# CNIT 25500: Lab #4 – Classes

## (Part A: 10 points, Part B: 15 points)

- **Part A** is due during lab sessions and checked off by your TA **(10 pts).**
- **Part A** handwritten assignment is due at the end of lab sessions **(when not submitted -5 pts).**
- **Part B** is due **Thursday, February 27 at 11:59 p.m. (15 pts).**
- **Part B** handwritten assignment is due at the start of the following lab sessions **(when not submitted -5 pts).**

## Objectives:

- Understand the basics of OOP: Classes, Objects, and Encapsulation.
- Learn about constructors, instance variables, methods, and object interactions.
- Practice creating and using objects across multiple classes.

## Part A:

### 1. Employee and Payroll System (10 pts)

**You will build a Payroll System that consists of three classes:**
1. **Employee** – Stores employee details (name, salary, seniority).
2. **Payroll** – Calculates salary bonuses.
3. **MainClass** – Runs the program and displays results.

#### Step 1: Create the Employee Class

(1). Define private instance variables:

- name (String)
- salary (double)
- seniority (int)

(2). Create a constructor to initialize these fields.

(3). Implement accessors and mutators to retrieve and modify employee details.

(4). Write a displayEmployee() method to print employee details.

#### Step 2: Create the Payroll Class

(1). Implement a method to apply an seniority-based bonus (applySeniorityBonus())

Example Output:

```
Applying Experience Bonus ($500 per year)...
Bonus Amount: $1000.0
Final Salary After Bonus: $76000.0
```

#### Step 3: Create the MainClass

(1). Create an Employee object.

(2). Use a Payroll object to apply a bonus.

(3). Display the updated employee details.

Final Program Output Example:

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/C
Employee Name: Alice
Years of seniority: 2
Salary: $75000.0

Applying Experience Bonus ($500 per year)...
Bonus Amount: $1000.0
Final Salary After Bonus: $76000.0

Employee Name: Alice
Years of seniority: 2
Salary: $76000.0
```

## Part B:

**Programming Assignment**

**1. Student and Course Management System (15 pts)**

In this assignment, you will create three classes that work together to manage students and their course enrollments. You will also practice using arrays and loops to handle multiple students.

Classes to Create:

(1). Student: Stores student details (name, ID, GPA). Should contain a displayStudent() method to print student information. Example Output:

*Student Name: John Doe*
*Student ID: 12345*
*GPA: 3.8*

(2). Course: Stores course name, credits, and assigns students. Should contain a displayCourseInfo() method to print course details and enrolled students, and a method enrollStudent(Student student) to add a student to the course. Example Output:

*Course: Java Programming*
*Credits: 3*
*Enrolled Students:*
*1. John Doe (ID: 12345, GPA: 3.8)*

(3). MainClass: Registers multiple students and assigns them to a course.
Create an array of three students (Student[3]). Use a loop to initialize and assign values to all students. Enroll all students in a course and display their details. Example Output:

*Course: Java Programming*
*Credits: 3*
*Enrolled Students:*
*1. Student1 (ID: 1000, GPA: 3.5)*
*2. Student2 (ID: 1001, GPA: 3.6)*
*3. Student3 (ID: 1002, GPA: 3.7)*

**Handwritten Assignment**

For this assignment, you are required to draw a UML (Unified Modeling Language) (Chapter 4.1) class diagram for the three classes you designed in Part A. Your hand-drawn diagram should clearly represent the structure and relationships between the classes.