BASR (بصر)
Broad Assistance Search and Rescue

LAB 6

This project was submitted as part of Lab 6 in the Web Development Bootcamp using the Java programming language, organized by Tuwaiq Academy under the Saudi Federation for Cybersecurity, Programming, and Drones

By Student

Hassan Yousef Al-Husseini

on July 22, 2025

Objective

This project was developed as part of learning **Spring Boot Validation** techniques. The primary objective is to analyze backend requirements and apply precise validation rules to ensure that each field in the system complies with specific constraints, enhancing data accuracy and reliability.

The project is a backend simulation of the system behind **"Basar"**, an AI-powered drone designed to assist in search-and-rescue operations — especially for individuals lost in desert environments.

By enforcing strong validation at the backend level, the system ensures that only clean, structured, and rule-compliant data flows into the AI components, which contributes to more accurate detection and faster rescue responses.

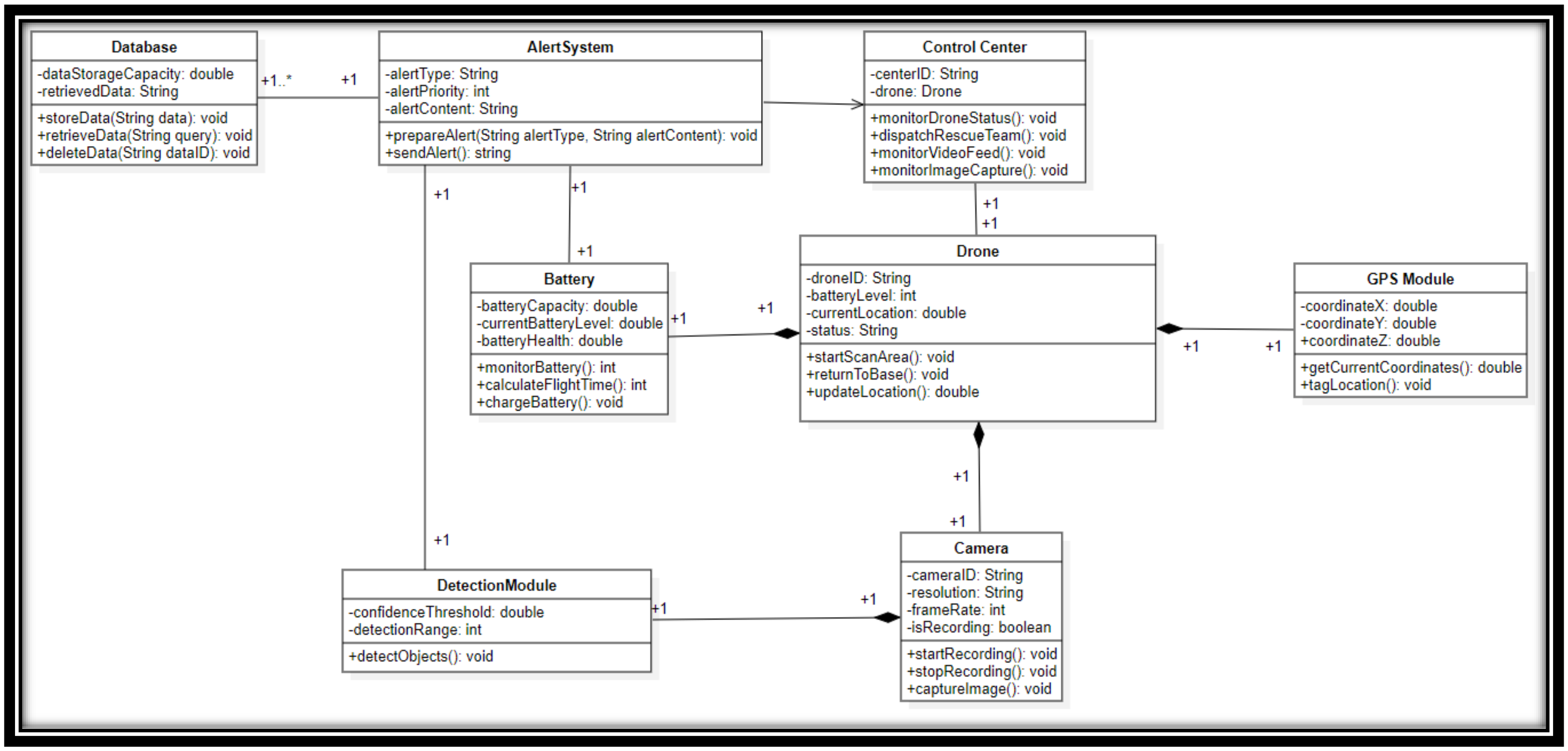
Project Overview

In our vast deserts and remote areas, many lives are lost despite the Ministry of Interior's efforts.

