**Report: Optimization of Healthcare System**

**Student Name: Muhammad Usman Khan**

**Student ID: 19901**

**Course: Object Oriented Programming**

**Instructor: Dr. Mehak Usmani**

**1. Problems Identified in the Original Code**

The initial implementation of the healthcare system had multiple design and structural issues:

- Code Duplication: Each class (Doctor, Nurse, Patient) independently declared common fields like name, resulting in redundant code.

- No Inheritance or Polymorphism: No base class was used to represent shared behavior.

- Tight Coupling & Poor Extensibility: Adding new roles required rewriting logic.

- Lack of Composition & Aggregation: Appointments and medical histories were not modeled as separate classes.

- No Exception Handling: Input validation was missing.

- No Use of Interfaces: No behavioral abstraction was defined.

- No Central Management: There was no Hospital class to manage data.

**2. Optimization Techniques Applied**

The refactored code integrates object-oriented design principles:

- Abstraction: Introduced an abstract Person class.

- Inheritance: Doctor, Nurse, Patient, and Receptionist inherit from Person.

- Polymorphism: Method overriding for role-specific behavior.

- Interfaces: Schedulable and Billable interfaces created.

- Composition: Schedule and MedicalRecord classes designed.

- Aggregation: Hospital class to manage collections.

- Exception Handling: Checks for invalid inputs.

- Collections: Used ArrayList for storage.

- Functional Extensions: Department-based doctor search.

**3. Key Design Changes and Their Justification**

|  |  |
| --- | --- |
| **Change** | **Justification** |
| Introduced Person base class | Reduced redundancy and improved maintainability |
| Added interfaces (Schedulable, Billable) | Enabled modular behavior and flexibility |
| Implemented Hospital class | Centralized staff and patient management |
| Designed Schedule and MedicalRecord classes | Increased modularity and data encapsulation |
| Used method overriding and interfaces | Provided role-specific functionality without tight coupling |
| Exception handling | Improved robustness and user input validation |
| Department-based doctor search | Improved system functionality |

**4. Use of OOP Concepts**

- Abstraction: Common features moved to Person superclass

- Inheritance: Hierarchical structure among roles

- Polymorphism: Interfaces and method overriding

- Composition: Appointments and histories via separate classes

- Aggregation: Hospital class manages components

**Conclusion**

The optimized system applies strong object-oriented principles. It eliminates flaws of the original design, adds robust functionality through abstraction and modularity, and ensures scalability for future enhancements.