**The Complex class:**

Devise a class for the addition, subtraction, multiplication, and division of complex numbers. The class should also contain a method to display complex numbers. A complex number is represented as (A + iB), where A is the real part, B is the imaginary part, and i = sqrt(-1). The following expressions show how addition/subtraction can be performed on two complex numbers.

Text

Description automatically generated

Text, letter

Description automatically generated

The methods for adding and subtracting should be static as given below.

1. static Complex addComplex(Complex A, Complex B);
2. static Complex subtractComplex(Complex A, Complex B);

Each method takes input 2 complex numbers and returns the resultant complex number object.

Also, create an instance method ‘modulus’ for computing the modulus of a complex number. The modulus of a complex number is computed as follows: |z| = √(x2 + y2), where x and y are real and imaginary parts of the complex number.

Assume a complex number with real part=5 and imaginary part = 3. You have to create a **toString** method, which returns the following string for such a complex number:

(5+3i)

**The ComplexPolynomial class:**

This problem extends the problem you solved in Lab Assignment 1. For your reference, the solution to Lab Assignment 1 is also uploaded.

In the polynomial class you created in Lab Assignment 1, you used an array of integers to represent the coefficients of a polynomial. Now, create a class named ComplexPolynomial, so that it has an array of complex numbers as coefficients of polynomials (instead of array of simple integers), i.e. Complex[] coefficients;. E.g. an example complex polynomial p1 is given below:

* (5+5i)\*x + (4+3i)\*x^2
  + Here, first coefficient is the complex number (5+3i) (degree 1) and second coefficient is the complex number 4+3i (degree 2).

Create the following constructor for the ComplexPolynomial class:

* ComplexPolynomial(Complex []); // The constructor simply receives an array of complex numbers and assigns to ‘coefficients’ array. A shallow copy is allowed here.
* Create a **toString** method which displays the complex polynomial (e.g. p1 polynomial above) in the following format.
  + P(x) = (5+5i)\*x + (4+3i)\*x^2
* Create the following 2 overloaded versions of **addComplexPolynomial** methods, which adds 2 complex polynomials:
  + void addComplexPolynomials(ComplexPolynomial p2); // updates the calling object
    - adds complex polynomial p2 to the calling complex polynomial object. Note that the add method should now add ‘complex numbers coefficients’ (using the add method of the Complex class) rather than simple numeric coefficients.
  + static ComplexPolynomial addComplexPolynomials(ComplexPolynomial p1, ComplexPolynomial p2);
    - a static class method which adds 2 complex polynomials p1 and p2 and returns the resultant complex polynomial.
      * evaluate(x): returns a value (double) of the ComplexPolynomial for the given value of x. This method should use the modulus of each complex number coefficient, while computing the polynomial value.
        + For example, for the ComplexPolynomial example **p1:** (5+5i)\*x + (4+3i)\*x^2, if we call p1.evaluate(2); Then, the method should return 34.14 (i.e. 7.07 \* 2 + 5 \* 2^2 = 14.14 + 20 = 34.14). Note that 7.07 is the modulus of complex coefficient 5+5i and 5 is the modulus of the complex coefficient 4+3i.

**The ComplexPolynomialTest class:**

* Inside the Test class main method:
  + Create a ComplexPolynomial object p1 for the polynomial in above example. Then, call the evaluate method and display the result.
  + Create another polynomial object p2 with complex coefficients of your choice. Then, demonstrate the usage of addPolynomial methods (both variants) by calling them with p1 and p2 as actual arguments. Display p1, p2 and the resultant polynomial in each case using the toString method.