Short Report on the Emergency Response Simulation

This C# code simulates the dispatch of emergency units (Police, Firefighter, Ambulance) to various incidents (Fire, Crime, Medical) occurring at different locations. The simulation proceeds in rounds, where random incidents are generated, and the system attempts to find a suitable emergency unit to respond. Points are awarded for successful responses and deducted when no appropriate unit is available.

Key Components:

- `Emergency Unit` **(Abstract Class): This serves as a blueprint for all emergency units. It defines common properties like** `Name` **and** `Speed`**, and abstract methods** `Can Handle(string incident Type)` **(to check if a unit can respond to a specific incident type) and** `Respond To Incident(Incident incident)` **(to simulate the unit's response). The** `To String()` **method is overridden for easy representation of unit** information.
- **Concrete Emergency Unit Classes (**`Police`**,** `Firefighter`**,** `Ambulance`**): These classes inherit from** `Emergency Unit` **and provide specific implementations for the abstract methods. Each unit type is designed to handle a particular type of incident.**
- `Incident` **Class: This class represents an emergency event with a** `Type` **(e.g., "Fire", "Crime", "Medical") and a** `Location`**.**
- `Simulation` **Class: This class contains the logic for running the simulation:**
  - `Generate Random Incident()`**: Creates a random incident based on predefined locations and incident types (which can be extended by the user).**
  - `Find Available Unit()`**: Iterates through a list of emergency units to find the first one that can handle the given incident type.**
  - `Run Simulation()`**: This is the main simulation loop. It prompts the user for the number of simulation rounds and initial unit information. It then generates incidents, attempts to dispatch units, updates the score, and finally displays the simulation results, including the locations, incident types, emergency units, and the final score.**

    - **`Get Unit Speed From User()`: A helper method to safely get a positive integer for the speed of each unit from the user.**
- **`Program` Class: This simple class contains the `Main` method, which is the entry point of the application and simply calls the `Run Simulation()` method.**

Key Features Demonstrated:

- **Abstraction: The `Emergency Unit` abstract class defines a common interface for all concrete unit types.**
- **Inheritance: `Police`, `Firefighter`, and `Ambulance` inherit from `Emergency Unit`, reusing common properties and methods while implementing their specific behaviors.**
- **Polymorphism: The `Can Handle()` and `Respond To Incident()` methods are declared in the base class but have different implementations in the derived classes, allowing different unit types to handle incidents in their own way.**
- **Encapsulation: Properties like `Name` and `Speed` in `Emergency Unit` are protected, controlling their access.**
- **User Input: The simulation takes user input for the number of rounds, initial unit information, and the ability to add new locations and incident types.**
- **Simulation Logic: The code simulates a basic dispatch system with random incident generation and a scoring mechanism.**

Potential Improvements:

- **More Sophisticated Dispatch Logic: The `Find Available Unit()` method simply picks the first available unit. A more complex system could consider unit locations, distances to incidents, or unit availability status.**
- **Event Handling: Instead of direct method calls, an event-driven system could be implemented where incidents are events that units can subscribe to.**
- **Data Persistence: The simulation data (units, scores, etc.) could be saved and loaded.**
- **User Interface: A graphical user interface (GUI) could make the simulation more interactive and visually appealing.**

Class Diagram (Text-Based Structure)