Introduction to Java

CS9053 Section I

Thursday 6:00 PM – 8:30 PM

Prof. Dean Christakos

Sept. 18th, 2025

Due: Sept. 26th, 2025 11:59 PM

Part I – Creating Objects

1. ComplexNumber: In the lecture you have seen the creation of a circle. Here you are going to create a complex number. As you will remember a complex number as a real part, **a**, and an imaginary part, **b**, given by .

The magnitude of a Complex Number is given by

Addition and subtraction are performed by adding/subtracting the real part with the real part and then the imaginary part with the imaginary part.

Multiplying two imaginary numbers and is given by:

You will create a class ComplexNumber using the following UML:

A screenshot of a computer

AI-generated content may be incorrect.

**In standard UML parlance, “+” indicates that a field or method is public and “-“ indicates that a field or method is private. An underlined field or method indicates it is static.**

Create two complex numbers, *7.5 + i4.2* and *8.2 + i9.4* and add then, subtract them, and multiply them.

toString() should return a String that says <**real> + i<imaginary>** where <real> and <imaginary> are the values of the real and imaginary fields.

1 Points Extra credit: implement ComplexNumber divide(ComplexNumber divisor)

1. Objects and Arrays of Objects

Your objective is to develop the **Student** and **Gradebook** classes and to use their methods. The classes are described below to guide you.

1. **Student**

The **Student** class contains data related to a student. Here is the UML:

A screenshot of a computer

AI-generated content may be incorrect.

**TASKS / Requirements:**

1. **Student Constructor** 🡪
   1. The grades array should have a length of x (in this case 5) and be initialized with the values of -1, indicating that there is no grade for that index yet.
   2. The constructor should automatically generate an ID.
2. **getGrades()** 🡪 should return a **copy** of the grades array, not the original grades array.
3. **setGrades(assignmentIndex, grade)** 🡪
   1. should set the grade of an existing index of the grades array with a grade value of between 0-100 (inclusive).
   2. Error Handling 🡪If the call violates either of those constraints, it should do nothing.
4. **getAverage()** 🡪
   1. should calculate the average of the existing grades in the grades array for a given student
   2. If there are only a few grades for a student then 🡪 -1 values do not count towards the average.
5. **Gradebook**

Here you are going to create a Gradebook class that contains an array of Student objects. It also has the following methods:

Requirements:

1. **void addStudent(Student s)** 🡪 add a student to the first available null slot
2. **Student findById(int id)** 🡪 return the student or null if not found
3. **Student getTopStudent()** 🡪 return the student with the highest average
4. **void printAll()** 🡪 print info for all Students
5. In the **main method**, you will:
   1. Create a Gradebook with space for 5 students
   2. Create and add 5 Student objects
   3. Set random grades for each assignment (from 0-100)
   4. Print all students
   5. Print the top student

The UML for the Gradebook is:

A screenshot of a computer

AI-generated content may be incorrect.