EXPERIMENT NO. 3.3

Aim

To implement a class hierarchy in Java with a base class `Person` and derived classes `Student` and `Teacher` using the principles of Object-Oriented Programming.

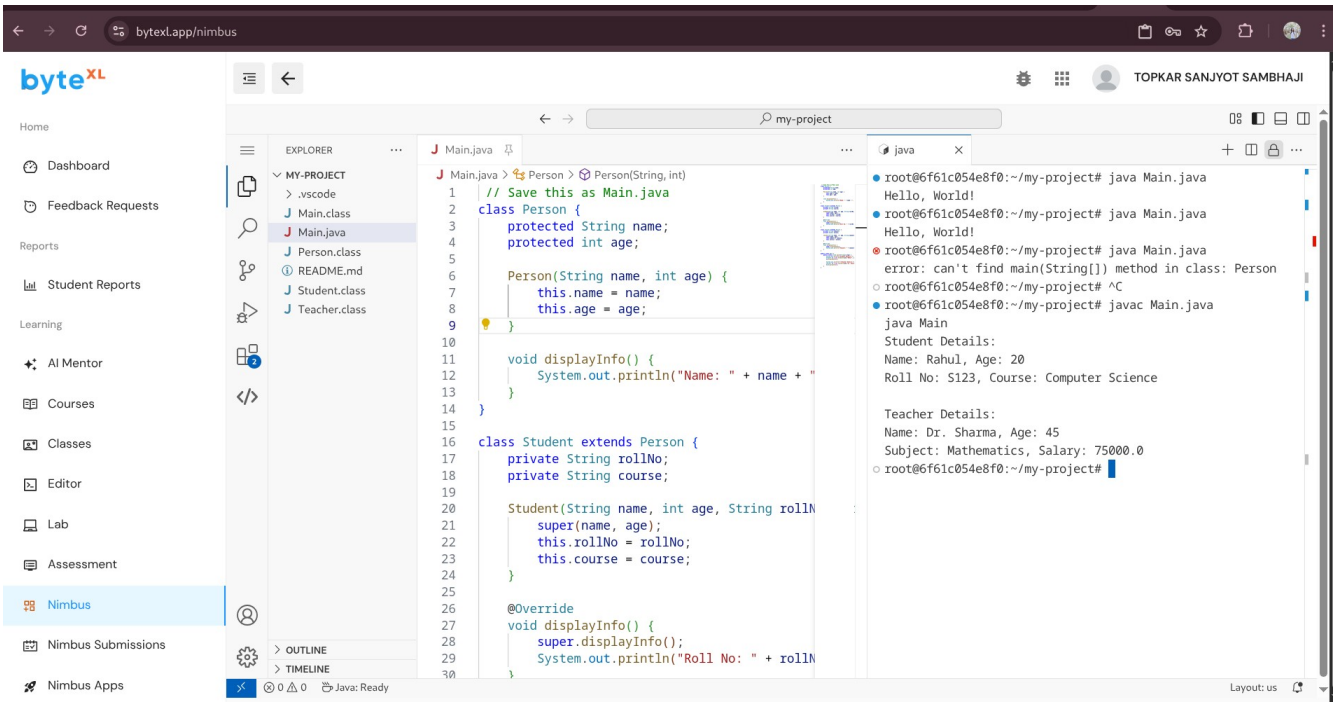
Theory

- **Object-Oriented Programming (OOP)** organizes code into classes and objects.
- **Inheritance** allows a child class to reuse attributes and methods of a parent class.
- **Polymorphism** allows overriding methods (e.g., `displayInfo()`) in subclasses.
- **Java Class Hierarchy Example:**
    - `Person` (parent class: name, age)
    - `Student` (child class: rollNo, course)
    - `Teacher` (child class: subject, salary)

Code (Java)

```
// Base class
class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void displayInfo() {
        System.out.println("Name: " + name + ", Age: " + age);
    }
}

// Derived class: Student
class Student extends Person {
    String rollNo;
    String course;

    Student(String name, int age, String rollNo, String course) {
        super(name, age); // calling parent constructor
        this.rollNo = rollNo;
        this.course = course;
    }
```

```
    @Override
    void displayInfo() {
        super.displayInfo();
        System.out.println("Roll No: " + rollNo + ", Course: " + course);
    }
}

// Derived class: Teacher
class Teacher extends Person {
    String subject;
    double salary;

    Teacher(String name, int age, String subject, double salary) {
        super(name, age);
        this.subject = subject;
        this.salary = salary;
    }

    @Override
    void displayInfo() {
        super.displayInfo();
        System.out.println("Subject: " + subject + ", Salary: " + salary);
    }
}

// Main class to run the program
public class Main {
    public static void main(String[] args) {
        System.out.println("Student Details:");
        Student s1 = new Student("Rahul", 20, "S123", "Computer Science");
        s1.displayInfo();

        System.out.println("\nTeacher Details:");
        Teacher t1 = new Teacher("Dr. Sharma", 45, "Mathematics", 75000);
        t1.displayInfo();
    }
}
```



Output

```
Student Details:
Name: Rahul, Age: 20
Roll No: S123, Course: Computer Science

Teacher Details:
Name: Dr. Sharma, Age: 45
Subject: Mathematics, Salary: 75000.0
```

Learning Outcomes

1. Learned how to implement **class hierarchy** in Java.
2. Understood **inheritance** and **method overriding** using `super`.
3. Differentiated between common attributes (Person) and specialized attributes (Student, Teacher).
4. Gained practical skills for writing **clean, reusable, and modular code**.
5. Understood how class hierarchies apply to **real-world full stack projects** (users, admins, etc.).