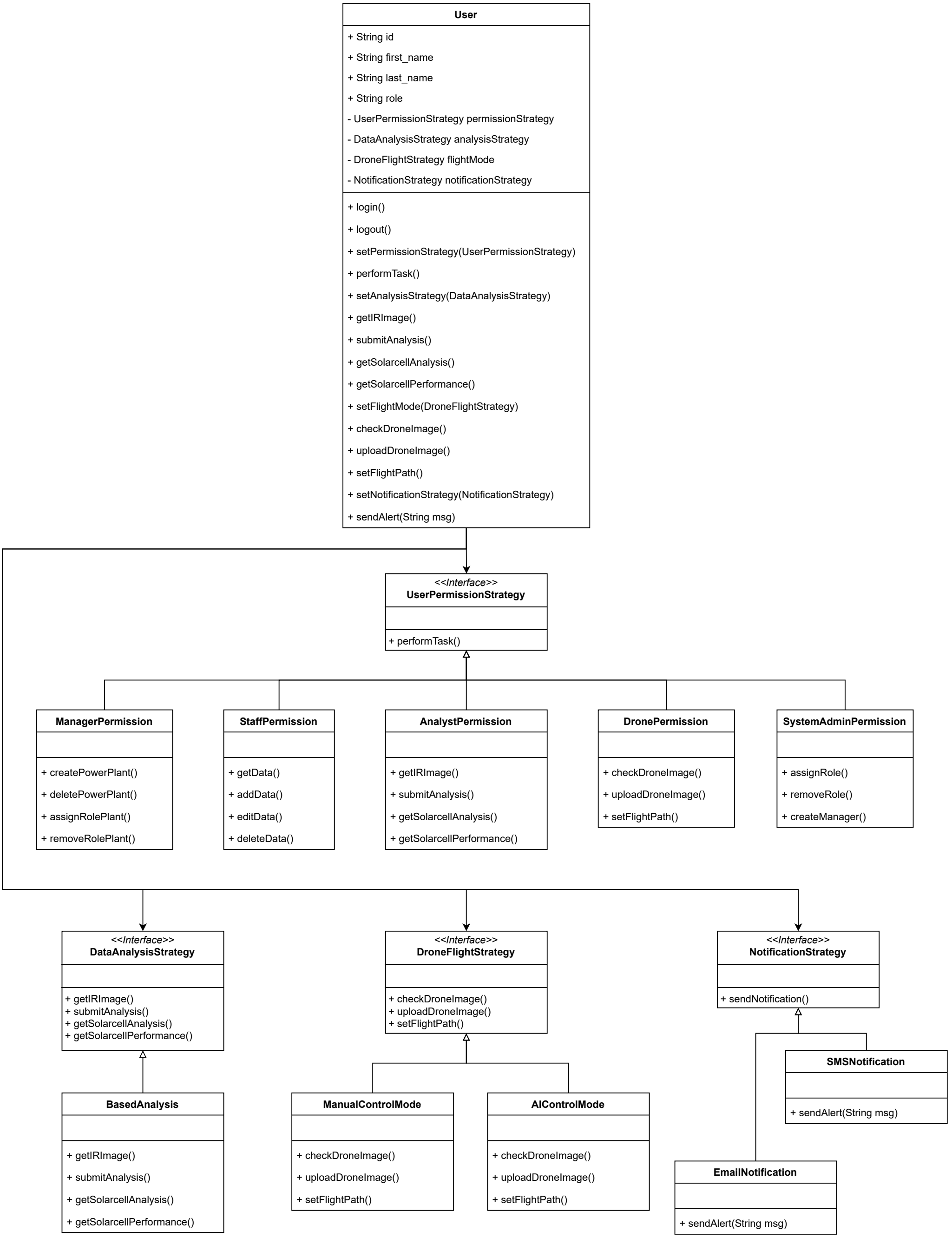
STATEGY PATTERN



Strategy Pattern

Strategy Pattern เป็น Behavioral Design Pattern ที่ช่วยให้สามารถเปลี่ยนพฤติกรรมของออบเจ็กต์ได้แบบไดนามิก โดยใช้ Composition แทนการใช้ Inheritance

- แทนที่ class หลักจะต้องกำหนดพฤติกรรม (method) เองทั้งหมด มันจะส่งต่อการทำงานไปยัง Strategy Object
- สามารถเปลี่ยนพฤติกรรมใน runtime ได้โดยการเปลี่ยน strategy ที่ใช้

