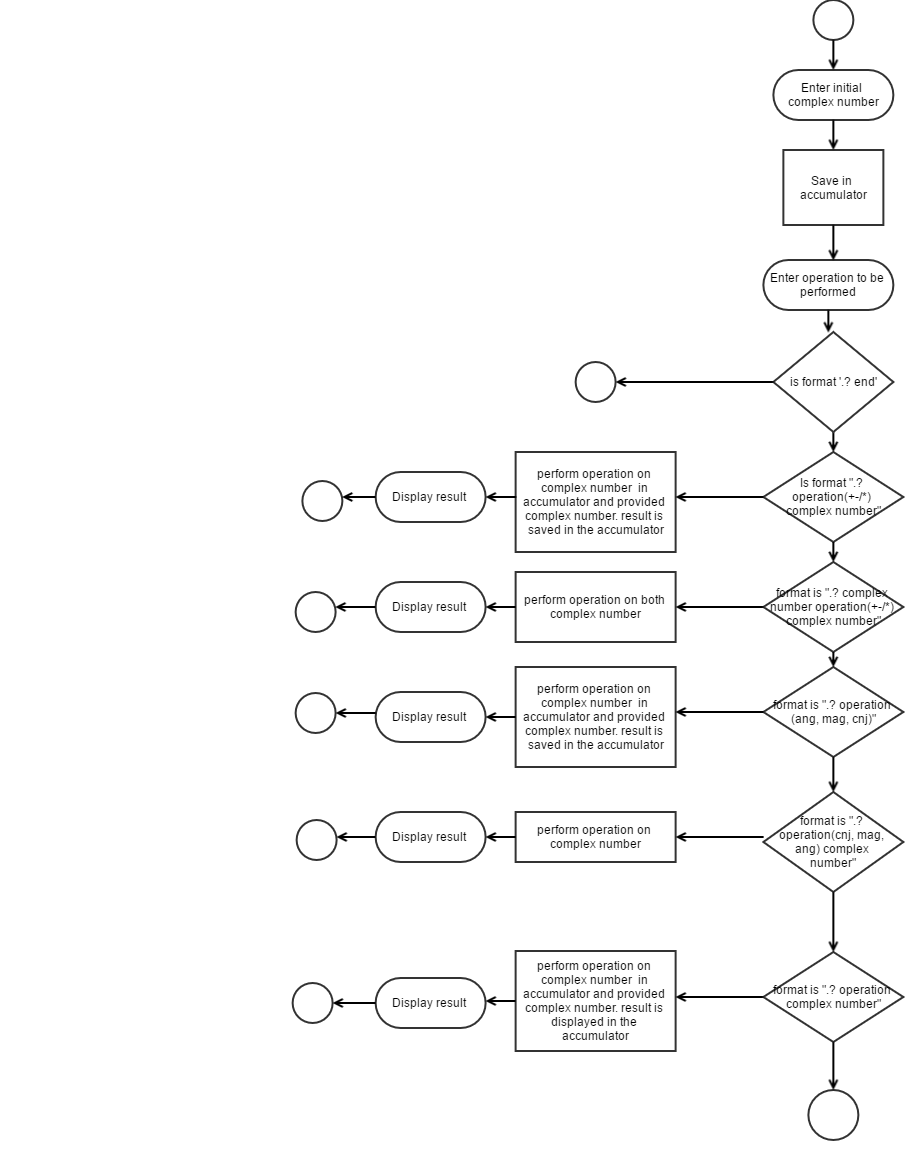
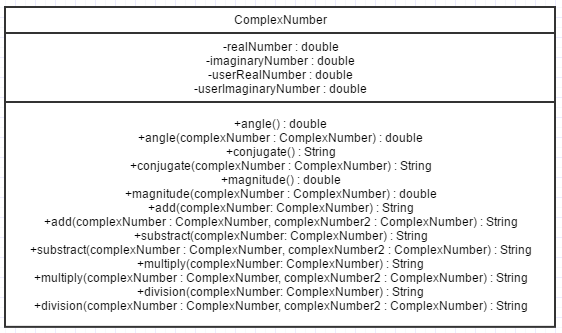
Assignment 3

This is a simple Complex number calculator.

