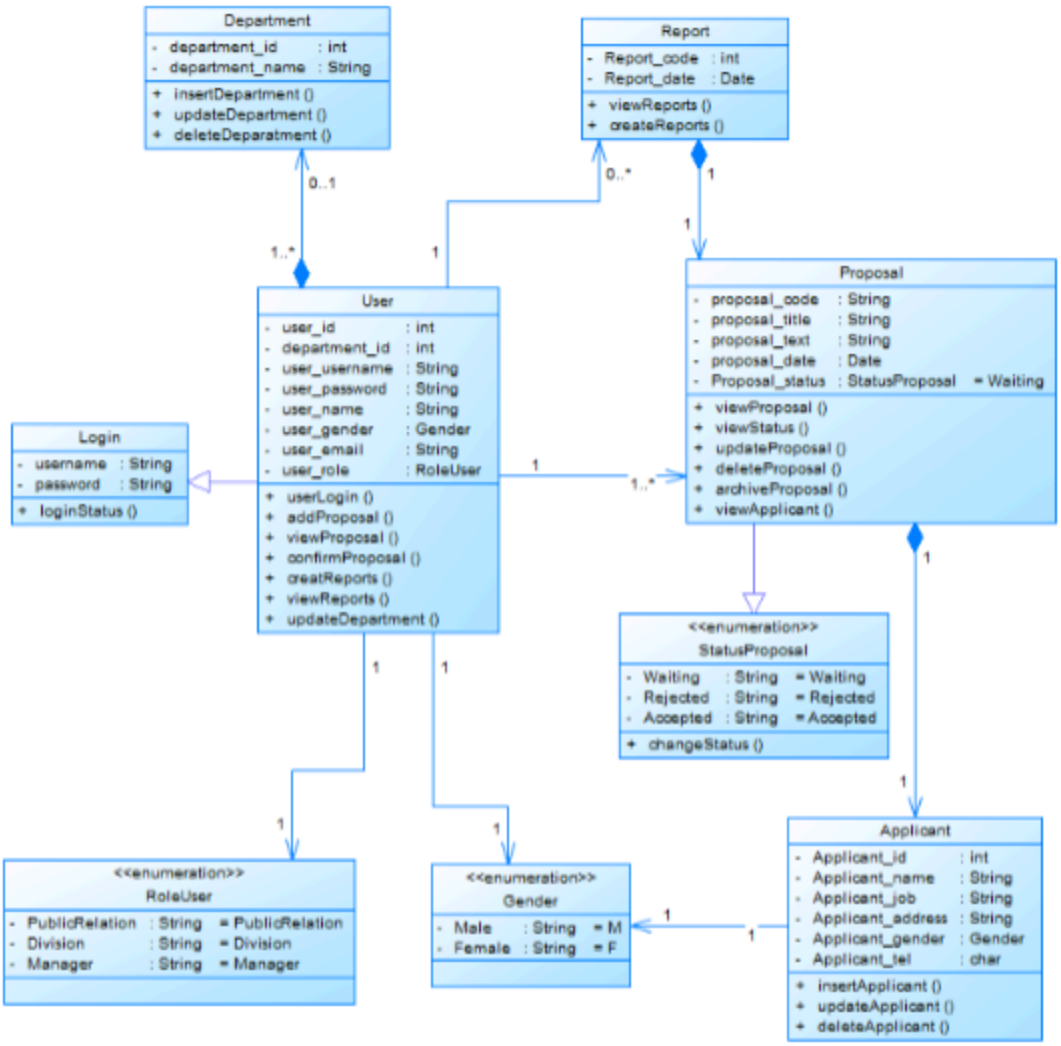

จากภาพที่กำหนด ให้นักศึกษาศึกษา Class Diagram ในภาพ

1. อธิบายหน้าที่ของแต่ละคลาส (Class Responsibility) ว่าใช้ทำอะไรในระบบ
2. อธิบาย **ความหมายของทุก attribute** (ชนิดข้อมูล + บทบาท/หน้าที่)
3. อธิบาย **ความหมายของทุก method** (ทำงานอะไร, รับ/คืนค่าอะไร "คาดเดาได้" จากชื่อเมธอด)
4. อธิบาย **ความสัมพันธ์ (Relationship) ระหว่างคลาสทั้งหมด** โดยต้องระบุประเภทความสัมพันธ์: Inheritance / Association / Composition

