COMP2026 Problem Solving Using Object Oriented Programming

Laboratory 3

Part A Discovery Exercises

Type your answers in XXXXXXXX_lab03.docx

Task 1: File Input

a) Create a project called **Lab03** in IntelliJ. Put the given **`FileInputExample.java`** program in to the **src** folder. Run the program and paste the output below.

b) The given **`FileInputExample.java`** program reads input from a file called input.txt. Put the given **input.txt** file into the project file **Lab03**. Run the program again and paste the output below.

Task 2: File Output

a) Put the given **`FileOutputExample.java`** program in to the **src** folder of the **Lab03** project. This program writes output into a file called **output.txt**.

i. Run the program. Where is the output.txt file?

____

____

ii. Paste the content of output.txt below.

iii. Run the program again. Paste the content of output.txt below.

Does the content of output.txt change?

Task 3: File Append

a) Put the given `FileAppendExample.java` program in to the **src** folder of the **Lab03** project. This program writes output into a file called **appendOutput.txt**.

i. Run the program once. Paste the content of appendOutput.txt below.

ii. Run the program again. Paste the content of appendOutput.txt below.

Does the content of appendOutput.txt change? Why?

Task 4: Loop Conversion

Rewrite the given loop into other types of loop. In some parts, you may need to add more statements rather than just writing a loop.

a) Given:

```
for (int i = 0; i < n; i++){  //n is an integer
    System.out.println(i);
}
```

i. Rewrite the above using a **while** loop.

ii. Rewrite the above using a **do...while** loop.

iii. Suppose the following print statement is added **after** the above three loops to print the value of **`i`**, what will be printed? Is there any side-effect when converting a for loop to another type?

```
System.out.println(i);
```

b) Given:

```
    while(in.hasNextLine()){         //in is a scanner object
        System.out.println(in.nextLine());
    }
```

i. Rewrite the above using a **for** loop.

ii. Rewrite the above using a **do..while** loop.

c) Given:

```
do{
    System.out.print("Enter an integer: ");
    n = in.nextInt();     //in is a scanner object
}while(n < 1 || n > 10);  //n is an integer
```

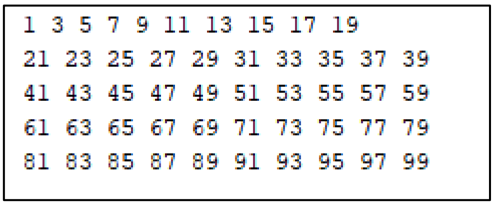
i. Rewrite the above using a **for** loop.

ii. Rewrite the above using a **while** loop.

Part B Programming Exercises

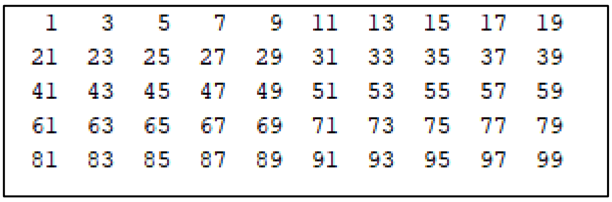
Task 1: Printing Odd Numbers

a) Write a program called **`OddNumbers`** that prints the first 50 odd positive integers with 10 numbers in a row.
Sample output:



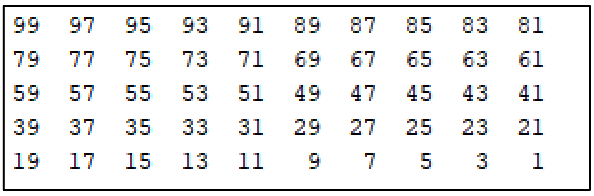
```
1 3 5 7 9 11 13 15 17 19
21 23 25 27 29 31 33 35 37 39
41 43 45 47 49 51 53 55 57 59
61 63 65 67 69 71 73 75 77 79
81 83 85 87 89 91 93 95 97 99
```

b) Modify part (a) to write a program called **`FormattedOddNumbers`** that prints the first 50 odd positive integers as follows. Hint: use printf().



```
   1   3   5   7   9  11  13  15  17  19
  21  23  25  27  29  31  33  35  37  39
  41  43  45  47  49  51  53  55  57  59
  61  63  65  67  69  71  73  75  77  79
  81  83  85  87  89  91  93  95  97  99
```

c) Write a program called **`ReverseOddNumbers`** that prints the first 50 odd positive integers with 10 numbers in a row, but in reverse order.
Sample output:

