1. Introducción

En este laboratorio usted deberá diseñar e implementar un sistema de gestión y simulación de pacientes en **unidades de urgencia hospitalaria**, utilizando estructuras de datos avanzadas como colas de prioridad, pilas, mapas y montones (heaps). La implementación estará basada en la categorización oficial del **Ministerio de Salud de Chile**, que clasifica a los pacientes en cinco niveles de atención según su gravedad y el tiempo máximo esperado de respuesta:

- **C1 – Emergencia Vital:** atención inmediata.
- **C2 – Urgencia / Alta Complejidad:** hasta 30 minutos.
- **C3 – Mediana Complejidad:** hasta 1 hora y 30 minutos.
- **C4 – Baja Complejidad:** hasta 3 horas.
- **C5 – Atención General:** sin tiempo máximo, depende de demanda.

1.1 Objetivo del laboratorio

El objetivo de este laboratorio es combinar conceptos de estructuras de datos con simulación de eventos, análisis algorítmico y evaluación empírica del sistema bajo condiciones realistas y cambiantes.

1.2 Simulación

Su sistema deberá simular, durante un periodo de 24 horas, el flujo de llegada y atención de pacientes en una unidad de emergencia, considerando que:

- **Cada 10 minutos llega un nuevo paciente**, categorizado aleatoriamente según la distribución de probabilidades:
    1. C1: 10%
    2. C2: 15%
    3. C3: 18%
    4. C4: 27%
    5. C5: 30%
- **Cada 15 minutos se atiende un paciente**, con una política de atención priorizada basada en:

1. Categoría de urgencia.
2. Tiempo de espera acumulado desde su ingreso.

En algunos momentos del día, el sistema deberá atender a más pacientes de lo habitual, simulando la respuesta hospitalaria a la demanda asistencial. El flujo seguirá esta lógica: **cada vez que se acumulan tres nuevos ingresos, se atienden dos pacientes** de la cola.

Además del sistema de gestión, usted deberá implementar una **estructura de historial de atención** por paciente, utilizando una **pila** que registre cambios de estado (por ejemplo, si la categorización es corregida).

Finalmente, cada área del hospital (por ejemplo: urgencia adulto, urgencia infantil, SAPU, etc.) será modelada mediante un **montón independiente**, lo que le permitirá aplicar **HeapSort** como herramienta de diagnóstico para analizar la carga de atención en cada área.

2. Implementación

En esta experiencia, usted deberá implementar una serie de clases que simulen el funcionamiento de un sistema de urgencias hospitalarias. El sistema deberá permitir la gestión de pacientes según nivel de prioridad, registro y modificación de sus atenciones, asignación de áreas de atención, y simulación de ingresos y egresos en tiempo real.

A continuación, se detallan las clases a implementar junto con sus atributos y métodos obligatorios:

2.1. Clase `Paciente`

Representa a una persona que requiere atención médica. Tiene los siguientes atributos:

1. **nombre** (`String`): nombre del paciente.
2. **apellido** (`String`): apellido del paciente.
3. **id** (`String`): identificador único (RUN o pasaporte).
4. **categoría** (`int`): nivel de urgencia del paciente. 1 a 5, según clasificación C1 a C5 (aleatorio según probabilidad de sección 1.2).
5. **tiempoLlegada** (`long`): instante de llegada en formato Unix timestamp.

6. **estado** (`String`): puede ser `en_espera`, `en_atencion`, `atendido`.

7. **area** (`String`): area en que debe ser atendido. Un valor entre `SAPU`, `urgencia_adulto` e `infantil`.

8. **historialCambios** (`Stack<String>`): pila que registra cada cambio relevante en la atención o categorización del paciente.

Métodos obligatorios:

- `long tiempoEsperaActual()`: calcula el tiempo transcurrido desde su llegada hasta el momento actual en minutos.

- `void registrarCambio(String descripcion)`: añade un cambio al historial del paciente.

- `String obtenerUltimoCambio()`: obtiene y remueve el último cambio registrado.

2.2. Clase `AreaAtencion`

Representa una zona del hospital (ej. urgencia adulto, infantil, SAPU, etc.), que se modela como un **montón (heap)** donde se insertan pacientes en función de su prioridad. Serán tres: `SAPU`, `urgencia_adulto` e `infantil`.

Atributos:

1. **nombre** (`String`): nombre del área.

2. **pacientesHeap** (`PriorityQueue<Paciente>`): cola de prioridad que mantiene a los pacientes ordenados por:

    - nivel de urgencia (1 a 5), siendo 1 el más urgente.