Cardinality (เช่น 1, 0..1, 1.., 0..) และแปลความหมายเป็นประโยคภาษาไทย

คำอธิบายแต่ละคลาส (Attributes / Methods)

A) Department

หน้าที่: เก็บข้อมูลหน่วยงาน/แผนกที่ผู้ใช้สังกัด

Attributes

- department_id : int → รหัสแผนก (Primary key)
- department_name : String → ชื่อแผนก

Methods

- insertDepartment() → เพิ่มข้อมูลแผนกใหม่
- updateDepartment() → แก้ไขข้อมูลแผนก
- deleteDepartment() → ลบข้อมูลแผนก

B) Login

หน้าที่: เก็บข้อมูลขั้นต่ำสำหรับการเข้าสู่ระบบ (บัญชีผู้ใช้)

Attributes

- username : String → ชื่อผู้ใช้สำหรับล็อกอิน
- password : String → รหัสผ่าน

Methods

- loginStatus() → ตรวจสอบ/คืนค่าสถานะการเข้าสู่ระบบ (เช่น สำเร็จ/ไม่สำเร็จ)

C) User

หน้าที่: ผู้ใช้งานระบบ (เช่น เจ้าหน้าที่, ประชาสัมพันธ์, ผู้จัดการ)

Attributes

- user_id : int → รหัสผู้ใช้ (Primary Key)
- department_id : int → รหัสแผนก

- user_username : String → ชื่อบัญชีผู้ใช้
- user_password : String → รหัสผ่าน
- user_name : String → ชื่อ-นามสกุล
- user_gender : Gender → เพศ (อ้างถึง enum Gender)
- user_email : String → อีเมล
- user_role : RoleUser → บทบาทผู้ใช้ (อ้างถึง enum RoleUser)

Methods

- userLogin() → กระบวนการล็อกอินของผู้ใช้
- addProposal() → สร้าง/เพิ่ม Proposal
- viewProposal() → ดูรายการ Proposal
- confirmProposal() → ยืนยัน/อนุมัติ/พิจารณา Proposal (ตามสิทธิ์)
- createReports() → สร้าง Report
- viewReports() → ดู Report
- updateDepartment() → แก้ไขข้อมูลแผนก (มักทำได้เฉพาะบาง Role)

D) Proposal

หน้าที่: เก็บข้อมูล "ข้อเสนอ/คำขอ" ที่ผู้ใช้ยื่น/บันทึกในระบบ

Attributes

- proposal_code : String → รหัสข้อเสนอ
- proposal_title : String → ชื่อเรื่องข้อเสนอ
- proposal_text : String → รายละเอียดข้อเสนอ
- proposal_date : Date → วันที่สร้าง/ยื่นข้อเสนอ
- Proposal_status : StatusProposal = Waiting → สถานะ (enum StatusProposal) ค่าเริ่มต้น Waiting

Methods

- viewProposal() → ดูรายละเอียด Proposal
- viewStatus() → ดูสถานะปัจจุบัน
- updateProposal() → แก้ไข Proposal
- deleteProposal() → ลบ Proposal
- archiveProposal() → เก็บถาวร/ย้ายเข้าคลัง
- viewApplicant() → ดูข้อมูลผู้สมัคร/ผู้ยื่น (Applicant) ที่ผูกกับ Proposal

E) Report

หน้าที่: รายงานที่ถูกสร้างจากข้อมูลในระบบ (เช่น สรุปข้อเสนอ)