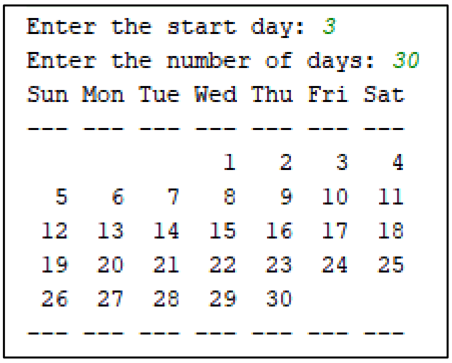


```
99  97  95  93  91  89  87  85  83  81
79  77  75  73  71  69  67  65  63  61
59  57  55  53  51  49  47  45  43  41
39  37  35  33  31  29  27  25  23  21
19  17  15  13  11   9   7   5   3   1
```

Task 2: Calendar Month

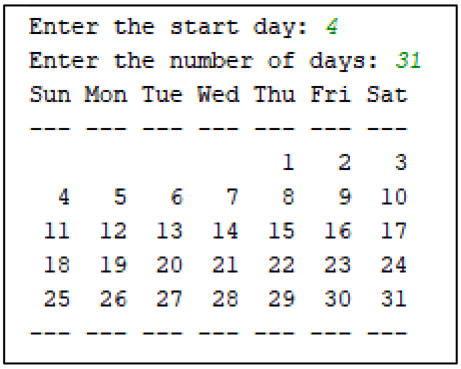
Write a program called **CalendarMonth** that prints a calendar month. The program should prompt user to input the start day of the month (Sunday is 0, Monday is 1, …) and the number of days in the month.

Sample output 1:



```
Enter the start day: 3
Enter the number of days: 30
Sun Mon Tue Wed Thu Fri Sat
--- --- --- --- --- --- ---
              1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30
--- --- --- --- --- --- ---
```

Sample output 2:

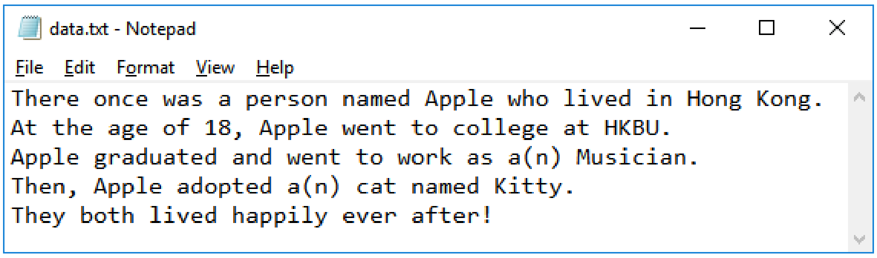


```
Enter the start day: 4
Enter the number of days: 31
Sun Mon Tue Wed Thu Fri Sat
--- --- --- --- --- --- ---
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15  16  17
 18  19  20  21  22  23  24
 25  26  27  28  29  30  31
--- --- --- --- --- --- ---
```

Task 3: Line Number

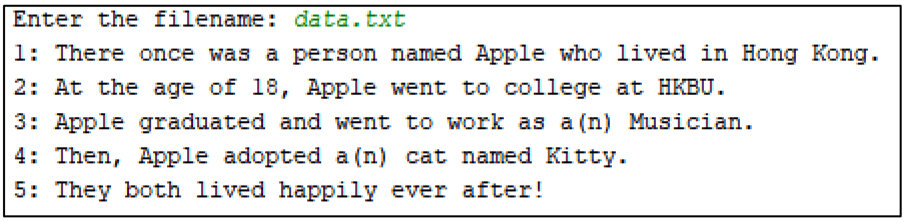
Write a program called **LineNumber** that asks the user for the name of a text file and reads strings from the text file, one string per line. Then the program displays the contents of the file with each line preceded with a line number followed by a colon. The line numbering should start at 1.
*Note: Put the input file, e.g. data.txt, inside the **Project folder** of IntelliJ IDEA.*

