Flow of Execution of Hospital Management System

Scenario: Managing Patients in a Hospital

1. Patient Registration and Checkin:

Reallife Action: A new patient arrives at the hospital to register.

Code Execution:

- The hospital staff registers the patient using the `register_patient(patient)` method.
- This method adds the patient to the AVL Tree (`self.avl.insert(self.avl.root, patient)`) to keep the records sorted by `patient_id`.
- The patient is also automatically checked in, meaning they are now part of the hospital's managed records.

2. Scheduling Appointments:

Reallife Action: Patients request an appointment with a doctor.

Code Execution:

- When a patient schedules an appointment, the `schedule_appointment(patient)` method is called.
- This method enqueues the patient in the Queue (`self.appointment_queue.enqueue(patient)`) to maintain the order of appointments based on the time they were booked.

3. Handling Emergencies:

Reallife Action: An emergency patient arrives at the hospital.

Code Execution:

- The hospital staff pushes the emergency patient to the Stack using `handle_emergency(patient)`.
- This method uses `self.emergency_stack.push(patient)`, ensuring the most recent emergency is prioritized (LIFO order).

4. Processing Appointments and Emergencies:

Reallife Action: Doctors see patients based on their appointment schedule and handle emergencies as they occur.

Code Execution:

- To get the next appointment, the hospital uses `get_next_appointment()`, which dequeues the first patient from the Queue (`self.appointment_queue.dequeue()`).
- For emergencies, `get_next_emergency()` pops the last patient from the Stack (`self.emergency_stack.pop()`), handling the most urgent cases first.

5. Patient Checkout:

Reallife Action: A patient completes their visit and checks out of the hospital.

Code Execution:

- When a patient checks out, the hospital staff uses `check_out_patient(patient_id)`.
- This method removes the patient from the AVL Tree using `self.avl.delete(self.avl.root, patient_id)`, ensuring that the records are updated and balanced after the deletion.

6. Searching for Patient Records:

Reallife Action: The hospital staff needs to quickly access a patient's medical history.

Code Execution:

- The `search_patient(patient_id)` method is called to find a patient in the AVL Tree (`self.avl.search(self.avl.root, patient_id)`).
- The AVL Tree ensures that the search operation is efficient and fast (O(log n) time complexity).

Summary of Execution Flow:

- AVL Tree is used to manage patient records efficiently, allowing fast search, insert, and delete operations.
- Queue is used for managing patient appointments in a firstcome, firstserved order.

- Stack is used to manage emergencies, allowing the most recent emergency to be handled first.
- The HMS system handles all these operations smoothly, ensuring a wellorganized and efficient management of hospital operations.

HMS system ensures that patient management in a hospital setting is both efficient and responsive to realworld demands, such as scheduled appointments and unpredictable emergencies.