This tragic reality has driven us to develop (BASR) an AI-powered drone system to assist in locating and assessing the condition of lost individuals.

The system will send their location to rescue teams, aiming to reduce fatalities and support the ministry's rescue operations.

Class Diagram

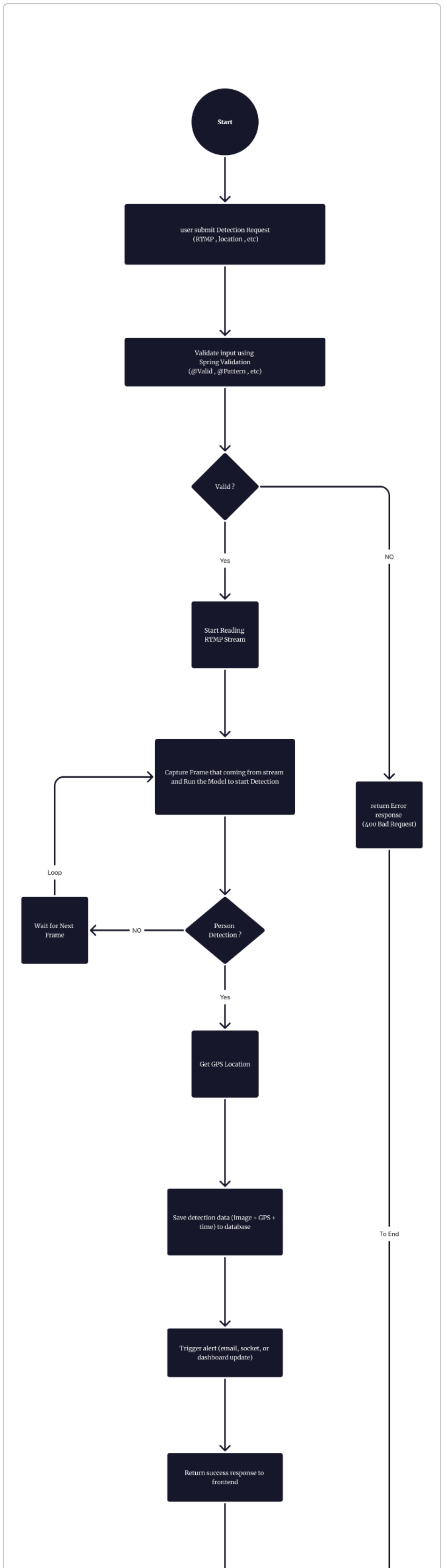


Class Diagram Description

Class	**Description**
Drone	The Drone class encapsulates all the information and functionalities of a drone. It includes details such as battery level, current location, and operational status. It interacts with the Camera, Battery, GPS Module classes to perform tasks like scanning, capturing data, and updating its position.
Camera	The Camera class captures video and photos during drone operations. It has attributes that define its resolution, frame rate, and recording status. The

	Camera can start or stop recording and capture still images, allowing it to document drone activities and provide visual data for rescue missions.
Battery	The Battery class manages the battery status of the drone. It monitors battery capacity, health, and level, and provides methods to calculate flight time and charge the battery. It has a one-to-one relationship with the Drone class to ensure real-time battery status updates.
DetectionModule	The DetectionModule class is responsible for detecting objects in the drone's operational range. It uses a confidence threshold to filter out detections based on reliability and has methods to initiate object detection. This class plays a crucial role in scanning areas and identifying relevant objects during rescue missions.
GPSModule	The GPS Module class provides real-time location information for the drone. It includes methods to get the drone's current coordinates and tag specific locations. This class has a one-to-one relationship with the Drone class, ensuring accurate positioning.
AlertSystem	The AlertSystem class handles the generation and dispatching of alerts. It prepares alerts based on the type, priority, and content, and sends them to the respective stakeholders. This class has an association with the Control Center class, where alerts are monitored and acted upon.
Control Center	The Control Center class manages drone operations and coordinates rescue missions. It monitors the status of Drone and Image/Video capture also has methods to dispatch a rescue team when necessary.
Database	The Database class stores and manages all data related to drone operations, alerts, and monitoring. It provides methods for storing, retrieving, and deleting data. Data is first stored on an onboard storage device and later transferred to the server at the rescue base, ensuring data backup and accessibility for rescue teams. It maintains a one-to-many relationship with the Alert System class, allowing multiple alert events to be logged and accessed.

