Seneca College **Jan 24, 2019**

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

**JAC444** **Demo Due dates : Jan 30 & Feb 06, 2019**

**Final Code Submission Date: Feb 08, 2019**

# Workshop 3

**Notes:**

- **i.** Each task should be presented during the lab, demo worth 70% of the workshop marks and code uploading worth the other 30%.
- **ii.** At least one of the tasks should be demoed in Jan 30th lab. The rest should be demoed in Feb 06th lab.
- **iii.** Make sure you have all security and check measures in place, like wrong data types etc., implement the proper Exception Handling in your tasks.
- **iv.** Make your project in proper hierarchy; introduce proper class coherence in your project. Proper packages and your project should be handled by only one main method which should be in a TesterClass.
- **v.** Given output structure is just for student to have a glimpse what the output can look, students are free to make the output better in any way.

Other inputs can be given during demo, so make sure you test your program properly.

**Task 1:** A complex number is a number in the form *a* + *bi*, where *a* and *b* are real numbers and *i* is 2-1. The numbers a and b are known as the real part and imaginary part of the complex number, respectively. You can perform addition, subtraction, multiplication, and division for complex numbers using the following formulas:

$$\boldsymbol{a + bi + c + di = (a + c) + \ (b + d)i}$$

$$\boldsymbol{a + bi - (c + di) = (a - c) + \ (b - d)i}$$

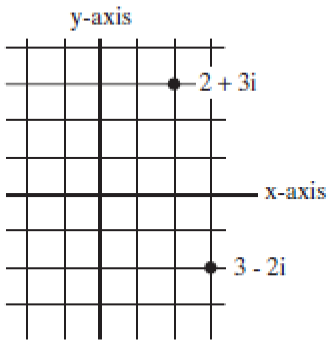
$$\boldsymbol{(a + bi) * (c + di) = (ac - bd) + \ (bc + ad)i}$$

$$\boldsymbol{(a + bi)/(c + di) = (ac + bd)/(c^2 + d^2) + \ (bc - ad)i/(c^2 + d^2)}$$

You can also obtain the absolute value for a complex number using the following formula:

$$\boldsymbol{|a + bi| = \ \sqrt{a^2 + b^2}}$$

(A complex number can be interpreted as a point on a plane by identifying the (*a,b*) values as the coordinates of the point. The absolute value of the complex number corresponds to the distance of the point to the origin, as shown in Figure)



- Design a class named **Complex** for representing complex numbers.
- The methods
  - **add**,
  - **subtract**,
  - **multiply**,
  - **divide**, and
  - **abs**

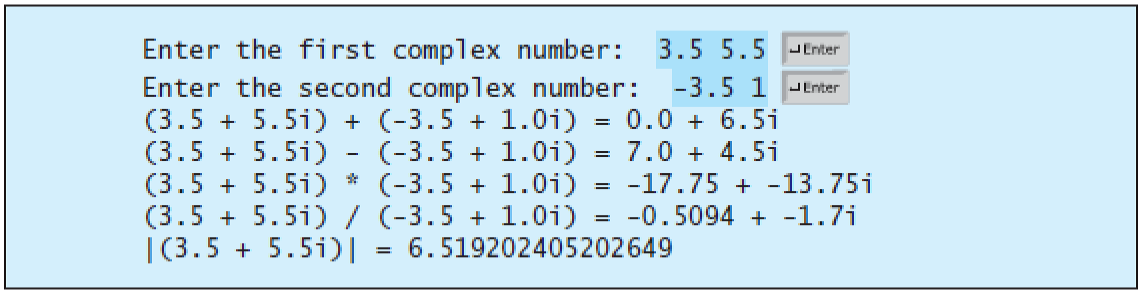
  for performing complexnumber operations.
- Override **toString** method for returning a string representation for a complex number. (The **toString** method returns **(a + bi)** as a string. If **b** is **0**, it simply returns **a**. )
- Your **Complex** class should also implement the **Cloneable** interface.

Provide three constructors

- **Complex(a, b)**,
- **Complex(a)**, and
- **Complex()**.

**Complex()** creates a **Complex** object for number **0** and **Complex(a)** creates a **Complex** object with **0** for **b**. Also provide the **getRealPart()** and **getImaginaryPart()** methods for returning the real and imaginary part of the complex number, respectively.