    - tiempo de espera (prioridad a quien lleva más tiempo en caso de empate).

3. **capacidadMaxima** (`int`): cantidad máxima de pacientes que puede manejar simultáneamente.

Métodos obligatorios:

- `void ingresarPaciente(Paciente p)`: inserta un nuevo paciente en el heap.

- `Paciente atenderPaciente()`: obtiene y remueve al paciente con mayor prioridad.

- `boolean estaSaturada()`: retorna `true` si se alcanzó la capacidad máxima.

- `List<Paciente> obtenerPacientesPorHeapSort()`: devuelve los pacientes según prioridad usando HeapSort.

2.3. Clase `Hospital`

Administra el ingreso, atención y seguimiento de pacientes a nivel global del hospital.

Atributos:

1. **pacientesTotales** (`Map<String, Paciente>`): Un mapa que asocia el identificador único de cada paciente (ID) con el paciente en el hospital, permitiendo un acceso rápido a la información de cada paciente registrado.

2. **colaAtencion** (`PriorityQueue<Paciente>`): cola central que gestiona a los pacientes en espera.

3. **areasAtencion** (`Map<String, AreaAtencion>`): asignación de pacientes a áreas específicas.

4. **pacientesAtendidos** (`List<Paciente>`): historial de todos los pacientes que han sido atendidos.

Métodos obligatorios:

- `void registrarPaciente(Paciente p)`: inserta al paciente en la cola general y en la estructura `pacientesTotales`. También asigna su categoría y su área de atención.

- `void reasignarCategoria(String id, int nuevaCategoria)`: permite actualizar la categoría de un paciente y registrar el cambio en su historial.

- `Paciente atenderSiguiente()`: extrae de la cola general el paciente con mayor prioridad y lo asigna a su área.

- `List<Paciente> obtenerPacientesPorCategoria(int categoria)`: lista de pacientes en espera de una determinada categoría.

- `AreaAtencion obtenerArea(String nombre)`: devuelve el área correspondiente por nombre.

3. Experimentación

La experimentación se basa en la simulación del funcionamiento de una sala de urgencias hospitalaria durante un período de 24 horas, considerando tanto la **llegada de pacientes** como su **atención priorizada**. El proceso implica la **generación aleatoria de pacientes**, su registro, y el seguimiento de la dinámica de atención en función de políticas de prioridad médica y carga operativa.

3.1. Generación de Datos

Debe implementar una clase llamada `GeneradorPacientes` que cree datos de prueba consistentes con la realidad hospitalaria. Esta clase permitirá:

1. Generar una lista aleatoria de pacientes (`List<Paciente>`) de tamaño N

Los atributos de cada paciente se deben generar siguiendo las siguientes reglas:

- **Nombre y Apellido:** sin restricciones, puede usar una lista fija o generarlos aleatoriamente.
- **ID:** debe ser único, puede simularse con un prefijo y un número incremental.
- **Categoría:** se genera de acuerdo a la siguiente distribución de probabilidad:
    - C1 → 10%
    - C2 → 15%
    - C3 → 18%
    - C4 → 27%
    - C5 → 30%

Tiempo de llegada: cada paciente llegará cada 10 minutos, partiendo desde un timestamp base (`00:00` del día simulado). El timestamp de llegada será:

```
llegada_i = timestamp_inicio + (i * 600)
```

- **Estado inicial:** todos deben comenzar con el estado `en_espera`.
- **Historial:** debe iniciarse vacío.

Guarde esta lista en un archivo de texto llamado `Pacientes_24h.txt`.

3.2. Simulación

Implemente una clase `SimuladorUrgencia`, que ejecute una simulación realista de una jornada completa (24 horas). Esta clase debe tener un método:

```
public void simular(int pacientesPorDia)
```

Lógica de la simulación:

- El ciclo de simulación avanza en intervalos de **1 minuto**.
- Cada **10 minutos**, llega un nuevo paciente (utilice los datos generados).
- Cada **15 minutos**, se atiende a un paciente.
- Además, **cada vez que se acumulan 3 nuevos pacientes**, se atienden **2 pacientes** de forma inmediata.

Condiciones clave de atención:

- Los pacientes se atienden en orden de prioridad (categoría más baja = mayor urgencia).
- Si un paciente ha esperado más del **tiempo máximo permitido por su categoría**, debe ser atendido inmediatamente en la próxima oportunidad.
- Solo se tienen las tres areas de atencion mencionadas en la clase Area de Atencion.
- Al atender un paciente, se actualiza su estado a `atendido` y se agrega al historial del hospital.