Attributes

- Report_code : int → รหัสรายงาน
- Report_date : Date → วันที่ออกรายงาน

Methods

- viewReports() → ดูรายงาน
- createReports() → สร้างรายงานใหม่

F) Applicant

หน้าที่: ข้อมูลผู้สมัคร/ผู้ยื่นคำขอ (ผูกกับ Proposal)

Attributes

- Applicant_id : int → รหัสผู้สมัคร
- Applicant_name : String → ชื่อผู้สมัคร
- Applicant_job : String → อาชีพ/ตำแหน่ง

- Applicant_address : String → ที่อยู่
- Applicant_gender : Gender → เพศ (enum Gender)
- Applicant_tel : char → เบอร์โทร (ในงานจริงควรเป็น String มากกว่า)

Methods

- insertApplicant() → เพิ่มผู้สมัคร
- updateApplicant() → แก้ไขข้อมูลผู้สมัคร
- deleteApplicant() → ลบผู้สมัคร

3) อธิบาย Enum ทั้งหมด

RoleUser (กำหนดบทบาทผู้ใช้)

- PublicRelation
- Division
- Manager

Gender

- Male : String = M
- Female : String = F

StatusProposal

- Waiting : String = Waiting
- Rejected : String = Rejected
- Accepted : String = Accepted
- changeStatus() → เปลี่ยนสถานะ (แนวคิด: จาก Waiting → Accepted/Rejected)

4) ความสัมพันธ์ระหว่างคลาส (Relationship + Cardinality)

4.1 User — Department

- Cardinality ที่อ่านจากภาพ:
  - **Department 1 แผนก มี User 1..*** (ผู้ใช้หลายคน)

    - **User 1 คน** สังกัดได้ **Department 0..1** (อาจยังไม่กำหนดแผนกก็ได้)
- ความหมายเชิงระบบ: ผู้ใช้ส่วนใหญ่ต้องมีแผนก แต่อนุญาตให้บางบัญชี "ยังไม่สังกัด" ได้

4.2 User สืบทอดจาก Login (Inheritance)

- ความหมาย: User เป็น "ชนิดหนึ่งของ" Login
  ผู้ใช้ทุกคนมี username/password และพฤติกรรมพื้นฐานของการเข้าสู่ระบบ

4.3 User — Proposal

- Cardinality: **User 1 คน** สร้าง/ดูแล **Proposal 1..***
- และ **Proposal 1 รายการ** มีผู้สร้าง/ผู้รับผิดชอบ **User 1 คน**

4.4 User — Report

- Cardinality: **User 1 คน** สร้างได้ **Report 0..*** (อาจยังไม่สร้างรายงาน)
- **Report 1 รายการ** ผูกกับ **User 1 คน** (ผู้สร้างรายงาน)

4.5 Report — Proposal (มีสัญลักษณ์เพชรทึบ = Composition ในภาพ)

- Cardinality: **1 ต่อ 1** (ตามที่แสดง "1" ทั้งสองด้าน)
- แปลความหมายแบบตรงภาพ: รายงาน 1 ฉบับประกอบด้วยข้อมูลจาก Proposal 1 รายการ และรายงานนั้นผูกแน่นกับ Proposal

4.6 Proposal — Applicant (Composition)

- Cardinality: **1 ต่อ 1**
- ความหมาย: Proposal 1 รายการ มี Applicant 1 คน และ Applicant จะอยู่ได้ก็ต่อเมื่อมี Proposal ที่ผูกอยู่ (ลบ Proposal → Applicant หายตาม)

4.7 การใช้ Enum

- User.user_gender และ Applicant.Applicant_gender ใช้ Gender
- User.user_role ใช้ RoleUser
- Proposal.Proposal_status ใช้ StatusProposal