Content of the input file:



```
There once was a person named Apple who lived in Hong Kong.
At the age of 18, Apple went to college at HKBU.
Apple graduated and went to work as a(n) Musician.
Then, Apple adopted a(n) cat named Kitty.
They both lived happily ever after!
```

Sample output:

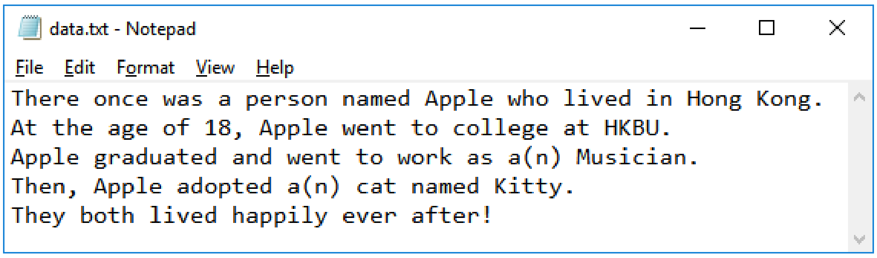


```
Enter the filename: data.txt
1: There once was a person named Apple who lived in Hong Kong.
2: At the age of 18, Apple went to college at HKBU.
3: Apple graduated and went to work as a(n) Musician.
4: Then, Apple adopted a(n) cat named Kitty.
5: They both lived happily ever after!
```

Task 4: Line Number to File

Modify the program you have done in Task 3 that asks the user for both input filename and output filename. Then the program writes the contents of the file with each line preceded with a line number followed by a colon to the output file. Name your program **`LineNumberToFile`**.

Content of the input file:

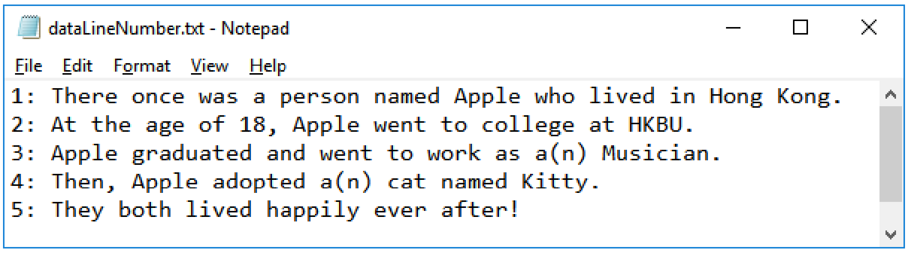


```
There once was a person named Apple who lived in Hong Kong.
At the age of 18, Apple went to college at HKBU.
Apple graduated and went to work as a(n) Musician.
Then, Apple adopted a(n) cat named Kitty.
They both lived happily ever after!
```

Sample output:

```
Enter the input filename: data.txt
Enter the output filename: dataLineNumber.txt
```

Content of the output file:

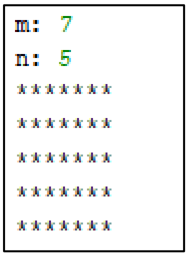


```
1: There once was a person named Apple who lived in Hong Kong.
2: At the age of 18, Apple went to college at HKBU.
3: Apple graduated and went to work as a(n) Musician.
4: Then, Apple adopted a(n) cat named Kitty.
5: They both lived happily ever after!
```

Task 5: Printing Patterns

a) Write a program called **`Rect`** that reads in two numbers *m* and *n* and then prints *n* rows of *m* asterisks.

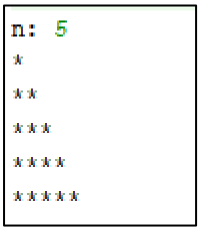
Sample output:



```
m: 7
n: 5
*******
*******
*******
*******
*******
```

b) Modify the program in part (a) that reads in a number *n* and prints a triangle with *n* rows of asterisks. Name the program **`RightTriangle`**.

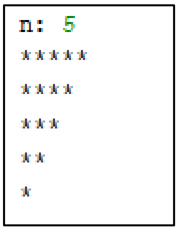
Sample output:



```
n: 5
*
**
***
****
*****
```

c) Modify the program in part (b) to prints a triangle with *n* rows of asterisks as follow. Name the program **`InvertedTriangle`**.