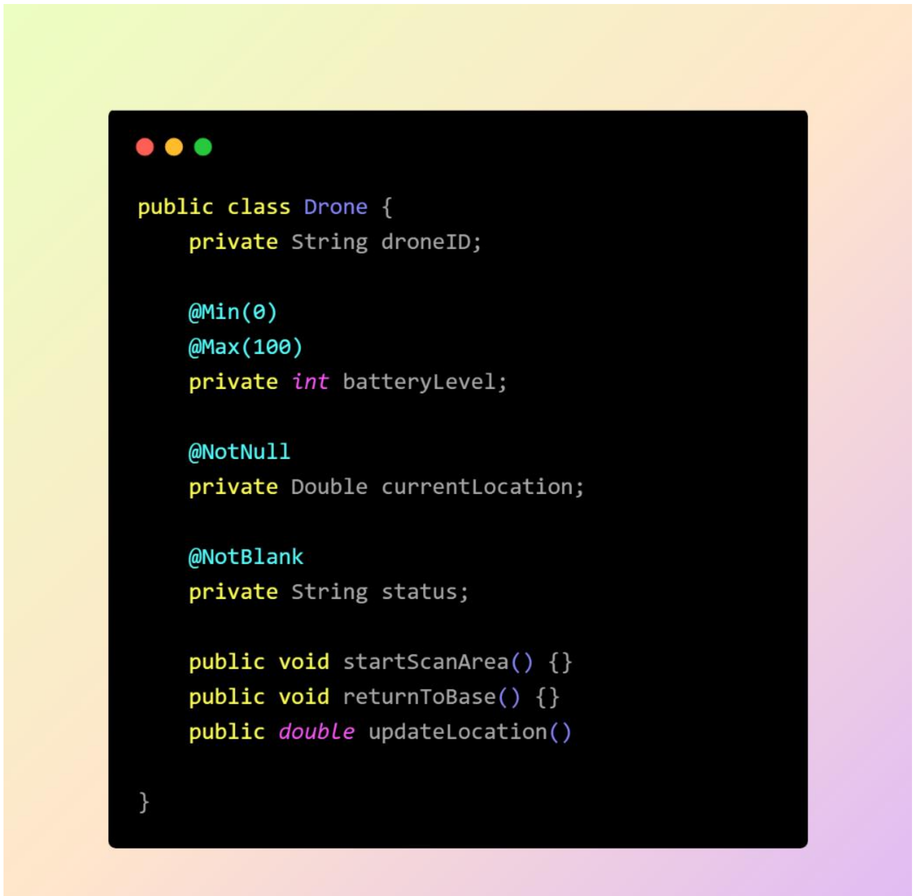
BASR Flow Chart Diagram


Start
user submit Detection Request (RTMP , location , etc)
Validate input using Spring Validation (@Valid , @Pattern , etc)
Valid ?
Yes
NO
Start Reading RTMP Stream
Capture Frame that coming from stream and Run the Model to start Detection
return Error response (400 Bad Request)
Loop
Wait for Next Frame
NO
Person Detection ?
Yes
Get GPS Location
Save detection data (image + GPS + time) to database
To End
Trigger alert (email, socket, or dashboard update)
Return success response to frontend