1) วัตถุประสงค์ของระบบ

- ระบบนี้ถูกออกแบบมาเพื่อจัดการ **ข้อเสนอ (Proposal)** และการออก **รายงาน (Report)** โดยมีการแบ่งสิทธิ์ผู้ใช้งานตามแผนกและบทบาทหน้าที่ เพื่อให้การบริหารจัดการข้อมูลเป็นไปอย่างมีระเบียบ

2) อธิบายหน้าที่ของแต่ละคลาส (Class Responsibility)

- **Department หน้าที่:**
  เก็บข้อมูลหน่วยงาน/แผนกในองค์กร เพื่อระบุว่าผู้ใช้แต่ละคนสังกัดแผนกใด
- **Login หน้าที่:**
  จัดการข้อมูลพื้นฐานที่จำเป็นต่อการเข้าสู่ระบบ **(Authentication) เช่น username และ password**
- **User หน้าที่:**
  แทนผู้ใช้งานระบบ เช่น เจ้าหน้าที่ ประชาสัมพันธ์ หรือผู้จัดการ
  เป็นศูนย์กลางของระบบ สามารถจัดการ **Proposal, Report** และข้อมูลแผนก (ตามสิทธิ์)
- **Proposal หน้าที่:**
  เก็บข้อมูลข้อเสนอ/คำขอที่ผู้ใช้สร้างขึ้นในระบบ พร้อมสถานะการพิจารณา
- **Report หน้าที่:**
  รายงานสรุปข้อมูลในระบบ เช่น รายงานสรุป **Proposal** เพื่อการบริหารหรือการตัดสินใจ
- **Applicant หน้าที่:**
  เก็บข้อมูลผู้สมัครหรือผู้ยื่นคำขอที่เกี่ยวข้องกับ **Proposal**

3. การวิเคราะห์แอตทริบิวต์และเมธอด (Attributes & Methods Analysis)

3.1 ข้อมูลผู้ใช้งาน (User & Login)

- **Attributes:** มีการเก็บรหัสพนักงาน (`user_id`), ชื่อบัญชีและรหัสผ่านที่สืบทอดมาจาก Login , รวมถึงการระบุเพศและบทบาทผ่านระบบ Enum
- **Methods:** ผู้ใช้สามารถทำการ `userLogin()`, `addProposal()` และในระดับผู้จัดการสามารถ `confirmProposal()` เพื่ออนุมัติงานได้

3.2 ข้อมูลข้อเสนอและผู้สมัคร (Proposal & Applicant)

- **Attributes:** ข้อเสนอจะระบุ `proposal_code` และสถานะ (`proposal_status`) ซึ่งค่าเริ่มต้นจะเป็น "Waiting" ส่วนผู้สมัครจะเก็บข้อมูลพื้นฐาน เช่น ชื่อ, อาชีพ และเบอร์โทรศัพท์
- **Methods:** ระบบสามารถ `viewStatus()` เพื่อดูความคืบหน้า และ `archiveProposal()` เพื่อเก็บเข้าคลังข้อมูล

4. ความสัมพันธ์ระหว่างคลาส (Class Relationships)

จากการวิเคราะห์สัญลักษณ์ใน Class Diagram สามารถสรุปความสัมพันธ์ได้ดังนี้:

- **Inheritance (การสืบทอด):**
  - ➢ **User สืบทอดจาก Login:** หมายถึง User ทุกคนต้องมีคุณสมบัติการเข้าใช้งานระบบ
- **Composition (ความสัมพันธ์แบบองค์ประกอบ - เพชรสีน้ำเงินทึบ):**
  - ➢ **Proposal และ Applicant:** มีความสัมพันธ์แบบ 1 ต่อ 1 หมายความว่าผู้สมัคร 1 คน จะผูกติดกับข้อเสนอ 1 รายการ หากข้อเสนอถูกลบข้อมูลผู้สมัครในส่วนนั้นจะหายไปด้วย
  - ➢ **Report และ Proposal:** รายงาน 1 ฉบับจะถูกสร้างขึ้นจากข้อมูลข้อเสนอ 1 รายการเสมอ
- **Association (ความสัมพันธ์ทั่วไป):**
  - ➢ **Department และ User:** ความสัมพันธ์แบบ 1 ต่อ 0..* หมายความว่า 1 แผนกสามารถมีพนักงานได้หลายคน แต่พนักงานบางคนอาจยังไม่สังกัดแผนกใดก็ได้
  - ➢ **User และ Proposal:** ผู้ใช้ 1 คนสามารถดูแลหรือสร้างข้อเสนอได้หลายรายการ