อธิบายโครงสร้างของ UML Diagram

User (Context)

User เป็น ตัวหลักของระบบ ที่จะเลือกใช้กลยุทธ์ (strategy) ต่าง ๆ โดยอิงจากบทบาทและสถานการณ์ที่เกิดขึ้น โดย User มี 4 กลยุทธ์หลัก (Strategy Pattern) ดังนี้

- UserPermissionStrategy ⟶ กำหนดสิทธิ์ของผู้ใช้
- DataAnalysisStrategy ⟶ กำหนดวิธีวิเคราะห์ข้อมูล
- DroneFlightStrategy ⟶ กำหนดโหมดการบินของโดรน
- NotificationStrategy ⟶ กำหนดรูปแบบการแจ้งเตือน

User สามารถเปลี่ยน strategy ได้โดยใช้ setter method เช่น

- setPermissionStrategy(UserPermissionStrategy strategy)
- setAnalysisStrategy(DataAnalysisStrategy strategy)
- setFlightMode(DroneFlightStrategy strategy)
- setNotificationStrategy(NotificationStrategy strategy)

UserPermissionStrategy (Strategy)

เป็น interface ที่กำหนดพฤติกรรมเกี่ยวกับสิทธิ์การเข้าถึงระบบ (performTask()) โดย Class ลูกแต่ละตัว (Concrete Strategy) จะกำหนดสิทธิ์ที่แตกต่างกัน เช่น

- ManagerPermission ⟶ สร้าง, ลบ, และจัดการโรงไฟฟ้า
- StaffPermission ⟶ ดึง, เพิ่ม, แก้ไข, ลบข้อมูล
- AnalystPermission ⟶ วิเคราะห์ข้อมูลพลังงานแสงอาทิตย์
- DronePermission ⟶ ควบคุมโดรน
- SystemAdminPermission ⟶ จัดการบัญชีผู้ใช้

ประโยชน์ของ Strategy Pattern ในส่วนนี้

- ทำให้สามารถเพิ่มสิทธิ์ใหม่ โดยไม่ต้องแก้ไขโค้ดหลัก (User)
- ลดการใช้ if-else/switch-case ในการกำหนดสิทธิ์

DataAnalysisStrategy (Strategy)

เป็น interface ที่กำหนดวิธีการวิเคราะห์ข้อมูล (getIRImage(), submitAnalysis(), getSolarcellAnalysis(), getSolarcellPerformance()) โดย Class ลูก (Concrete Strategy) จะกำหนดวิธีวิเคราะห์ที่แตกต่างกัน

ประโยชน์ของ Strategy Pattern ในส่วนนี้

- สามารถเปลี่ยนวิธีการวิเคราะห์แบบไดนามิกโดยไม่ต้องแก้ไขโค้ดของ User
- ทำให้รองรับเทคนิคใหม่ๆ ได้ง่ายขึ้น เช่น เปลี่ยนจาก Rule-based เป็น ML

DroneFlightStrategy (Strategy)

เป็น interface ที่กำหนดโหมดการบินของโดรน (checkDroneImage(), uploadDroneImage(), setFlightPath()) โดย Class ลูก (Concrete Strategy) จะกำหนดพฤติกรรมที่แตกต่างกัน ดังนี้

- ManualControlMode ⟶ ผู้ใช้ควบคุมเอง
- AIControlMode ⟶ ใช้ AI ควบคุม

ประโยชน์ของ Strategy Pattern ในส่วนนี้

- สามารถสลับโหมดการบินของโดรนแบบ runtime ได้
- เพิ่มโหมดใหม่ๆ ได้ง่ายโดยไม่ต้องแก้ไขโค้ดของ User

NotificationStrategy (Strategy)

เป็น interface ที่กำหนดวิธีการแจ้งเตือน (sendAlert(String msg)) โดย Class ลูก (Concrete Strategy) กำหนดรูปแบบแจ้งเตือนที่แตกต่างกัน เช่น

- EmailNotification ⟶ แจ้งเตือนผ่าน Email
- SMSNotification ⟶ แจ้งเตือนผ่าน SMS

ประโยชน์ของ Strategy Pattern ในส่วนนี้

- เปลี่ยนช่องทางแจ้งเตือนได้ง่ายโดยไม่ต้องแก้โค้ดของ User
- สามารถเพิ่มรูปแบบการแจ้งเตือนใหม่ๆ ได้โดยไม่ต้องแก้ไขโครงสร้างหลัก