# IS 380: Object-Oriented Programming
# Spring 2017

## Extra Credit Assignment

Due Date: **3:59 PM, March 29, 2017** (Submit via WebCampus).
Weights: 10 points each. Up to 60% of the points you missed in the midterm exam

For example, if you got 73 in the midterm, you can choose to do any 2 of the following questions.
However the maximum extra points you can earn is (100-73)*60% = 16.2 point.

1. **[File Processing and Repetition Structure]**
   Write code that does the following
   (1) Ask the user to enter a file name.
   (2) Ask the user for two integers as the lower bound and the upper bound.
   (3) Use a loop to write all the odd numbers between the lower bound and the upper bound (inclusive) to the file, and then close the file.

```
Please enter the file name: D:\temp\oddNumbers.txt
Please enter an integer as the lower bound: 2
Please enter an integer as the upper bound
(NOTE: the upper bound should be greater than the lower bound.): 55
The numbers were written to the file D:\temp\oddNumbers.txt
```

***Grading criteria:***

(1) Correctness
   (a) The code can be compiled without any syntax error.
   (b) The code can generate the requested results.
   (c) The program is documented using comments.
(2) Technique used
   (a) Either **`while`** loop, **`do while`** loop, or **`for`** loop should be used to complete the program.
   (b) **`PrintWriter`** class is used.
   (c) The file is closed before the program terminates.

2. **[Decision structure and Method]**
Write a program that has variables to hold three test score. The program should ask the user to enter three test scores and then assign the value entered to the variables. The program should display the average of the test score and the letter grade that is assigned for the test score average.

The program should use a method (other than the main method) to determine the letter grade. The method should accept a test score as the parameter, and return a letter grade (a string). The letter grade is determined using the grading scheme in the following table:

| | |
|---|---|
| 90-100 | A |
| 80-89 | B |
| 70-79 | C |
| 60-69 | D |
| Below 60 | F |

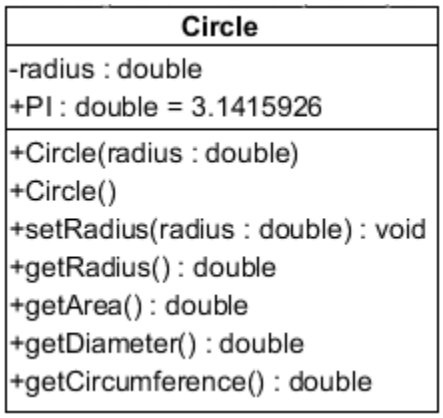
```
Please enter the first score: 80
Please enter the second score: 90
Please enter the third score: 100
The average score is: 90.0, which is A
```

***Grading criteria:***

(1) Correctness
   (a) The code can be compiled without any syntax error.
   (b) The code can generate the requested results.
   (c) The program is documented using comments.

(2) Technique used
   (a) Variable and/or constant declaration and initialization is done following the naming convention of Java programs.
   (b) Getting user input is done.
   (c) Arithmetic operators are used.
   (d) Showing output is done.
   (e) **`if-else`** or **`switch`** statements should be used to complete the program.
   (f) A method is implemented as specified in the question.

3. **[Class and Object]**
Write a **`Circle`** class.




(1) The class has the following fields: radius: a double. PI: a final double initialized with the value 3.1415926.
(2) The class should have the following methods:
   (a) Constructor: Accepts the radius of the circle as an argument.
   (b) No-arg constructor.
   (c) **`setRadius`**: A set method for the radius field.
   (d) **`getRadius`**: A get method for the radius field.
   (e) **`getArea`**: Returns the area of the circle, which is calculated as
   area = PI * radius * radius
   (f) **`getDiameter`**: Returns thediameter of the circle, which is calculated as
   diameter = radius *2
   (g) **`getCircumference`**: Returns the circumference of the circle, which is calculated as
   circumference = 2 * PI * radius.

Then write a program to test the **`Circle`** class: It should ask the user for the circle's radius. It then creates a Circle object named **`clock`**. It prints out the area, diameter and circumference of the clock.

***Grading criteria:***

(1) Correctness
   (a) The code can be compiled without any syntax error.
   (b) The code can generate the requested results.
   (c) The program is documented using comments.
(2) Technique used
   (a) **Circle** class is implemented as specified.
   (b) Fields and Methods are named properly, following the naming convention discussed in class.
   (c) Arithmetic operators are used.

4. **[Repetition Structure and Input Validation (Decision Structure)]**
Write a program that collect data and calculate the average rainfall over a period of years.
The program would first ask the user for the number of year. The program then asks the user for the inches of rainfall for each month. After all iterations, the program should display the number of months, the total inches of rainfall, and the average rainfall per month for the entire period.

The program should also validate user's input: it does not accept a number less than 1 for the number of years; it does not accept negative numbers for the monthly rain fall.

```
<terminated> AverageRainfall [Java Application] C:\Program Files\Ja
Enter the number of years: 0
Invalid. Enter 1 or greater: 2
Enter the rainfall, in inches, for each month.
Year 1 month 1: -1
Invalid. Enter 0 or greater: 0
Year 1 month 2: 0
Year 1 month 3: 3
Year 1 month 4: 2
Year 1 month 5: 5
Year 1 month 6: 0
Year 1 month 7: 0
Year 1 month 8: 0
Year 1 month 9: 0
Year 1 month 10: 12
Year 1 month 11: 0
Year 1 month 12: 0
Year 2 month 1: 3
Year 2 month 2: 5
Year 2 month 3: 13
Year 2 month 4: 3
Year 2 month 5: 2
Year 2 month 6: 1
Year 2 month 7: 0
Year 2 month 8: 0
Year 2 month 9: 3
Year 2 month 10: 2
Year 2 month 11: 1
Year 2 month 12: 0

Number of months: 24
Total rainfall: 55.0 inches
Average monthly rainfall: 2.29 inches
```

***Grading criteria:***

(1) Correctness
   (a) The code can be compiled without any syntax error.
   (b) The code can generate the requested results.
   (c) The program is documented using comments.

(2) Technique used
   (a) Either **`while`** loop, **`do while`** loop, or **`for`** loop should be used to complete the program.
   (b) Arithmetic operators are used.
   (c) User input validation is properly implemented.