



TECHNICAL INTELLIGENCE REPORT

HospitalSystem

Generated on: 2026-01-10 06:53

Class: Staff

PURPOSE:

Represents a generic staff member within a hospital system.

PATTERNS:

Inheritance (base class for specialized staff roles).

RELATIONSHIPS:

Inherited by `Doctor` and `Nurse`.

COMPLEXITY LEVEL:

Low

SECURITY NOTE:

No input validation for `staff_id` or `name`, allowing potential injection of invalid types/values.

IMPACT ANALYSIS:

Changing attributes may break subclass (`Doctor` / `Nurse`) initialization logic.

BEST PRACTICES:

Attributes lack encapsulation (no getters/setters). Consider using properties for future validation.

Constructor: __init__

DESCRIPTION:

Initializes a Staff instance with ID and name.

LOGIC FLOW:

1. Assign `staff_id` to instance.
2. Assign `name` to instance.

PARAMETERS:

- `staff_id` : Unique identifier for staff (integer).
- `name` : Full name of staff (string).

RETURNS:

None

ERROR HANDLING:

No validation; fails if non-integer `staff_id` passed.

Architectural Recommendations:

Convert to an abstract class if all staff must have specialized roles.

Class: Doctor

PURPOSE:

Represents a medical doctor with specialized fields and patient management capabilities.

PATTERNS:

Inheritance (extends `Staff`).

RELATIONSHIPS:

Inherits from `Staff`. Aggregation with `Patient` via `patients` list.

COMPLEXITY LEVEL:

Low

SECURITY NOTE:

`prescribe_medication()` exposes patient data without access control.
Medicine input is unvalidated.

IMPACT ANALYSIS:

Modifying `specialty` does not affect prescription logic. The `patients` list is unused and therefore does not affect functionality.

BEST PRACTICES:

Unused `patients` list violates YAGNI principle; remove or implement patient-management methods.

Constructor: __init__

DESCRIPTION:

Initializes a Doctor instance with specialty and empty patient list.

LOGIC FLOW:

1. Call parent (`Staff`) constructor.
2. Assign `specialty` to instance.
3. Initialize empty `patients` list.

PARAMETERS:

- `staff_id` : Unique identifier (integer).
- `name` : Doctor's name (string).
- `specialty` : Medical specialty (string).

RETURNS:

None

ERROR HANDLING:

No validation for `specialty` type/length.

Method: prescribe_medication()

DESCRIPTION:

Generates a prescription message for a patient.

Source Code Analysis

HospitalSystem

CODE

```
class Staff:
    def __init__(self, staff_id: int, name: str):
        self.staff_id = staff_id
        self.name = name

class Doctor(Staff):
    def __init__(self, staff_id: int, name: str, specialty: str):
        super().__init__(staff_id, name)
        self.specialty = specialty
        self.patients = []

    def prescribe_medication(self, patient, medicine):
        return f"Doctor {self.name} prescribed {medicine} for {patient.name}"

class Nurse(Staff):
    def __init__(self, staff_id: int, name: str, shift: str):
        super().__init__(staff_id, name)
        self.shift = shift

class Patient:
    def __init__(self, patient_id: int, name: str, illness: str):
        self.patient_id = patient_id
        self.name = name
        self.illness = illness

class Hospital:
    def __init__(self, hospital_name: str):
        self.hospital_name = hospital_name
        self.staff_members = []
        self.departments = ["Emergency", "Radiology", "Surgery"]

    def hire_staff(self, member: Staff):
        self.staff_members.append(member)
```