Remote Health Monitoring System (RHMS)

Project Overview

This project implements a Remote Health Monitoring System for Near East University Hospital. The system tracks patients with chronic conditions (hypertension, diabetes) using wearable devices that automatically transmit vital signs to the hospital system.

System Features

- **Patient Management**: Register and track patients with chronic conditions
- **Wearable Device Integration**: Simulate IoT devices recording vital signs
- **Real-time Monitoring**: Automatic data transmission and analysis
- **Alert Generation**: Intelligent detection of abnormal vital signs
- **Doctor Notification**: Automatic alert forwarding to appropriate specialists
- **Medical Recommendations**: Doctors review alerts and provide guidance
- **Follow-up Scheduling**: Coordinate patient appointments based on severity

Project Structure

```
RemoteHealthMonitoringSystem/
  Patient.java              # Patient entity class
  VitalSigns.java          # Health metrics data structure
  WearableDevice.java      # IoT device simulation
  Alert.java               # Alert entity with severity levels
  Doctor.java              # Doctor entity with review capabilities
  MonitoringSystem.java    # Core system coordination class
  Main.java                # Application entry point & demo
  README.md                # This file
```

Class Relationships

Inheritance

Currently, the system uses composition over inheritance for flexibility. Future versions could implement: - `Person` superclass for `Patient` and `Doctor` - `MedicalDevice` superclass for different device types

Associations

- **MonitoringSystem** manages multiple **Patients** and **Doctors** (1-to-many)
- **Patient** uses **WearableDevice** (1-to-1)
- **WearableDevice** generates **VitalSigns** (1-to-many)
- **VitalSigns** triggers **Alert** if abnormal (1-to-0..1)
- **Alert** is reviewed by **Doctor** (many-to-1)
- **Doctor** provides recommendations to **Patient** (many-to-many)

Composition

- **MonitoringSystem** contains collections of all entities
- **Alert** contains **VitalSigns** data
- **VitalSigns** includes timestamp and patient reference

How to Run

Prerequisites

- Java Development Kit (JDK) 8 or higher
- Any Java IDE (Eclipse, IntelliJ IDEA, VS Code) or command line

Compilation

```
javac *.java
```

Execution

```
java Main
```

Sample Output

The program simulates three monitoring cycles: 1. Patient with hypertension showing elevated blood pressure 2. Patient with diabetes showing abnormal readings 3. Patient with hypertension showing critical values

Each cycle demonstrates: - Vital signs recording - Data transmission - Alert generation (if needed) - Doctor review and recommendations - Follow-up scheduling

Normal Vital Sign Ranges

- **Heart Rate**: 60-100 bpm
- **Blood Pressure**: <120/80 mmHg
- **Temperature**: 36.1-37.2°C

Alert Severity Levels

- **LOW**: Single minor abnormality
- **MEDIUM**: Multiple abnormalities
- **HIGH**: Critical values requiring immediate attention

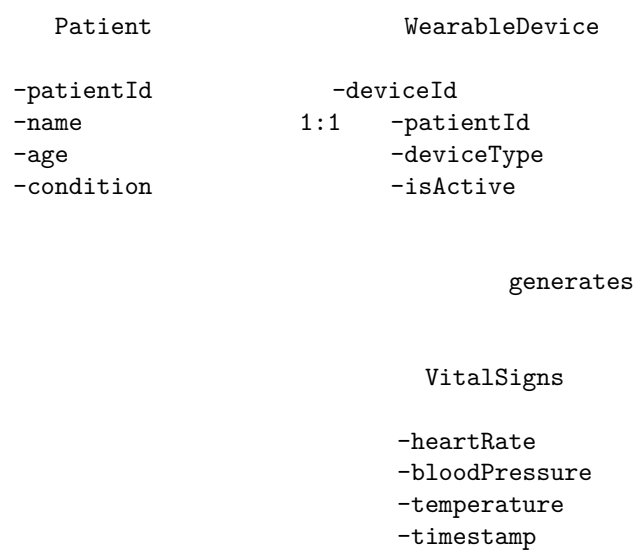
Design Patterns Used

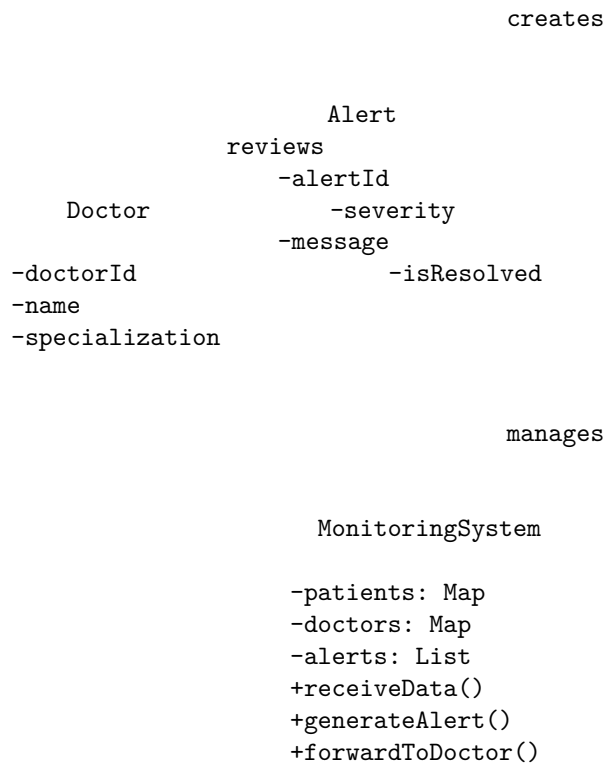
1. **Singleton Pattern**: MonitoringSystem coordinates all operations
2. **Observer Pattern**: Devices notify system of new data
3. **Factory Pattern**: Alert severity determination
4. **Strategy Pattern**: Different recommendation strategies based on condition

Future Enhancements

- Database integration for persistent storage
- Real-time web dashboard for monitoring
- Machine learning for predictive alerts
- Mobile app for patients and doctors
- Integration with actual IoT device APIs
- Multi-hospital support
- Patient portal for viewing own data
- Historical trend analysis

UML Class Diagram





Authors

- Student Name
- Course: Software Engineering
- Institution: Near East University

License

Educational Project - Near East University

GitHub Repository

[Add your GitHub repository link here after uploading]

```
git clone [your-repo-url]
cd RemoteHealthMonitoringSystem
javac *.java
java Main
```