Analysis

Design





Implementation

## Tester.class

**package** com.java.assignment;

**import** java.util.Scanner;

**public** **class** Tester {

//complex number = a + bi i.e realNumber + imaginaryNumber\*i;

**double** realNumber = 0;

**double** imaginaryNumber = 0;

**static** **boolean** *validInitialValue* = **false**;

**static** **boolean** *continueOperation* = **true**;

**public** **static** **void** main(String[] args) {

ComplexNumber complexNumber = **new** ComplexNumber();

Scanner userInput = **new** Scanner(System.***in***);

String[] userOperation = **null**;

String resultOutput = "";

// System.out.println("Welcome to the complex number calculator.");

// System.out.println("=========================================");

//

**while**(!*validInitialValue*){

System.***out***.print("Please enter the initial complex number, Acceptable format is a+bi : ");

**try** {

complexNumber.splitter(userInput.nextLine());

} **catch** (Exception e) {

}

**if**(ComplexNumber.*realNumber* == 0 && ComplexNumber.*imaginaryNumber* == 0){

System.***out***.println("Invalid complex number. Acceptable format is a+bj");

}**else**{

*validInitialValue* = **true**;

}

}

**while** (*continueOperation*) {

System.***out***.println("you can perform any operation. sample format .? add 3.2+j2.4");

userOperation = userInput.nextLine().split(" ");

**if**(userOperation.length <= 1 || userOperation == **null**){

System.***out***.println("Please provide a vaild operation format");

**continue**;

}**else** **if**(userOperation.length == 2){

//End the program

**if**(userOperation[1].equalsIgnoreCase("end")){

System.***out***.println("Goodbye");

*continueOperation* = **false**;

**break**;

}

**if**(userOperation[1].equalsIgnoreCase("mag")){

resultOutput = ComplexNumber.*magnitude*()+"";

}

**if**(userOperation[1].equalsIgnoreCase("cnj")){

resultOutput = ComplexNumber.*conjugate*();

}

**if**(userOperation[1].equalsIgnoreCase("ang")){

resultOutput = ComplexNumber.*angle*()+"";

}

}**else** **if**(userOperation.length == 3){

Tester tester = **new** Tester();

tester.splitter(userOperation[2]);

complexNumber.userRealNumber = tester.realNumber;

complexNumber.userImaginaryNumber = tester.imaginaryNumber;

**if** (userOperation[1].equalsIgnoreCase("add") || userOperation[1].equalsIgnoreCase("+")) {

resultOutput = complexNumber.add(complexNumber);

}

**if** (userOperation[1].equalsIgnoreCase("substract") || userOperation[1].equalsIgnoreCase("-")) {

resultOutput = complexNumber.substract(complexNumber);

}

**if** (userOperation[1].equalsIgnoreCase("multiply") || userOperation[1].equalsIgnoreCase("\*")) {

resultOutput = complexNumber.multiple(complexNumber);

}

**if** (userOperation[1].equalsIgnoreCase("division") || userOperation[1].equalsIgnoreCase("/")) {

resultOutput = complexNumber.divide(complexNumber);

}

**if**(userOperation[1].equalsIgnoreCase("mag")){

resultOutput = complexNumber.magnitude(complexNumber)+"";

}

**if**(userOperation[1].equalsIgnoreCase("cnj")){

resultOutput = complexNumber.conjugate(complexNumber);

}

**if**(userOperation[1].equalsIgnoreCase("ang")){

resultOutput = complexNumber.angle(complexNumber)+"";

}

}**else** **if**(userOperation.length == 4){

Tester tester = **new** Tester();

tester.splitter(userOperation[1]);

complexNumber.userRealNumber = tester.realNumber;

complexNumber.userImaginaryNumber = tester.imaginaryNumber;