5. การใช้งาน Enumeration (ค่าคงที่ระบบ)

ระบบมีการใช้ Enum เพื่อควบคุมความถูกต้องของข้อมูล (Data Integrity):

- **RoleUser:** จำกัดบทบาทไว้ที่ PublicRelation, Division และ Manager
- **Gender:** กำหนดเพศเป็น M (ชาย) และ F (หญิง)
- **StatusProposal:** ควบคุมสถานะข้อเสนอให้อยู่ในกลุ่ม Waiting, Rejected และ Accepted เท่านั้น

6. สรุปผลการวิเคราะห์

ระบบนี้มีโครงสร้างที่ยืดหยุ่นด้วยการใช้ **Inheritance** ในการจัดการความปลอดภัย และใช้ **Composition** เพื่อให้มั่นใจว่าข้อมูลที่มีความเกี่ยวข้องกัน เช่น ข้อเสนอและผู้สมัคร จะถูกจัดการไปพร้อมกันอย่างถูกต้องตามกฎของธุรกิจ **(Business Logic)**

```
Somchai Logged in successfully.
Proposal added.
Viewing Report ID: 901
Status: Waiting
```

Department.java

Department.java > Language Support for Java(TM) by Red Hat > Department

```java
class Department {
    private int department_id;
    private String department_name;

    public Department(int id, String name) {
        this.department_id = id;
        this.department_name = name;
    }

    public void insertDepartment() { System.out.println(x: "Inserted Department"); }
    public void updateDepartment() { System.out.println(x: "Updated Department"); }
    public void deleteDepartment() { System.out.println(x: "Deleted Department"); }
}
```

Login.java

Login.java > Java > Login

```java
class Login {
    protected String username;
    protected String password;

    public Login(String username, String password) {
        this.username = username;
        this.password = password;
    }

    // Methods
    public boolean loginStatus() {
        return true; // สมมติว่าล็อกอินผ่าน
    }
}
```

User.java

User.java > Java > User > viewProposal()

```java
import java.util.ArrayList;
import java.util.List;

public class User extends Login {
    private int user_id;
    private int department_id;
    private String user_username;
    private String user_password;
    private String user_name;
    private Gender user_gender;      // ต้องมีไฟล์ Gender.java
    private String user_email;
    private RoleUser user_role;      // ต้องมีไฟล์ RoleUser.java

    private Department department;
    private List<Proposal> myProposals = new ArrayList<>();

    // Constructor
    public User(String username, String password, int uid, String name, RoleUser role) {
        super(username, password);
        this.user_id = uid;
        this.user_name = name;
        this.user_role = role;
    }

    public void setDepartment(Department dept) {
        this.department = dept;
        this.department_id = 1;
    }

    // --- เมธอดที่ Error แจ้งว่าหาไม่เจอ ---
    public void userLogin() {
        if(super.loginStatus()) {
            System.out.println(user_name + " Logged in successfully.");
        }
    }

    public void addProposal(Proposal p) {
        myProposals.add(p);
        System.out.println(x: "Proposal added.");
    }
    // ------------------------------------

    public void viewProposal() { System.out.println(x: "Viewing all proposals..."); }
    public void confirmProposal() { System.out.println(x: "Proposal Confirmed."); }
    public void createReports() { System.out.println(x: "Report Created by User."); }
    public void viewReports() { System.out.println(x: "Viewing Reports."); }

    public void updateDepartment() {
        if(department != null) department.updateDepartment();
    }
}
```