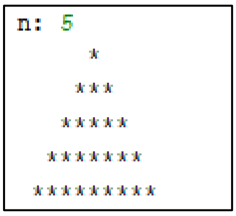
Sample output:



```
n: 5
*****
****
***
**
*
```

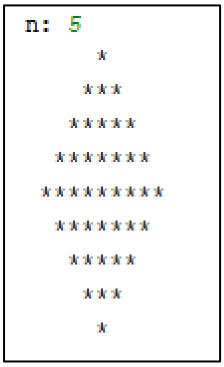
d) Write a program called **`IsoscelesTriangle`** that reads in a number *n* and prints a triangle with *n* rows of asterisks as follow.

Sample output:



```
n: 5
    *
   ***
  *****
 *******
*********
```

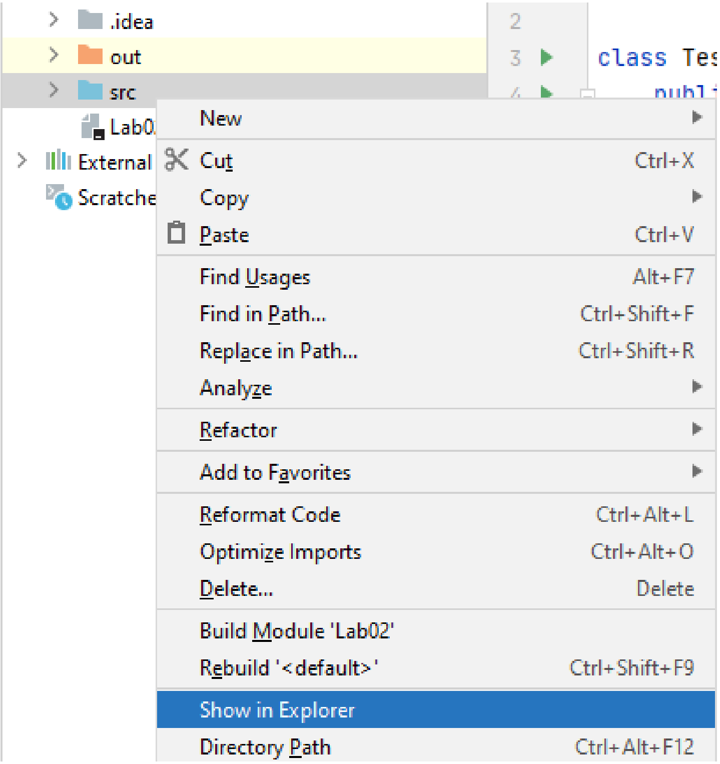
e) Write a program that reads in a number *n* and prints a diamond shape with *2n – 1* rows of asterisks as follow. Name the program **`Diamond.`**
Sample output:



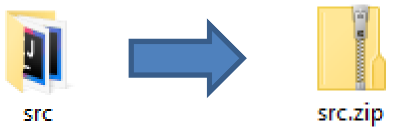
```
n: 5
     *
    ***
   *****
  *******
 *********
  *******
   *****
    ***
     *
```

Part C Submitting Exercises

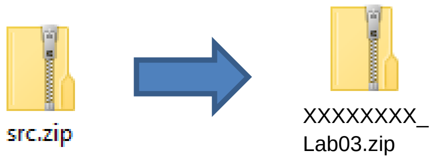
Step 1: Right-click the **src** folder and select **Show in Explorer**



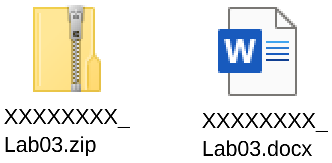
Step 2: Zip the **src** folder into **src.zip**



Step 3: Rename the src.zip file to ***XXXXXXXX_lab03.zip*** where ***XXXXXXXX*** is your **student id**



Step 4: Submit ***XXXXXXXX_lab03*.zip** and **XXXXXXXX_lab03.docx** to Moodle.



References

[1] Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. Dubuque, IA: McGraw-Hill.

[2] Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.

[3] Farrell, J. (2012). *Java programming. Boston, MA: Course Technology Cengage Learning*

[4] Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.

[5] Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.

[6] Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.

[7] Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.

[8] Schildt, H., & Skrien, D. J. (2013). Java programming: A comprehensive introduction. New York: McGraw-Hill.

[9] Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education

[10] Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.

[11] yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming

[12] Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.