ComplexNumber complexNumber2 = **new** ComplexNumber();

tester.splitter(userOperation[3]);

complexNumber2.userRealNumber = tester.realNumber;

complexNumber2.userImaginaryNumber = tester.imaginaryNumber;

**if** (userOperation[2].equalsIgnoreCase("add") || userOperation[2].equalsIgnoreCase("+")) {

resultOutput = complexNumber.add(complexNumber, complexNumber2);

}

**if** (userOperation[2].equalsIgnoreCase("substract") || userOperation[2].equalsIgnoreCase("-")) {

resultOutput = complexNumber.substract(complexNumber, complexNumber2);

}

**if** (userOperation[2].equalsIgnoreCase("multiply") || userOperation[2].equalsIgnoreCase("\*")) {

resultOutput = complexNumber.multiple(complexNumber, complexNumber2);

}

**if** (userOperation[2].equalsIgnoreCase("division") || userOperation[2].equalsIgnoreCase("/")) {

resultOutput = complexNumber.divide(complexNumber, complexNumber2);

}

}

System.***out***.println("The output is " + resultOutput);

System.***out***.println("Enter .? end to quit.");

}

}

/\*\*

\* Split user input

\* **@param** complex

\*/

**public** **void** splitter(String complex){

String[] splittedString = **null**;

**if**(complex.contains("+")){

splittedString = complex.split("\\+");

imaginaryNumber = splittedString != **null** ? Double.*parseDouble*(splittedString[1].replace("j", "").replace("i", "")) : 0;

}

**else** **if**(complex.contains("-")){

splittedString = complex.split("\\-");

imaginaryNumber = -1 \* (splittedString != **null** ? Double.*parseDouble*(splittedString[1].replace("j", "").replace("i", "")) : 0);

}

**else**{

imaginaryNumber = Double.*parseDouble*(complex.replace("j", "").replace("i", ""));

}

realNumber = splittedString != **null** ? Double.*parseDouble*(splittedString[0]) : 0;

}

}

ComplexNumber.class

**package** com.java.assignment;

**public** **final** **class** ComplexNumber {

**public** **static** **double** *realNumber* = 0;

**public** **static** **double** *imaginaryNumber* = 0;

**public** **double** userRealNumber=0;

**public** **double** userImaginaryNumber = 0;

/\*\*

\* Returns the angle of complex number

\* **@return** {@link Double}

\*/

**public** **static** **double** angle(){

**return** Math.*atan*(ComplexNumber.*imaginaryNumber*/ComplexNumber.*realNumber*);

}

/\*\*

\* Returns the angle of the provided complex number

\* **@param** complexNumber

\* **@return**

\*/

**public** **double** angle(ComplexNumber complexNumber){

**return** Math.*atan*(complexNumber.userImaginaryNumber/complexNumber.userRealNumber);

}

/\*\*

\* Returns the conjugate of complex number

\* **@return** {@link Double}

\*/

**public** **static** String conjugate(){

ComplexNumber.*imaginaryNumber* = -1 \* ComplexNumber.*imaginaryNumber*;

**return** ComplexNumber.*realNumber*+(ComplexNumber.*imaginaryNumber* >= 0 ? "+" : "")+ComplexNumber.*imaginaryNumber*+"j";

}

/\*\*

\* Returns the conjugate of the provided complex number

\* **@return** {@link Double}

\* **@param** complexNumber

\*/

**public** String conjugate(ComplexNumber complexNumber){

complexNumber.userImaginaryNumber = -1 \* complexNumber.userImaginaryNumber;

**return** complexNumber.userRealNumber+(complexNumber.userImaginaryNumber >= 0 ? "+" : "")+complexNumber.userImaginaryNumber+"j";

