1. Write a java program to create a software solution to manage different types of electronic devices in a store. The store sells various electronic devices, and each device has different functionalities, like powering on and connecting to a network. Implement the following requirements:

    a. **Interface PoweredDevice**: represents any device that requires power to function. This interface should have methods **powerOn()** to power on the device and **powerOff()** to power off the device.

    b. **Interface NetworkEnabled** represents any device that can connect to a network. This interface should have methods **connectToNetwork(String networkName)** to connect the device to a specified network and **disconnectFromNetwork()** to disconnect the device from the network.

    c. **Class SmartPhone**: class should implement both the **PoweredDevice** and **NetworkEnabled** interfaces. It should have attributes *model* to store the model name, *isPoweredOn* to indicate if the phone is powered on, *connectedNetwork* to store the name of the connected network. This class must implement all the methods in the PoweredDevice and NetworkEnabled interface and must contain a *toString()* method to return a string representation of the phone's model, power state, and connected network (if any).

    d. Write a main method that simulates the usage of a SmartPhone in the store. In this method, create an instance of SmartPhone with a sample model name. Then call the powerOn(), connectToNetwork(), and disconnectFromNetwork() methods in sequence to demonstrate the device's capabilities. Finally, use the toString() method to print out the phone's details, verifying that the software solution is ready to help manage the store's inventory of electronic devices effectively.

2. Rohit has been tasked with creating an online platform for educational institutions to function seamlessly during any pandemic.The platform should manage users such as Student and Faculty and allow for remote learning and virtual classrooms.

    In this platform:

    - Faculty can conduct live sessions ,upload pre-recorded lectures and assignments.

- Students can attend live sessions, view recorded lectures, and submit assignments online.

Based on the class diagram given below and the described scenario,develop a Java Program.

Sample Input and output

Input: The **name**, **email**, and **user ID** for both `Student` and `Faculty` are hardcoded.

Output:

For student case

Portal Access:

- Viewing online courses.

- Attending live session.

- Submitting assignments online.

For Faculty case

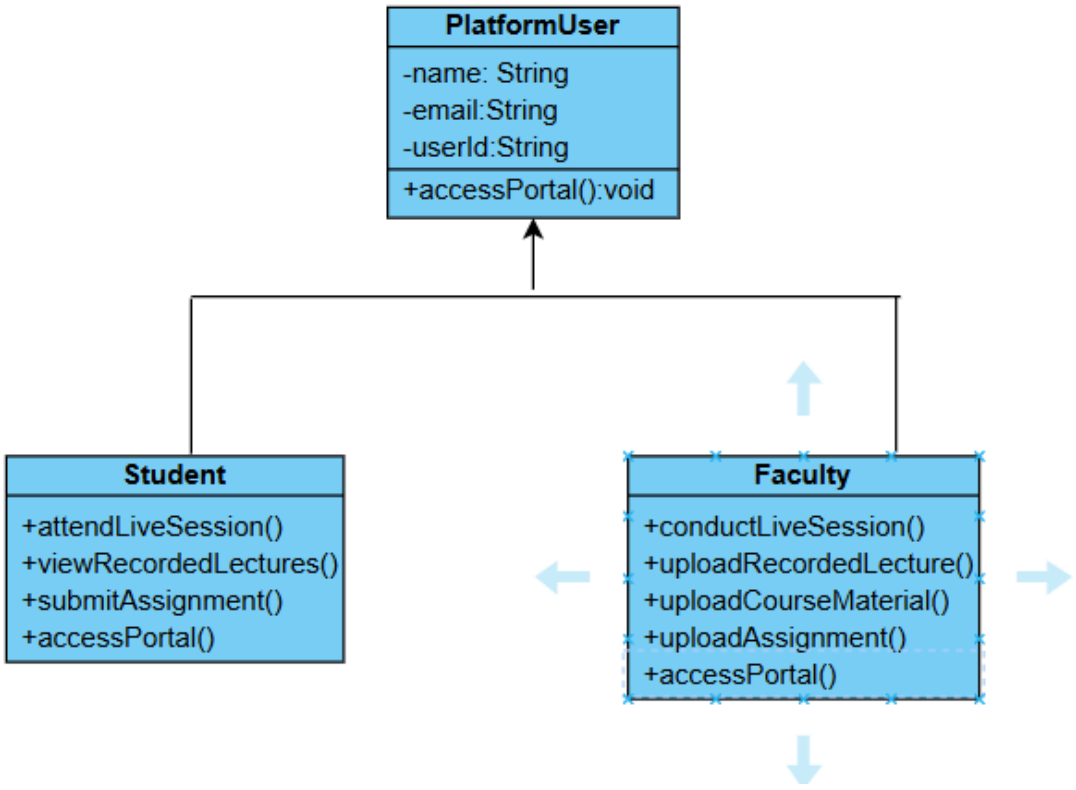
-- Faculty Access --

Name: Dr. Smith
Email: smith@university.com
UserID: FAC001

Portal Access:
- Uploading course material.
- Conducting live online lecture.

