A screenshot of a diagram

Description automatically generated

**Patient Management System Report**

**Introduction**

This project enhances a Patient Management System by introducing medication and surgery treatment plans, and prescription services. The code follows **SOLID principles** to ensure maintainability, flexibility, and scalability.

**Single Responsibility Principle (SRP)**

Each class has one responsibility:

* **Patient Class**: Manages patient data (name, birth date).
* **Inpatient and Outpatient Classes**: Handle patient-specific attributes (room numbers, followup dates,etc ).
* **PrescriptionService Interface**: Manages prescription-saving logic, separating it from other functionalities.

**Liskov Substitution Principle (LSP)**

Derived classes, like Inpatient and Outpatient, substitute the Patient class without breaking the system. Both classes align with Patient behavior, ensuring no unexpected errors.

**Dependency Inversion Principle (DIP)**

High-level classes depend on abstractions, not concrete implementations:

* The **Physician** class interacts with TreatmentPlan and PrescriptionService interfaces, promoting loose coupling and making it easier to swap or extend these services.

**Newly Implemented Classes**

1. **TreatmentPlan Interface**: Defines a method to create treatment plans.
2. **MedicationTreatmentPlan**: Manages outpatient medication treatment.
3. **SurgeryTreatmentPlan**: Manages inpatient surgery plans.
4. **OnlinePrescriptionService and PrintablePrescriptionService**: Save prescriptions in different formats.

**Modifications**

* **Patient Class**: Abstract class representing general patient details.
* **Inpatient and Outpatient**: Specialized subclasses handling hospitalization details.
* **Prescription Class**: Stores disease, dosage, and follow-up details.

**JUnit Testing**

JUnit tests verify:

* Treatment plans adjust properly based on patient data.
* Prescription services save prescriptions in correct formats.