}

/\*\*

\* Return the magnitude of complex number

\* **@return** {@link Double}

\*/

**public** **static** **double** magnitude(){

//r = |sqrt((realnumber \* realnumber) + (imaginaryNumber \* imaginaryNumber))|

**return** Math.*sqrt*( Math.*pow*(ComplexNumber.*realNumber*, 2) + Math.*pow*(ComplexNumber.*imaginaryNumber*, 2));

}

/\*\*

\* Return the magnitude of the provided complex number

\* **@param** complexNumber

\* **@return** {@link Double}

\*/

**public** **double** magnitude(ComplexNumber complexNumber){

//r = |sqrt((realnumber \* realnumber) + (imaginaryNumber \* imaginaryNumber))|

**return** Math.*sqrt*( Math.*pow*(complexNumber.userRealNumber, 2) + Math.*pow*(complexNumber.userImaginaryNumber, 2));

}

/\*\*

\* (2+3j) + (4+7j)

\* (2+4)+(3+7)j

\* 6+10j--Answer

\* Sum complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** String add(ComplexNumber complexNumber){

ComplexNumber.*realNumber* = (ComplexNumber.*realNumber* + complexNumber.userRealNumber);

ComplexNumber.*imaginaryNumber* = (ComplexNumber.*imaginaryNumber* + complexNumber.userImaginaryNumber);

**return** ComplexNumber.*realNumber* +(ComplexNumber.*imaginaryNumber* >= 0 ? "+" : "")+ComplexNumber.*imaginaryNumber*+"j";

}

/\*\*

\* (2+3j) + (4+7j)

\* (2+4)+(3+7)j

\* 6+10j--Answer

\* Sum complex numbers

\* **@param** complexNumber

\* **@param** complexNumber2

\* **@return**

\*/

**public** String add(ComplexNumber complexNumber, ComplexNumber complexNumber2){

complexNumber2.userRealNumber = (complexNumber2.userRealNumber + complexNumber.userRealNumber);

complexNumber2.userImaginaryNumber = (complexNumber2.userImaginaryNumber + complexNumber.userImaginaryNumber);

**return** complexNumber2.userRealNumber +(complexNumber2.userImaginaryNumber >= 0 ? "+" : "")+complexNumber2.userImaginaryNumber+"j";

}

/\*\*

\* Sum complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** **static** String Add(ComplexNumber complexNumber){

**return** **null**;

}

/\*\*

\* (4+7j) - (2+3j)

\* (4+7j) + (-2-3j)

\* Substract complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** String substract(ComplexNumber complexNumber){

ComplexNumber.*realNumber* = (ComplexNumber.*realNumber* - complexNumber.userRealNumber);

ComplexNumber.*imaginaryNumber* = (ComplexNumber.*imaginaryNumber* - complexNumber.userImaginaryNumber);

**return** ComplexNumber.*realNumber* +(ComplexNumber.*imaginaryNumber* >= 0 ? "+" : "")+ComplexNumber.*imaginaryNumber*+"j";

}

/\*\*

\* (4+7j) - (2+3j)

\* (4+7j) + (-2-3j)

\* Substract complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** String substract(ComplexNumber complexNumber, ComplexNumber complexNumber2){

complexNumber2.userRealNumber = (complexNumber2.userRealNumber - complexNumber.userRealNumber);

complexNumber2.userImaginaryNumber = (complexNumber2.userImaginaryNumber - complexNumber.userImaginaryNumber);

**return** complexNumber2.userRealNumber +(complexNumber2.userImaginaryNumber >= 0 ? "+" : "")+complexNumber2.userImaginaryNumber+"j";

}

/\*\*

\* Substract complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** **static** String Substract(ComplexNumber complexNumber){

**return** **null**;

}

/\*\*

\* Multiply complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** String multiple(ComplexNumber complexNumber){

ComplexNumber.*realNumber* = (ComplexNumber.*realNumber* \* complexNumber.userRealNumber);

**if**(ComplexNumber.*imaginaryNumber* == 0){

ComplexNumber.*imaginaryNumber* = complexNumber.userImaginaryNumber;

}**else**{

ComplexNumber.*imaginaryNumber* = -1 \* (ComplexNumber.*imaginaryNumber* \* complexNumber.userImaginaryNumber);