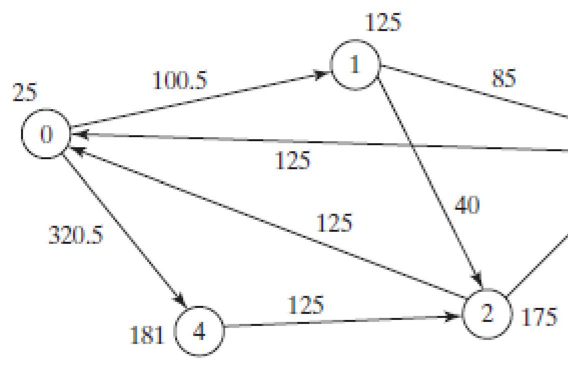
Write a test program that prompts the user to enter two complex numbers and displays the result of their addition, subtraction, multiplication, division, and absolute value. Here is a sample run:



```
Enter the first complex number:  3.5 5.5
Enter the second complex number:  -3.5 1
(3.5 + 5.5i) + (-3.5 + 1.0i) = 0.0 + 6.5i
(3.5 + 5.5i) - (-3.5 + 1.0i) = 7.0 + 4.5i
(3.5 + 5.5i) * (-3.5 + 1.0i) = -17.75 + -13.75i
(3.5 + 5.5i) / (-3.5 + 1.0i) = -0.5094 + -1.7i
|(3.5 + 5.5i)| = 6.519202405202649
```

**Task 2:**

Banks lend money to each other. In tough economic times, if a bank goes bankrupt, it may not be able to pay back the loan. A bank's total assets are its current balance plus its loans to other banks. The diagram below shows five banks. The banks' current balances are 25, 125, 175, 75, and 181 million dollars, respectively. The directed edge from node 1 to node 2 indicates that bank 1 lends 40 million dollars to bank 2.



If a bank's total assets are under a certain limit, the bank is unsafe. The money it borrowed cannot be returned to the lender, and the lender cannot count the loan in its total assets. Consequently, the lender may also be unsafe, if its total assets are under the limit.

Write a program to find all the unsafe banks. Your program reads the input as follows.

1. It first reads two integers **n** and **limit**, where **n** indicates the number of banks and **limit** is the minimum total assets for keeping a bank safe.
2. It then reads **n** lines that describe the information for **n** banks with IDs from **0** to **n-1**.

The first number in the line is the bank's balance, the second number indicates the number of banks that borrowed money from the bank, and the rest are pairs of two numbers. Each pair describes a borrower. The first number in the pair is the borrower's ID and the second is the amount borrowed. For example, the input for the five banks in above picture is as follows (**note that the limit is 201**):

Number of banks: 5
Minimum asset limit: 201
Bank # 0 → Balance: 25 → Number of banks Loaned: 2 → Bank ID: 1 → Amount: 100.5 → Bank ID: 4 → Amount: 320.5
Bank # 1 → Balance: 125 → Number of banks Loaned: 2 → Bank ID: 2 → Amount: 40 → Bank ID: 3 → Amount: 85
Bank # 2 → Balance: 175 → Number of banks Loaned: 2 → Bank ID: 0 → Amount: 125 → Bank ID: 3 → Amount: 75
Bank # 3 → Balance: 75 → Number of banks Loaned: 1 → Bank ID: 0 → Amount: 125
Bank # 4 → Balance: 181 → Number of banks Loaned: 1 → Bank ID: 2 → Amount: 125

The total assets of bank 3 are (75 + 125), which is under 201, so bank 3 is unsafe. After bank 3 becomes unsafe, the total assets of bank 1 fall below (125 + 40). Thus, bank 1 is also unsafe.
**Note: Program should take inputs from the user like Number of banks, Minimum asset limit and then all other inputs**

**The output of the program should be**
Unsafe banks are 3 and Bank 1

**Task 3.**

- Design an interface named **Colorable** with a **void** method named **howToColor()**. Every class of a colorable object must implement the **Colorable** interface.
- Design a class named **Square** that extends **GeometricObject** and implements **Colorable**.
- Implement **howToColor** to display the message **Color all four sides**.
- Write a test program that creates an array of five **GeometricObject**s. For each object in the array, display its area and invoke its **howToColor** method if it is colorable.