**CIS 368**

**Section 50**

**Assignment 5**

**Michael Stiffler - 2731139**

**Tarik Taye - 2715602**

Problem Description:

Design a class named Complex for representing complex numbers and the methods add, subtract, multiply, divide, abs for performing complex-number operations, and 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. 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. Your Complex class should also implement the Cloneable interface. Write a test program that prompts the user to enter two complex numbers and display the result of their addition, subtraction, multiplication, and division. What is asked of us:

Submit the following items:

1. Compile, Run, and Submit word or pdf file to blackboard (you must submit the program regardless whether it complete or incomplete, correct or incorrect)

2. Submit screen shots of sample run

THE PROGRAM IS COMPLETE

**Diagram**:

Diagram

Description automatically generated

Screenshots:

Sample Run

**Enter the first complex number:**

Graphical user interface, text

Description automatically generated

**Enter the second complex number:**

Graphical user interface, text

Description automatically generated

**Results:**

Text

Description automatically generated

**Code:**

import java.util.Scanner;

import java.lang.Math;

public class Test {

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

        Scanner input = new Scanner(System.in);

        System.out.print("Enter the first complex number: ");

*double* a = input.nextDouble();

*double* b = input.nextDouble();

        Complex c1 = new Complex(a, b);

        System.out.print("Enter the second complex number: ");

*double* c = input.nextDouble();

*double* d = input.nextDouble();

        Complex c2 = new Complex(c, d);

        System.out.println("(" + c1 + ")" + " + " + "(" + c2 + ")" + " = "

                + c1.add(c2));

        System.out.println("(" + c1 + ")" + " - " + "(" + c2 + ")" + " = "

                + c1.subtract(c2));

        System.out.println("(" + c1 + ")" + " \* " + "(" + c2 + ")" + " = "

                + c1.multiply(c2));

        System.out.println("(" + c1 + ")" + " / " + "(" + c2 + ")" + " = "

                + c1.divide(c2));

        System.out.println("|" + c1 + "| = " + c1.abs());

        Complex c3 = (Complex) c1.clone();

        System.out.println(c1 == c3);

        System.out.println(c3.getRealPart());

        System.out.println(c3.getImaginaryPart());

    }

}

class Complex implements Cloneable {

    // Create two private variables real and imaginary holding their respective

    // parts

    private *double* real;

    private *double* imaginary;

    // Constructor to assign real and imaginary

    Complex(*double* *real*, *double* *imaginary*) {

        this.real = *real*;

        this.imaginary = *imaginary*;

    }

    // Constructor to assign real and assign 0 to imaginary

    Complex(*double* *real*) {

        this.real = *real*;

        this.imaginary = 0;

    }

    // Constructor to assign real and imaginary to 0

    Complex() {

        this.real = 0;

        this.imaginary = 0;

    }

    // Returns the value of real as a string

    public String getRealPart() {

        return String.valueOf(this.real);

    }

    // Returns the value of imaginary as a string

    public String getImaginaryPart() {

        return String.valueOf(this.imaginary);

    }

    public Complex add(Complex *c2*) {

        // Create new variables newA and newB by getting the real part of the complex

        // number entered and turning it into a double and adding it to this complex

        // numbers real part. Then doing the same for imaginary

*double* newA = this.real + Double.parseDouble(*c2*.getRealPart());

*double* newB = this.imaginary + Double.parseDouble(*c2*.getImaginaryPart());

        // Return a new complex number with the newA and newB

        return new Complex(newA, newB);

    }

    public Complex subtract(Complex *c2*) {

        // Create new variables newA and newB by getting the real part of the complex

        // number entered and turning it into a double and subtracting it to this

        // complex numbers real part. Then doing the same for imaginary

*double* newA = this.real - Double.parseDouble(*c2*.getRealPart());

*double* newB = this.imaginary - Double.parseDouble(*c2*.getImaginaryPart());

        // Return a new complex number with the newA and newB

        return new Complex(newA, newB);

    }

    public Complex multiply(Complex *c2*) {

        // (a + bi)\*(c + di) = (ac −bd) + (bc + ad)i

        // Follow the equation above, I split the equation into their respective parts.

        // Assigning ac, bd, bc, and ad. Once these are assigned I calculate the left

        // and right side of the equation

*double* newAC = this.real \* Double.parseDouble(*c2*.getRealPart());

*double* newBD = this.imaginary \* Double.parseDouble(*c2*.getImaginaryPart());

*double* newBC = this.imaginary \* Double.parseDouble(*c2*.getRealPart());

*double* newAD = this.real \* Double.parseDouble(*c2*.getImaginaryPart());

*double* left = newAC - newBD;

*double* right = newBC + newAD;

        // Return a new complex number with the left and right side of the equation

        // calculated

        return new Complex(left, right);

    }

    public Complex divide(Complex *c2*) {

        // (ac + bd) / (c^2 + d^2) + (bc − ad)i / (c^2 + d^2)

        // Same as multiply, I calculate each ac, bd, c^2, d^2, bc, and ad.

*double* newAC = this.real \* Double.parseDouble(*c2*.getRealPart());

*double* newBD = this.imaginary \* Double.parseDouble(*c2*.getImaginaryPart());

*double* newBC = this.imaginary \* Double.parseDouble(*c2*.getRealPart());

*double* newAD = this.real \* Double.parseDouble(*c2*.getImaginaryPart());

*double* cSquared = Math.pow(Double.parseDouble(*c2*.getRealPart()), 2);

*double* dSquared = Math.pow(Double.parseDouble(*c2*.getImaginaryPart()), 2);

        // Once I calculate the left and right side I return a new complex object with

        // that left and right

*double* left = (newAC + newBD) / (cSquared + dSquared);

*double* right = (newBC - newAD) / (cSquared + dSquared);

        return new Complex(left, right);

    }

    @*Override*

    // Return the real part and imaginary part in the structure (real +

    // imaginary(i))

    public String toString() {

        return this.real + " + " + this.imaginary + "i";

    }

    public String abs() {

        // Calculate a^2 and b^2 and return the sqrt of that as a string

*double* aSquared = Math.pow(this.real, 2);

*double* bSquared = Math.pow(this.imaginary, 2);

        return String.valueOf(Math.sqrt(aSquared + bSquared));

    }

    @*Override*

    protected Object clone()

            throws CloneNotSupportedException {

        // Return the cloned version of this object

        return super.clone();

    }

}