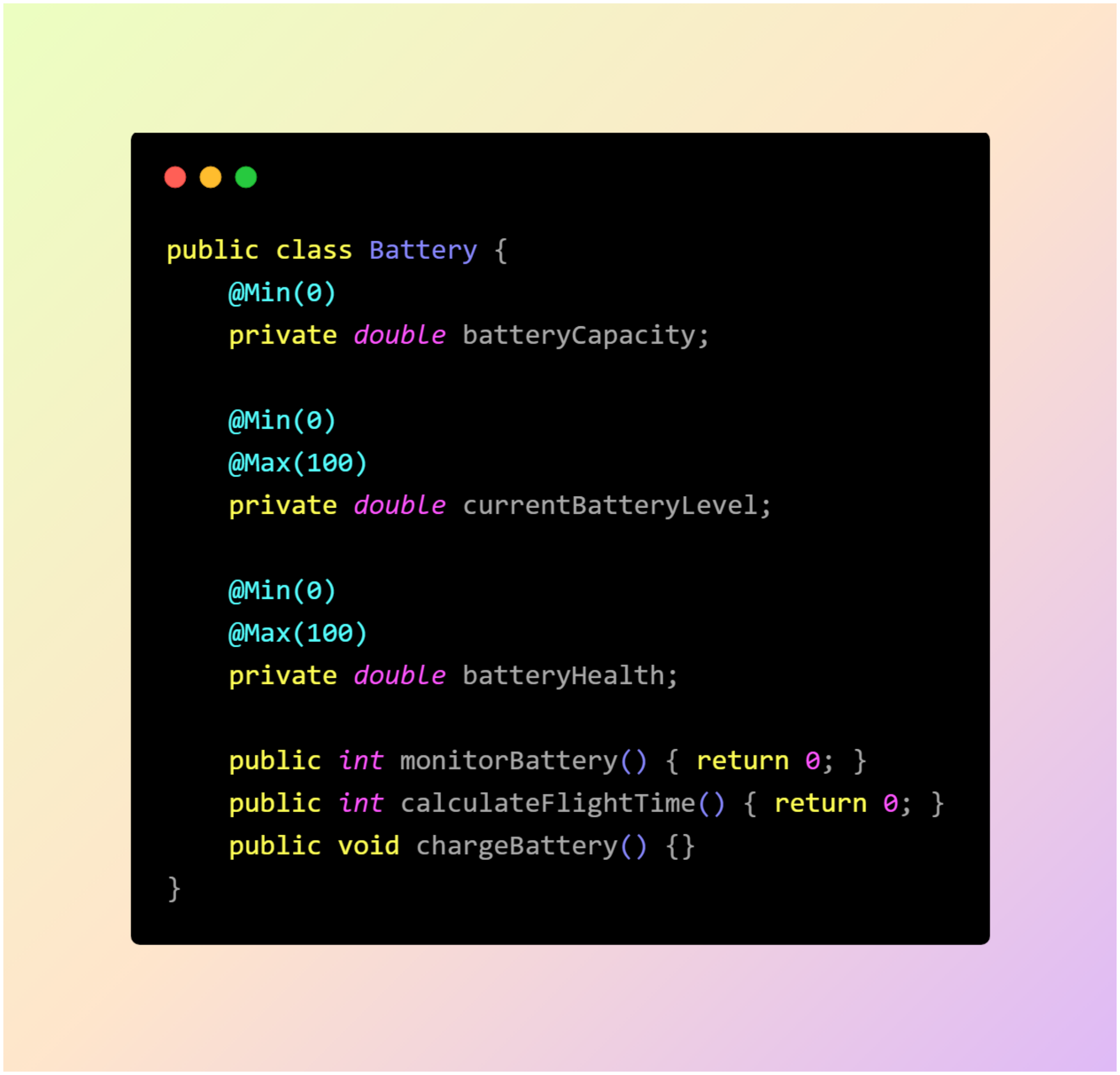
Predicted End Point

Operation	HTTP Method	Endpoint	Notes
Start detection	POST	/api/detect/start	Includes RTMP URL and GPS data from user
Receive detection frame	POST	/api/detect/frame	Internal – triggered by (YOLO) Model
Get detection list	GET	/api/detect/list	Returns list of all detections
Get detection details	GET	/api/detect/{id}	Returns data of a specific detection
Send alert	POST	/api/alert/send	Internal – triggered when detection occurs

