COSC 237 – ASSIGNMENT 02

Submission Instruction

Failure to follow the below instructions will result in a 20% penalty.

1. Do each programming problem in a single file with the name format: ***"<your_last_name>_HW_XX_YY.java"*** where ***XX*** is the assignment number and ***YY*** is the problem number. For instance, for assignment 01 and student name "David Smith", the filenames would be "Smith_HW_01_01.java", "Smith_HW_01_02.java", etc for questions #1, #2, and so on.
2. When you are done, create a folder named ***"<first_name>_<last_name>_HW_XX"*** and put all your ***.java*** and input files or test files (NO ***.o*** or ***.class*** files) in this folder. For example: folder name *"David_Smith_HW_01"* for student name "David Smith". No *.class* files or test files should be included.
3. Finally, create a ZIP file of this folder with the same name (i.e., "David_Smith_HW_01.zip") and submit it over Blackboard. If you have an issue, change the file extension to ***.txt*** (i.e., change "David_Smith_HW_01.zip" to "David_Smith_HW_01.txt") and resubmit.

For this assignment, you will work with a partner and both of you should contribute significantly to the solution for every question. I suggest that both of you do the programs alone, pass your version to your partner, criticize the version written by your partner, and present a solution that has the best features. In principle, you should not discuss the homework with anyone, except your homework partner and your instructor. If you do discuss it with someone else, you should say this in the preamble of the homework. You should only submit work that is completely your own and you should be able to explain all your homework to me. Make sure you handle **invalid input, including type mismatch** - input validation is a requirement! Focus on good design, good style, and, most importantly, reuse of code. Start early and remember the rules about late assignments/syllabus → **no late assignments!**

Question 1 [40pts]. A complex ("imaginary") number has the form a + bi, where a is called the real part, b is called the imaginary part, and i = sqrt(-1). A complex number a + bi can be expressed as the ordered pair of real numbers (a, b).

Arithmetic operations on two complex numbers (a, b) and (c, d) are as follows:

Addition: (a, b) + (c, d) = (a + c, b + d)

Subtraction: (a, b) - (c, d) = (a - c, b - d)

Multiplication: (a, b) * (c, d) = (a * c - b * d, a * d + b * c)

Division: (a, b) / (c, d) = ((a * c + b * d)/(c^2 + d^2), (b * c - a * d)/(c^2 + d^2))

Absolute value: |(a, b)| = sqrt(a^2 + b^2)

Design and implement a **ComplexNumber** class that represents the real and imaginary parts as double values and provides at least the following methods:

- **Constructors** for default and explicit initialization.
- A method to **read** a complex number. Look at the sample output screen for the design required.
- A method to **print** a complex number as (a, b). Have 2 decimals for both.
- A method called **getReal** that returns the real part of a complex number.
- A method called **getImaginary** that returns the imaginary part of a complex number.
- Methods **equal**, **copy**, **getCopy**, **toString**.
- Arithmetic methods to **add**, **subtract**, **multiply**, and **divide** two complex numbers.
- A method called **cAbs** to implement the absolute value of a complex number.

  To test your class write a client that has at least a function **menu()** with options for the methods implemented and an option to exit. Your program should loop until the user chooses to exit. In this loop you are required to use a **switch** statement for all possible cases (similar design as the one used for Problem#1 in Assignment#1).