ComplexNumber.*realNumber* = ComplexNumber.*realNumber* + ComplexNumber.*imaginaryNumber*;

ComplexNumber.*imaginaryNumber* = 0;

**return** ComplexNumber.*realNumber*+"";

}

**return** ComplexNumber.*realNumber* +(ComplexNumber.*imaginaryNumber* >= 0 ? "+" : "")+ComplexNumber.*imaginaryNumber*+"j";

}

/\*\*

\* Multiply complex numbers

\* **@param** complexNumber

\* **@param** complexNumber2

\* **@return**

\*/

**public** String multiple(ComplexNumber complexNumber, ComplexNumber complexNumber2){

complexNumber2.userRealNumber = (complexNumber2.userRealNumber \* complexNumber.userRealNumber);

**if**(complexNumber2.userImaginaryNumber == 0){

complexNumber2.userImaginaryNumber = complexNumber.userImaginaryNumber;

}**else**{

complexNumber2.userImaginaryNumber = -1 \* (complexNumber2.userImaginaryNumber \* complexNumber.userImaginaryNumber);

complexNumber2.userRealNumber = complexNumber2.userRealNumber + complexNumber2.userImaginaryNumber;

complexNumber2.userImaginaryNumber = 0;

**return** complexNumber2.userRealNumber+"";

}

**return** complexNumber2.userRealNumber +(complexNumber2.userImaginaryNumber >= 0 ? "+" : "")+complexNumber2.userImaginaryNumber+"j";

}

/\*\*

\* Multiply complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** **static** String Multiple(ComplexNumber complexNumber){

**return** **null**;

}

/\*\*

\* Divide complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** String divide(ComplexNumber complexNumber){

//a+jb /c+ja

//a+jb \* c+ja / c+ja \* c+ja

complexNumber.userRealNumber = (complexNumber.userRealNumber \* complexNumber.userRealNumber);

complexNumber.userImaginaryNumber = -1 \* (complexNumber.userImaginaryNumber \* complexNumber.userImaginaryNumber);

complexNumber.userRealNumber = complexNumber.userRealNumber + complexNumber.userImaginaryNumber;

**return** multiple(complexNumber)+"/"+complexNumber.userRealNumber;

}

/\*\*

\* Divide complex numbers

\* **@param** complexNumber

\* **@param** complexNumber2

\* **@return**

\*/

**public** String divide(ComplexNumber complexNumber2, ComplexNumber complexNumber){

//a+jb /c+ja

//a+jb \* c+ja / c+ja \* c+ja

complexNumber.userRealNumber = (complexNumber.userRealNumber \* complexNumber.userRealNumber);

complexNumber.userImaginaryNumber = -1 \* (complexNumber.userImaginaryNumber \* complexNumber.userImaginaryNumber);

complexNumber.userRealNumber = complexNumber.userRealNumber + complexNumber.userImaginaryNumber;

**return** multiple(complexNumber, complexNumber2)+"/"+complexNumber.userRealNumber;

}

/\*\*

\* Divide complex numbers

\* **@param** complexNumber

\* **@return**

\*/

**public** **static** String Divide(**double** realNumber, **double** imaginaryNumber){

**return** **null**;

}

/\*\*

\* Split user input

\* **@param** complex

\*/

**public** **void** splitter(String complex){

String[] splittedString = **null**;

**if**(complex.contains("+")){

splittedString = complex.split("\\+");

*imaginaryNumber* = splittedString != **null** ? Double.*parseDouble*(splittedString[1].replace("j", "").replace("i", "")) : 0;

}

**else** **if**(complex.contains("-")){

splittedString = complex.split("\\-");

*imaginaryNumber* = -1 \* (splittedString != **null** ? Double.*parseDouble*(splittedString[1].replace("j", "").replace("i", "")) : 0);

}

**else** {

*imaginaryNumber* = Double.*parseDouble*(complex.replace("j", "").replace("i", ""));

}

*realNumber* = splittedString != **null** ? Double.*parseDouble*(splittedString[0]) : 0;

}

}

Output screen shot