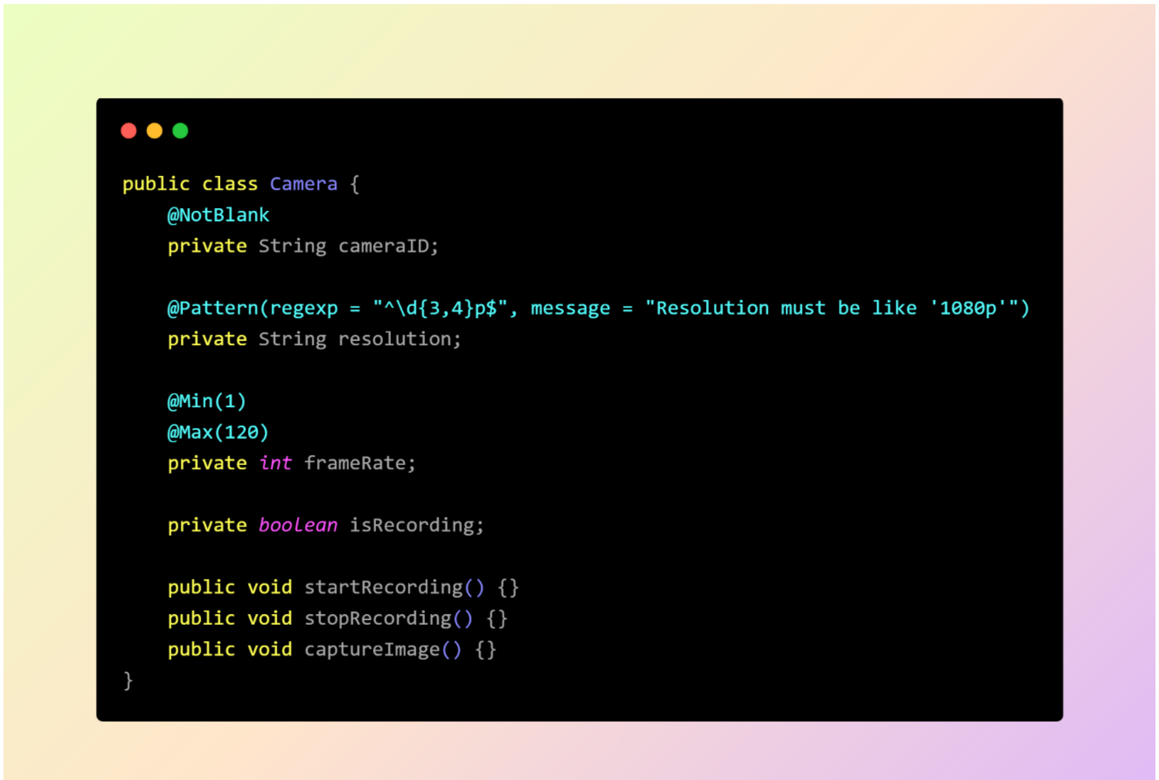
Validation Classes



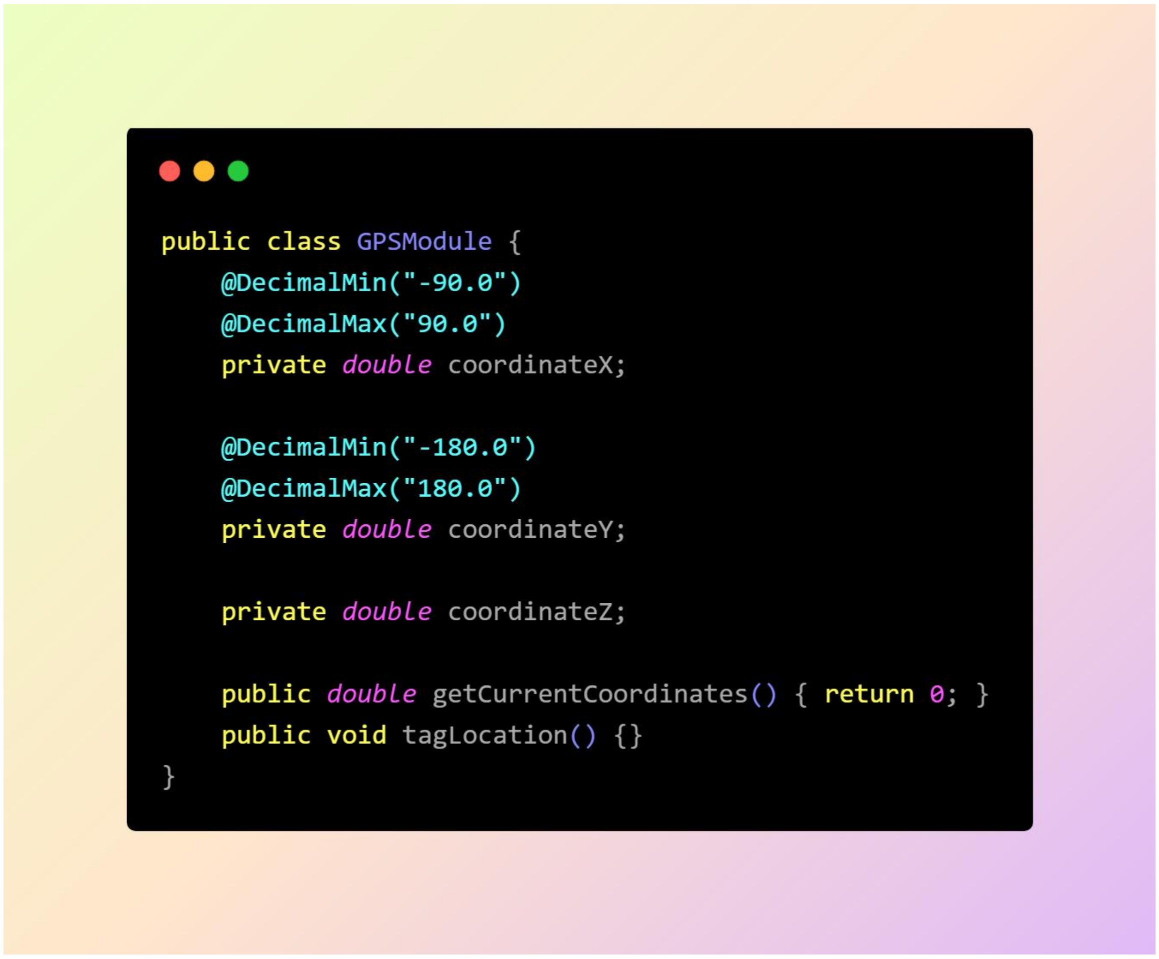
```
public class Drone {
    private String droneID;

    @Min(0)
    @Max(100)
    private int batteryLevel;

    @NotNull
    private Double currentLocation;

    @NotBlank
    private String status;

    public void startScanArea() {}
    public void returnToBase() {}
    public double updateLocation()

}
```