Datos a almacenar durante la simulación:

1. Tiempos de atención por paciente.
2. Cantidad de pacientes por categoría atendidos en total.
3. Promedios de tiempo de espera por categoría.
4. Lista de pacientes que excedieron el tiempo máximo de espera.

3.3. Pruebas

Realice las siguientes pruebas para validar su sistema:

- **Seguimiento individual:** seleccione un paciente de categoría C4 y registre exactamente cuánto tiempo tarda en ser atendido.

- **Promedio por categoría:** ejecute la simulación 15 veces y calcule el tiempo promedio de atención para cada categoría.

- **Saturación del sistema:** simule con una cantidad mayor de pacientes (por ejemplo, 200) y registre qué categorías se ven más afectadas en términos de demora.

- **Cambio de categoría:** simule el caso de un paciente mal categorizado (ej. C3 que debe ser C1) y utilice el historial (pila) para registrar la corrección.

3.4 Extensión

Observe que la implementación actual puede provocar que pacientes de baja prioridad no sean atendidos durante todo el día (pensando en el caso de una entrada frecuente de pacientes de mayor prioridad que eviten la atención de pacientes de menor prioridad).

Para solucionar este problema diseñe e implemente un nuevo sistema de prioridad que evite una acumulación grande de pacientes, sobre todo en categorías más bajas. Repita la prueba de **promedio por categoría** y concluya si su diseño redujo los tiempos de espera.

4. Análisis

En esta sección, usted debe analizar en profundidad el comportamiento del sistema simulado, evaluar su desempeño, y reflexionar críticamente sobre las decisiones de diseño adoptadas. Para ello, debe responder los siguientes puntos de manera clara, concisa y argumentada.

4.1. Análisis de Resultados Generales

1. ¿Cuántos pacientes fueron atendidos en total al finalizar la simulación?
2. ¿Cuántos pacientes por categoría (C1 a C5) fueron atendidos?
3. ¿Cuál fue el tiempo promedio de espera por categoría?
4. ¿Cuántos pacientes excedieron el tiempo máximo de espera definido para su categoría?
5. ¿Cuáles fueron los peores tiempos de espera registrados por categoría?

4.2. Análisis Asintótico de Métodos Críticos

Analice el tiempo de ejecución teórico (notación Big-O) de los siguientes métodos clave:

a. `registrarPaciente` de la clase `SalaUrgencia`.
b. `atenderSiguiente` de la clase `SalaUrgencia`.
c. `ingresarPaciente` de la clase `AreaAtencion`.
d. `ordenarPacientesPorHeapSort` de la clase `AreaAtencion`.
e. `heapSort` de la clase `HeapSortHospital`.
f. `reasignarCategoria`.

Para cada uno, justifique su análisis según las estructuras utilizadas (cola de prioridad, heap, mapa, lista, pila, etc.).

4.3. Decisiones de Diseño

Describa las principales decisiones tomadas para administrar pacientes, salas y prioridades. En particular:

- ¿Qué estructuras utilizó y por qué?
- ¿Cómo modeló la cola central de atención?
- ¿Cómo administró el uso de montones para áreas de atención?
- ¿Qué ventajas observó en su enfoque?
- ¿Detectó alguna limitación o ineficiencia?

4.4. Ventajas y Desventajas del Sistema

Mencione al menos **dos ventajas** y **dos desventajas** observadas en su implementación. Por ejemplo:

- Ventajas:
    - Permite simular el impacto de distintas distribuciones de pacientes.
    - Usa estructuras eficientes para organizar pacientes en tiempo real.
- Desventajas:
    - No contempla recursos humanos como médicos o turnos.
    - Dificultad para balancear equitativamente las áreas de atención con muchos pacientes de baja prioridad.

4.5. Desafíos Encontrados

Describa al menos **dos desafíos importantes** que enfrentó durante la implementación y simulación. Por ejemplo:

- Gestionar correctamente la prioridad múltiple (categoría y tiempo).
- Sincronizar correctamente el tiempo de llegada, atención y reasignación.

Explique cómo abordó cada desafío y qué alternativas consideró.

4.6. Extensión del Sistema: Turnos Médicos

Proponga un diseño para extender el sistema e incorporar **gestión de turnos de médicos**. Su propuesta debe considerar al menos (Recuerde no implementar solo definir como propuesta):

- Atributos y métodos de una clase `Medico`.
- Relación entre médicos y áreas de atención.
- