an abstract class such as Shape:

Advantage 1:

Advantage 2:

4. Performance (12 points). You instrumented the following code fragment to measure how long it takes for different input sizes. The input is stored in the `double` array `data`. Throughout this problem, ***N refers to the number of elements in the array***.

```
Stats stats = new Stats();
result = result + mysteryCalc(data);
System.out.printf("Elapsed time = %.3f\n", stats.elapsedTime());
```

The following table gives the timing data for a series of experiments at different input sizes:

N	Elapsed time (seconds)
5	0.024
10	0.035
20	0.060
40	0.110
80	0.208
160	0.408
320	0.784
640	1.493

a) Which of the following best describes the order of growth of the running time of this program? ***Circle one*** of the following:

I. O(1), constant
II. O(N), linear
III. $O(N^2)$, quadratic
IV. $O(N^3)$, cubic
V. $O(N^4)$, quartic
VI. $O(2^N)$, exponential

b) Based on your answer in part a, give the equation showing the running time of the program in seconds as a function of the input size N. You should solve for the leading constant.

c) Based on your answer in part b, estimate the program's running time in seconds for an input size of N = 10,000.

d) You modify the code fragment to include an extra loop around the mystery calculation:

```
Stats stats = new Stats();
for (int i = 0; i < data.length / 2; i++)
{
   result = result + mysteryCalc(data);
}
System.out.printf("Elapsed time = %.3f\n",stats.elapsedTime());
```

Based on your answer in part a, which of the following would best describe the order of growth of the new code fragment? ***Circle one*** of the following:

I. O(1), constant
II. O(N), linear
III. $O(N^2)$, quadratic
IV. $O(N^3)$, cubic
V. $O(N^4)$, quartic
VI. $O(2^N)$, exponential

e) You further modify the code fragment to include a second extra inner loop:

```
Stats stats = new Stats();
for (int i = 0; i < data.length / 2; i++)
{
   for (int j = 0; j < 100; j++)
   {
      result = result + mysteryCalc(data);
   }
}
System.out.printf("Elapsed time = %.3f\n",stats.elapsedTime());
```

Based on your answer in part d, which of the following would best describe the order of growth of the new code fragment? ***Circle one*** of the following:

I. O(1), constant
II. O(N), linear
III. $O(N^2)$, quadratic
IV. $O(N^3)$, cubic
V. $O(N^4)$, quartic
VI. $O(2^N)$, exponential

5. Debugging, exceptions (12 points). You are working on a program that prints out the names of students who received an exam score between some minimum and maximum score (inclusive of the minimum and maximum). Your professor never awards fractional points, so all scores are integers. Here is an example data file, `scores.txt`:

```
Abe   87
Bob   53
Carol 92
Dave  78
Eve   43
```

Currently the program has a bug that causes all students' names to print regardless of their score. Here is an example run:

```
% java FindScores 80 100 scores.txt
Abe
Bob
Carol
Dave
Eve
DONE!
```

a) Find and correct the bug in the program. ***Circle the bug and provide the correct replacement code***.

```
    public class FindStudents
    {
       public static void main(String[] args)
       {
01       int minScore = Integer.parseInt(args[0]);
02       int maxScore = Integer.parseInt(args[1]);
         try
         {
03          Scanner scanner = new Scanner(new File(args[2]));
04          while (scanner.hasNext())
            {
05             String name  = scanner.next();
06             int    score = scanner.nextInt();

07             if ((score >= minScore) || (score <= maxScore))
08                System.out.println(name);
            }
09          scanner.close();
         }
         catch (FileNotFoundException e)
         {
10          System.out.println("File not found!");
         }
         catch (InputMismatchException e)
         {
11          System.out.println("Invalid file data!");
         }
12       System.out.println("DONE!");
       }
    }
```

5. Debugging, exceptions (continued)

Assume the previous bug is fixed. The main executable lines of code in the previous program are numbered. For each of the following situations, ***provide the sequence of line numbers*** that represents the flow of execution through the program. Additionally, circle whether the program crashes with a runtime exception or not (i.e. it would print a stack trace showing the location and type of exception and the final "DONE!" would NOT print).

b) The program is run with no command line arguments:

```
% java FindScores
```

Line sequence:

Crashes with exception? YES NO

c) The program is run with a filename that does not exist:

```
% java FindScores 50 100 bogusfilename.txt
```

Line sequence:

Crashes with exception? YES NO

d) The program is run with the following data file, `scores2.txt`:

```
Abe   87
Bob   53.5
Carol 92
Dave  78
Eve   43
```

```
% java FindScores 90 100 scores2.txt
```

Line sequence:

Crashes with exception? YES NO

e) Which of the following is a *checked exception* and thus its catch block is required in order for the program to compile? ***Circle one*** of the following:

I. `FileNotFoundException`
II. `InputMismatchException`
III. `Both`
IV. `Neither`