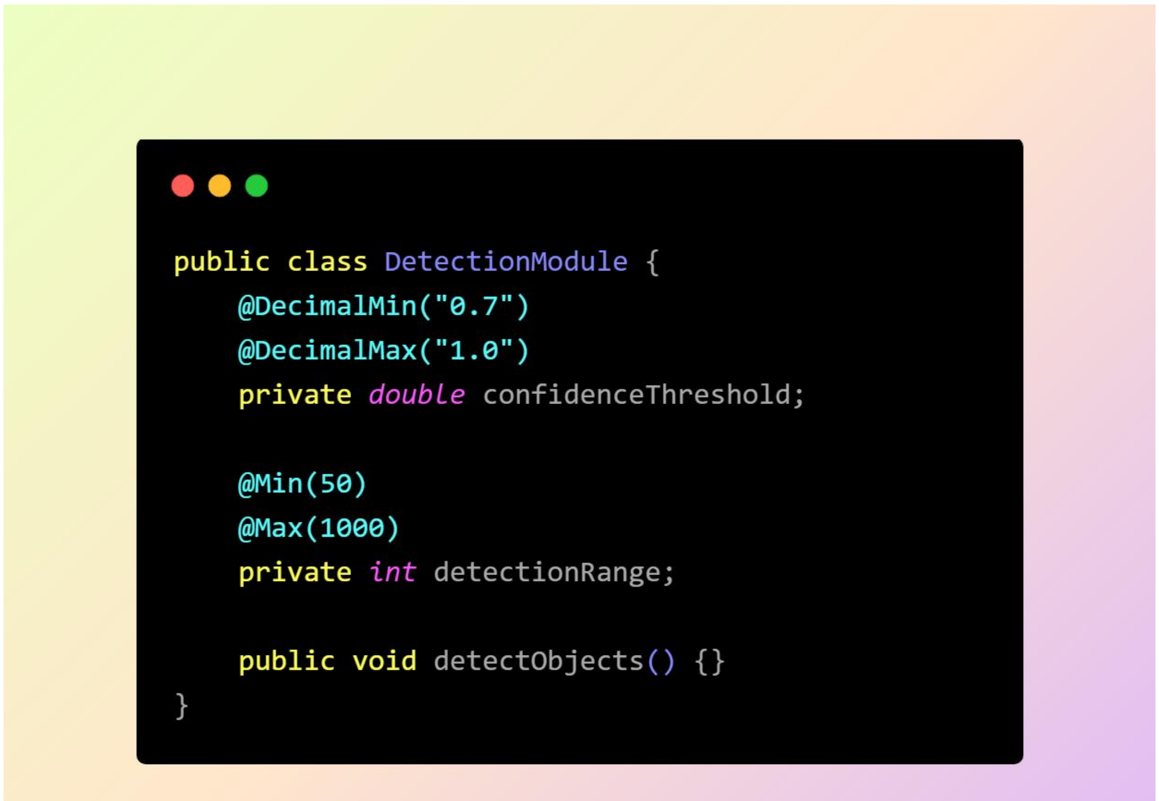
```java
public class Battery {
    @Min(0)
    private double batteryCapacity;

    @Min(0)
    @Max(100)
    private double currentBatteryLevel;

    @Min(0)
    @Max(100)
    private double batteryHealth;

    public int monitorBattery() { return 0; }
    public int calculateFlightTime() { return 0; }
    public void chargeBattery() {}
}
```