Proposal.java

Proposal.java > Language Support for Java(TM) by Red Hat > Proposal

```java
import java.util.Date;

class Proposal {
    private String proposal_code;
    private String proposal_title;
    private String proposal_text;
    private Date proposal_date;
    private StatusProposal proposal_status;

    //  Composition: Proposal 1 รายการ ต้องมี Applicant 1 คน
    private Applicant applicant;

    public Proposal(String code, String title, Applicant app) {
        this.proposal_code = code;
        this.proposal_title = title;
        this.proposal_date = new Date();
        this.proposal_status = StatusProposal.Waiting; //ค่าเริ่มต้น Waiting
        this.applicant = app; // ผูก Applicant เข้ากับ Proposal ทันที
    }

    //  Methods
    public void viewProposal() { System.out.println("Viewing Proposal: " + proposal_title); }
    public void viewStatus() { System.out.println("Status: " + proposal_status); }
    public void updateProposal() { System.out.println(x: "Proposal Updated"); }
    public void deleteProposal() { System.out.println(x: "Proposal Deleted"); }
    public void archiveProposal() { System.out.println(x: "Proposal Archived"); }
    public void viewApplicant() { System.out.println("Applicant is: " + applicant); }
}
```

Report.java

Report.java > Language Support for Java(TM) by Red Hat > Report > viewReports()

```java
import java.util.Date;

class Report {
    private int Report_code;
    private Date Report_date;

    // Composition: Report 1 ฉบับ ผูกกับ Proposal 1 รายการ
    private Proposal proposal;

    public Report(int code, Proposal prop) {
        this.Report_code = code;
        this.Report_date = new Date();
        this.proposal = prop;
    }

    // Methods
    public void viewReports() { System.out.println("Viewing Report ID: " + Report_code); }
    public void createReports() { System.out.println(x: "Report Created"); }
}
```

Applicant.java

Applicant.java > Language Support for Java(TM) by Red Hat > Applicant > insertApplicant()

```java
class Applicant {
    private int Applicant_id;
    private String Applicant_name;
    private String Applicant_job;
    private String Applicant_address;
    private Gender Applicant_gender;
    private String Applicant_tel;

    public Applicant(int id, String name, Gender gender) {
        this.Applicant_id = id;
        this.Applicant_name = name;
        this.Applicant_gender = gender;
    }

    //  Methods
    public void insertApplicant() { System.out.println(x: "Applicant Inserted"); }
    public void updateApplicant() { System.out.println(x: "Applicant Updated"); }
    public void deleteApplicant() { System.out.println(x: "Applicant Deleted"); }
}
```

RoleUser.java

userLogin.java > Language Support for Java(TM) by Red Hat > userLogin > updateDepartment()

```java
public void userLogin() {
        if(super.loginStatus()) {
            System.out.println(user_name + " Logged in successfully.");
        }
    }

    public void addProposal(Proposal p) {
        myProposals.add(p);
        System.out.println(x: "Proposal added.");
    }

    public void viewProposal() { System.out.println(x: "Viewing all proposals..."); }
    public void confirmProposal() { System.out.println(x: "Proposal Confirmed."); }
    public void createReports() { System.out.println(x: "Report Created by User."); }
    public void viewReports() { System.out.println(x: "Viewing Reports."); }

    // Method นี้ชื่อซ้ำกับใน Department แต่น่าจะเป็นการเรียกใช้ข้ามคลาส
    public void updateDepartment() {
        if(department != null) department.updateDepartment();
    }
```

Gender.java

```java
// Source code is decompiled from a .class
enum Gender {
    M,
    F;

    private Gender() {
    }
}

```

StatusProposal.java

StatusProposal.java > Language Support for Java(TM) by

```java
public enum StatusProposal {
    Waiting,
    Rejected,
    Accepted
}
```