CSE 11 **START EARLY!**
Fall 2013
Program Assignment #8

Due: 7 Dec 2013, 11:00 AM (12 hour extension)
Covers Chapters: 12, 16, 17, 18, 19, 20

Total 100 points. Put your NAME:, LOGIN:, ID: In every java file you turn in.

Problem #1: **(10 points)**
Consider the following program, which performs integration of a particular function using standard Trapezoid integration. (You can download `Integrate.java` from the class website)

```
public class Integrate {
        public static double integrate(int  a, int b, int steps)
        {
                double sum=0;
                double delta = 1.0 * (b - a)/steps;
                double x = a;
                double f = 0.5*x*x + 3*x + 5;

                for (int i = 0; i< steps; i++)
                {
                        x = x + delta;
                        double fr = 0.5*x*x + 3*x + 5;
                        double area = f * delta + 0.5*(fr - f)*delta;
                        sum += area;
                        f = fr;
                }
                return sum;
        }
        public static void main(String [] args)
        {
                int a, b, step;
                a = Integer.parseInt(args[0]);
                b = Integer.parseInt(args[1]);
                step = Integer.parseInt(args[2]);
                System.out.format("Integral is %f\n", integrate(a,b,step));
        }
}
```

Modify the program in the following way:

- Redefine `Integrate` to use a recursive function (method) called `rintegrate` in place of the iterative for-loop provided here. you will need to determine the correct base case and the arguments for your `rintegrate` method. Do NOT change the format of the output
-

Turn in: Integrate.java

Problem #2 (15 points)
Download Towers.java from the class website and modify so that it prints out the depth of the call stack for each call and move. Notice that each level is 4 dots and that each “Move” is

indented to indicate which recursion level was used to give the particular move instruction. The following is example output for the case of 3 moves.

```
$ java Towers 3
Starting Tower of Hanoi
     Disks: 3
Move from Tower A to Tower C

   A     B     C
   |     |     |
   3     |     |
   |     |     |
====================
disk: 3, src  A, aux  B, dst  C
....disk: 2, src  A, aux  C, dst  B
........disk: 1, src  A, aux  B, dst  C
........Move disk  1 from  A to  C on
....Move disk  2 from  A to  B B
........disk: 1, src  C, aux  A, dst  B
........Move disk  1 from  C to  B
Move disk  3 from  A to  C
....disk: 2, src  B, aux  A, dst  C
........disk: 1, src  B, aux  C, dst  A
........Move disk  1 from  B to  A
....Move disk  2 from  B to  C
........disk: 1, src  A, aux  B, dst  C
........Move disk  1 from  A to  C
Moves: 7
```

Turn in – your modified Towers.java

Problem #3 (25 points)

JAEA 17.9.1.

Download ButtonBallController.java and FallingBall.java from the class website to save you time.

please note: The text says "The SimpleButton class should be defined as an extension of JButton. Its constructor should expect a BallAndWindowController as a parameter.". It should say "expect a ButtonAndWindowController as a parameter"

Turn in - SimpleButton.java, ButtonAndWindowController.java

Problem #4 (20 points)

Write a program called CapCase.java that prompts the user for a line of text (a String). CapCase should then output the string with the first letter of every word capitalized. The words should be separated by a space an example input and output is.

```
$ java CapCase
Type in string: this is  a way to type with two   or     more spaces
This Is A Way To Type With Two Or More Spaces
```

Run your code with the different inputs to test

1. onlyone
2. ALL CAPS
3. a pretty long Sentence with some Words already Capitalized.
4. It's not everyday you meet a Wookie!
5. tHIS iS   sPORTSCENTER. go tritons!
6. This was the 10 last weeks.

Hint: use Scanner with System.in. Look at some already-defined String functions.
Turn in – CapCase.java

Problem #5 (30 points)

Define a program called Sorter.java. Sorter should

1. Read lines from the standard input (System.in).
2. Store each line in an array of Strings.
3. Implement a bubble sort to sort the Strings in **decreasing** order
4. Print out the Strings in sorted order

To reverse sort the lines of any text file, you should be able to run your program as

```
$ cat file | java Sorter
```

Do NOT assume a maximum number of lines for your input (you can use the same technique as ReverseBuf.java of a previous assignment.)

Turn in – Sorter.java

Ungraded Exercises
JAEA 18.6.1
JAEA 18.5.1
JAEA 18.4.1

Turning in your Program

YOU MUST BE ON THE LAB MACHINES FOR THIS TO WORK. PLEASE VERIFY WELL BEFORE THE DEADLINE THAT YOU CAN TURNIN FILES

You will be using the "bundleP8" program that will turn in the following files

```
Integrate.java
Towers.java
SimpleButton.java
ButtonAndWindowController.java
CapCase.java
Sorter.java
```