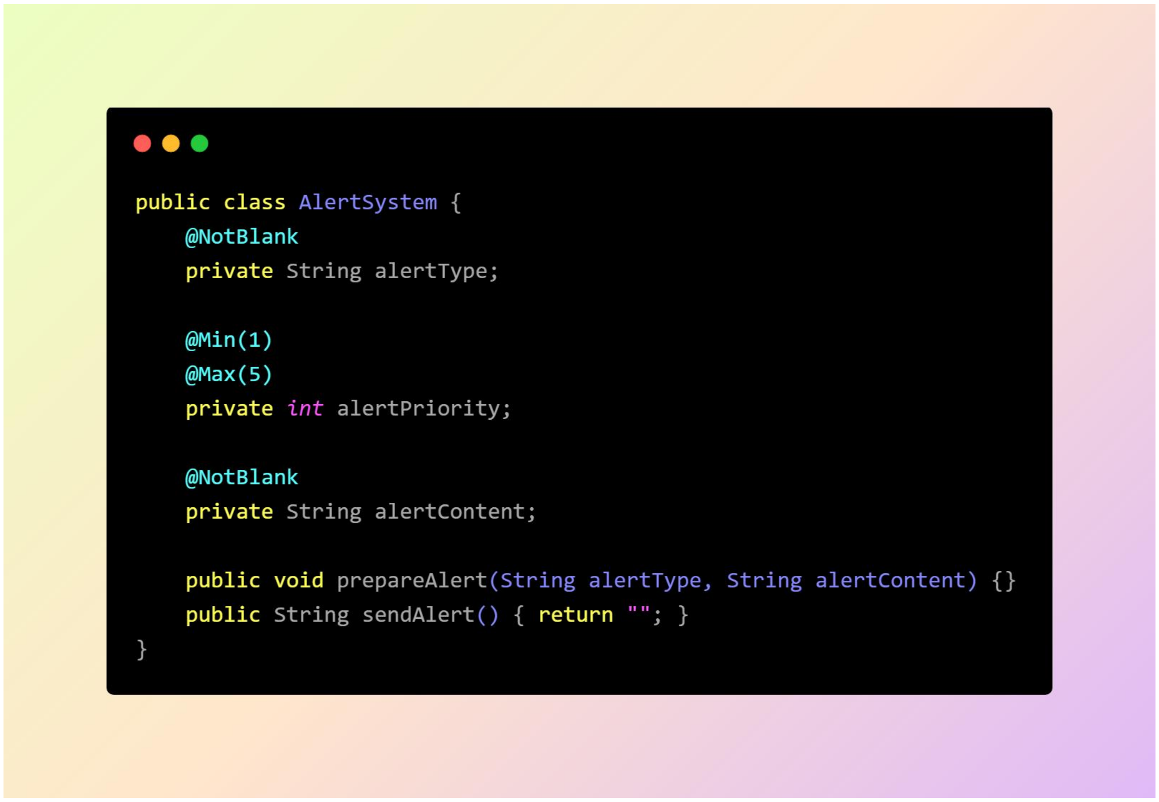
```java
public class Camera {
    @NotBlank
    private String cameraID;

    @Pattern(regexp = "^\d{3,4}p$", message = "Resolution must be like '1080p'")
    private String resolution;

    @Min(1)
    @Max(120)
    private int frameRate;

    private boolean isRecording;

    public void startRecording() {}
    public void stopRecording() {}
    public void captureImage() {}
}
```



