Emergency Response Simulation - Short Report

Project Overview

This C# console application simulates a basic emergency response system where specialized units respond to different types of incidents. It demonstrates core Object-Oriented Programming (OOP) concepts, including abstraction, inheritance, and polymorphism.

Code Structure

Classes

- **EmergencyUnit (Abstract Class)**
  - Properties:
    - `Name` (string)
    - `Speed` (int)
  - Abstract Methods:
    - `CanHandle(string incidentType) : bool`
    - `RespondToIncident(Incident incident) : void`
- **Police, Firefighter, Ambulance (Derived Classes)**
  - Override `CanHandle` to check for specific incident types ("Crime", "Fire", "Medical").
  - Override `RespondToIncident` to provide appropriate console output.
- **Incident**
  - Properties:
    - `Type` (string)
    - `Location` (string)
  - Represents an emergency scenario occurring at a location.
- **Program**
  - `Main()` method contains the simulation loop.
  - Manages unit creation, random incident generation, response handling, and score tracking.

Game Logic

1. A list of possible incident types and locations is defined.
2. Three units are instantiated: Police Unit 1, Firefighter Unit 1, and Ambulance Unit 1.
3. For 5 rounds:
   - A random incident is generated.
   - The program checks if any available unit can handle the incident.
   - If handled, +10 points are awarded.
   - If not handled, -5 points are deducted.

    - Current score is displayed after each round.
4. The final score is displayed after the 5th round.

Example Output

```
--- Turn 1 ---
Incident: Fire at Mall
Firefighter Unit 1 is extinguishing a fire at Mall.
+10 points
Current Score: 10

--- Simulation Complete ---
Final Score: 35
```

Concepts Demonstrated

- **Abstraction**: `EmergencyUnit` defines common interface for all units.
- **Inheritance**: `Police`, `Firefighter`, and `Ambulance` inherit from `EmergencyUnit`.
- **Polymorphism**: The program uses `CanHandle` and `RespondToIncident` methods polymorphically across different units.
- **Encapsulation**: Unit and incident properties are managed securely via public getters and setters.

Challenges Faced

- **Designing the Abstraction Properly**: Ensuring that the abstract class `EmergencyUnit` was generic enough to be extended by multiple types of emergency services without redundancy.
- **Handling Unexpected Incidents**: Since the incident types could include types not handled by any unit (e.g., "Flood", "Earthquake"), it was challenging to ensure proper scoring (-5 points) and graceful failure handling.
- **Managing Randomness**: Creating a fair and random selection of incidents while keeping the game both unpredictable and balanced was tricky.
- **Code Readability and Maintainability**: Keeping the code clean, organized, and understandable while still being functional required extra attention, especially as more features could easily increase complexity.

Future Improvements

- Add new unit types (e.g., Flood Rescue Team, Earthquake Rescue Team).
- Handle multiple incidents simultaneously.
- Implement unit availability and cooldown periods.
- Add different severity levels for incidents.
- Enhance console output with color coding and animations.