Conducting live session.
Uploading course material.
Uploading an online assignment for students.

Class Diagram



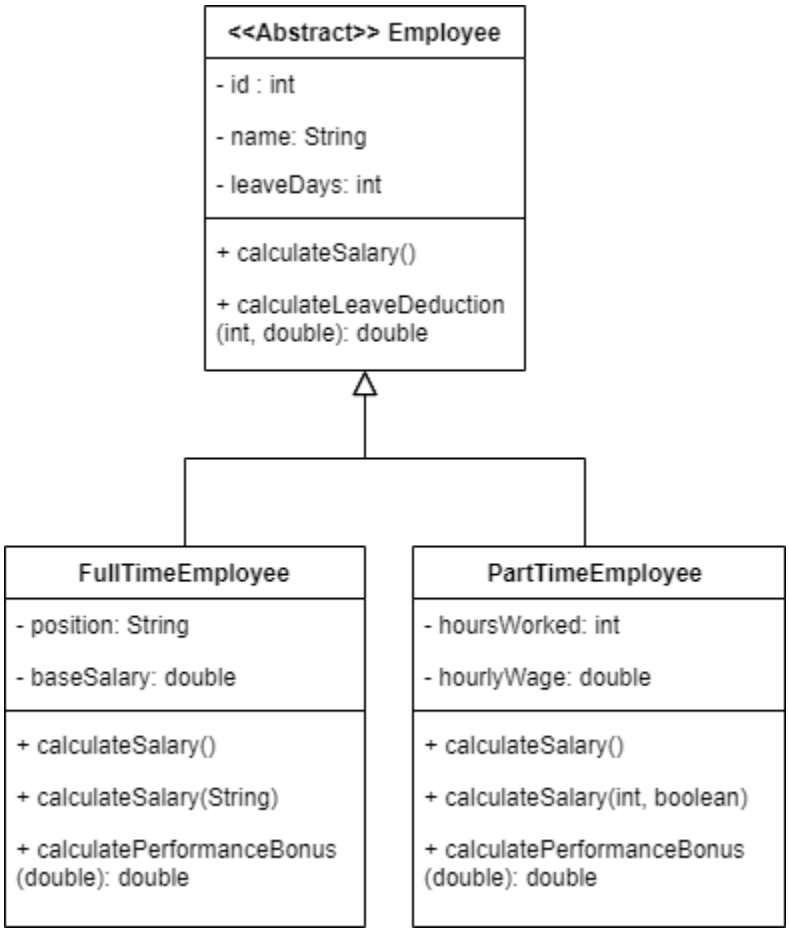
3. You are tasked to design a system for managing employee data in an organization. Create an abstract class `Employee` that contains a method `calculateSalary()` to be implemented by its subclasses.

   Define a class `FullTimeEmployee` where the salary calculation varies based on the employee's position (e.g., Manager or Staff), and it should also account for any leave days taken by the employee. Add functionality to calculate a performance-based bonus for full-time employees based on their performance score.

   Similarly, define a class `PartTimeEmployee`, where salary calculation depends on the number of hours worked and the hourly wage. Additionally, ensure it handles salary deductions for leave days, and provide an option to calculate a bonus if the employee has worked beyond a certain threshold. A performance-based bonus system should also be added for part-time employees, based on their performance score.

Ensure that both classes manage the employee's leave days and implement all relevant methods and attributes for employee details.

CLASS DIAGRAM



#Main

```
public class Main {

    public static void main(String[] args) {

        // Full-time employee example with leave deduction and performance bonus

        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee("Alice", 101, "Manager", 50000, 2); // 2 leave days

        System.out.println("Full-Time Salary (Manager, after leave deduction): " + fullTimeEmployee.calculateSalary("Manager"));

        System.out.println("Full-Time Performance Bonus: " + fullTimeEmployee.calculatePerformanceBonus(92));
```

```
        // Part-time employee example with leave deduction and performance bonus

        PartTimeEmployee partTimeEmployee = new PartTimeEmployee("Bob", 102, 45, 20, 1); //
1 leave day

        System.out.println("Part-Time Salary (with bonus, after leave deduction): " +
partTimeEmployee.calculateSalary(45, true));

        System.out.println("Part-Time Performance Bonus: " +
partTimeEmployee.calculatePerformanceBonus(87));

    }

}
```

Sample output

```
Full-Time Salary (Manager, after leave deduction): 47000.0

Full-Time Performance Bonus: 2000.0

Part-Time Salary (with bonus, after leave deduction): 850.0

Part-Time Performance Bonus: 500.0
```