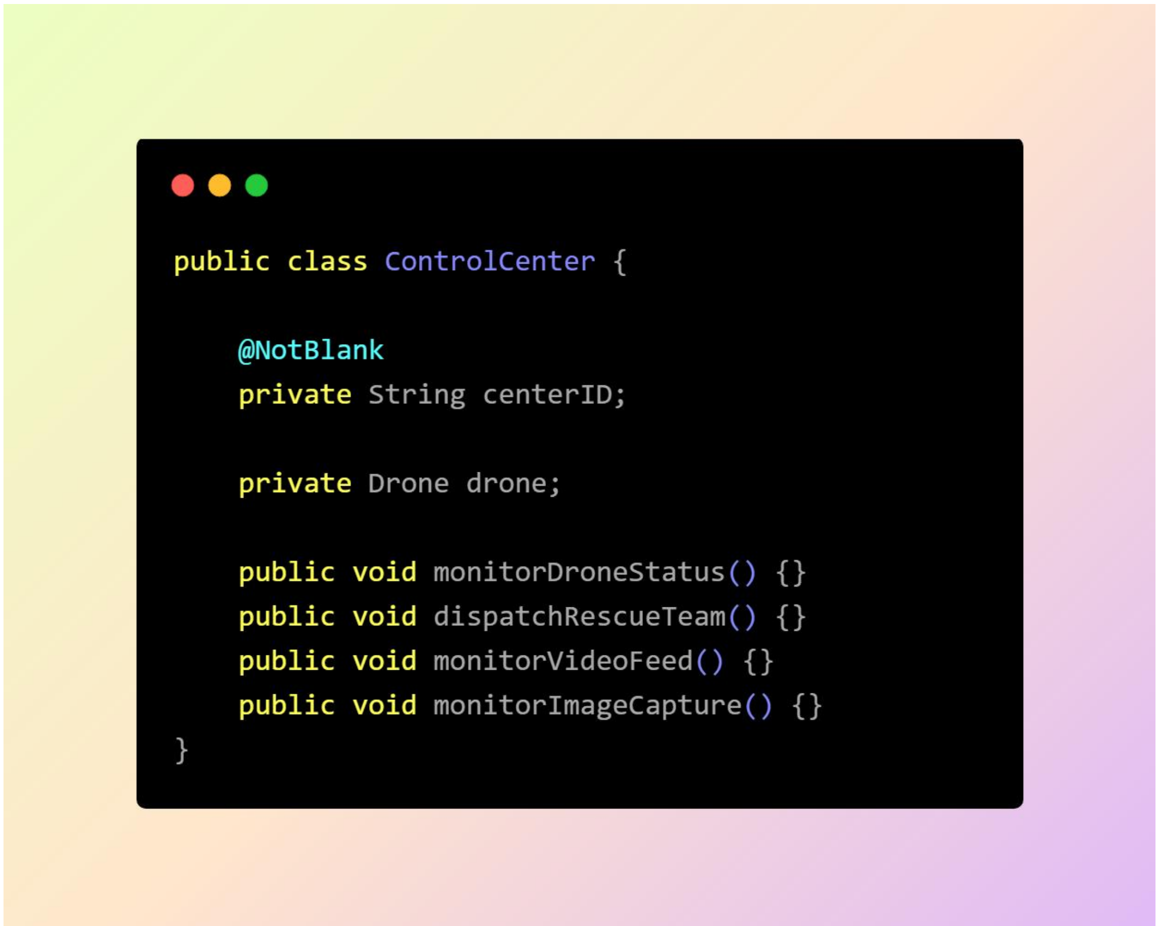
```java
public class GPSModule {
    @DecimalMin("-90.0")
    @DecimalMax("90.0")
    private double coordinateX;

    @DecimalMin("-180.0")
    @DecimalMax("180.0")
    private double coordinateY;

    private double coordinateZ;

    public double getCurrentCoordinates() { return 0; }
    public void tagLocation() {}
}
```



```java
public class DetectionModule {
    @DecimalMin("0.7")
    @DecimalMax("1.0")
    private double confidenceThreshold;

    @Min(50)
    @Max(1000)
    private int detectionRange;

    public void detectObjects() {}
}
```



```java
public class AlertSystem {
    @NotBlank
    private String alertType;

    @Min(1)
    @Max(5)
    private int alertPriority;

    @NotBlank
    private String alertContent;

    public void prepareAlert(String alertType, String alertContent) {}
    public String sendAlert() { return ""; }
}
```



```java
public class ControlCenter {

    @NotBlank
    private String centerID;

    private Drone drone;

    public void monitorDroneStatus() {}
    public void dispatchRescueTeam() {}
    public void monitorVideoFeed() {}
    public void monitorImageCapture() {}
}
```