SAMPLE OUTPUT:

```
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 2
Enter complex number (real imaginary): 3.4 5.6
Enter complex number (real imaginary): 1.23 2.56
First complex number is: (3.40, 5.60)
Second complex number is: (1.23, 2.56)
Result: (3.40, 5.60) - (1.23, 2.56) = (2.17, 3.04)
   Command number 1 completed.
```

```
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 4
Enter complex number (real imaginary): 11.2 22.1
Enter complex number (real imaginary): 1.45 3.56
First complex number is: (11.20, 22.10)
Second complex number is: (1.45, 3.56)
Result: (11.20, 22.10) / (1.45, 3.56) = (6.42, -0.53)
   Command number 2 completed.
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 2
Enter complex number (real imaginary): 1.78 4.5
Enter complex number (real imaginary): 3.56 8.9
First complex number is: (1.78, 4.50)
Second complex number is: (3.56, 8.90)
Result: (1.78, 4.50) - (3.56, 8.90) = (-1.78, -4.40)
   Command number 3 completed.
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 3
Enter complex number (real imaginary): 2.22 3.33
Enter complex number (real imaginary): 1.24 2.45
First complex number is: (2.22, 3.33)
Second complex number is: (1.24, 2.45)
Result: (2.22, 3.33) * (1.24, 2.45) = (-5.41, 9.57)
   Command number 4 completed.
Your options for Complex arithmetic are:
----------------------------------------
```

```
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 6
Enter complex number (real imaginary): 1.11 2.22
Enter complex number (real imaginary): 1.11 2.22
First complex number is: (1.11, 2.22)
Second complex number is: (1.11, 2.22)
The complex numbers are equal.
   Command number 5 completed.
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 6
Enter complex number (real imaginary): 1.2 2.3
Enter complex number (real imaginary): 11.2 2.3
First complex number is: (1.20, 2.30)
Second complex number is: (11.20, 2.30)
The complex numbers are NOT equal.
   Command number 6 completed.
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 5
Enter complex number (real imaginary): 11.1 22.2
The complex number is: (11.10, 22.20)
Result: |(11.1, 22.2)| = 24.82
   Command number 7 completed.
Your options for Complex arithmetic are:
----------------------------------------
 1) Add 2 complex numbers
 2) Subtract 2 complex numbers
 3) Multiply 2 complex numbers
 4) Divide 2 complex numbers
 5) Absolute value of a complex number
 6) Compare 2 complex numbers
 0) EXIT
Please enter your option: 0
Testing completed.
```

__**Question 2 [30pts].**__ **(Unsorted list: array implementation)** The unsorted list ADT discussed in class (file "Lecture_Array_Based_Lists.pdf") should be extended by the addition of two new methods:

1. A method named **merge** that concatenates 2 unordered lists into a third. Assume that list_1 and list_2 don't have any keys in common. The resulting list should be an unsorted list that contains all of the items from list_1 and list_2 (preserve the order).
2. A method named **split** that divides a list into 2 lists according to a key. If list_1 and list_2 are the resulting lists, list_1 should contain all the items of the original list whose keys are less than or equal to the key passed and list_2 should contain all the items of the original list whose keys are larger than the key passed.

Next, create a client to test your program. The client should work with 3 sorted lists named list_1, list_2 and result. Read the data for list_1 and list_2 from the files list1.txt and list2.txt (take input from user the names of the input files, handle the FileNotFoundException exception (try/catch) and consume unwanted input.) Merge list_1 and list_2 into result and split result according to a key (input from the user). Make sure you handle __**all**__ possible errors.

SAMPLE OUTPUT:

```
Please input the name of the file to be opened for first list: list1.txt
Please input the name of the file to be opened for second list: list2.txt
The first list is:
13  25  34  67  56  10  20  27  2  5  1  45  59
The second list is:
73  29  14  87  72  100  200  127  22  15  19  145  159  78
The merged list is:
13  25  34  67  56  10  20  27  2  5  1  45  59  73  29  14  87  72  100  200  127  22  15
  19  145  159  78
Enter key for split: 49
The first list after split is:
13  25  34  10  20  27  2  5  1  45  29  14  22  15  19
The second list after split is:
67  56  59  73  87  72  100  200  127  145  159  78
```

FOR input files:

```
list1.txt: 13 c v b 25 34 x x 67 56 10 a a 20 27 2 a s 5 1 45 59
list2.txt: 73 29 c c c 14 87 72 100 200 c c c 127 22 15 19 c v v v 145 159 78
```

Question 3 [30pts]. **(Sorted list: array implementation)** Same problem for **this** sorted list ADT discussed in class.

NOTE: The merge method for the sorted list shouldn't look like the same method in the unsorted list. You should merge the two lists in **one traversal** (do not make repeated calls to insert – very inefficient!). You will not get credit for the same method in both classes.

SAMPLE OUTPUT:

```
Please input the name of the file to be opened for first list: list1.txt
Please input the name of the file to be opened for second list: list2.txt
The first list is:
2   5   8   11   14   29   33   43   45   51   55   65   70   75
The second list is:
1   4   7   9   10   15   35   49   57   59   67
The merged list is:
1   2   4   5   7   8   9   10   11   14   15   29   33   35   43   45   49   51   55   57   59   65   67   70   7
5
Enter key for split: 13
The first list after split is:
1   2   4   5   7   8   9   10   11
The second list after split is:
14 15   29   33   35   43   45   49   51   55   57   59   65   67   70   75
```

FOR input files:

```
list1.txt: 2 5 8 m m b 11 14 29 x 33 43 45 51 z z 55 65 70 b 75
list2.txt: 1 ccc 4 bb 7 mm 9 10 15 35 x x x 49 57 59 67
```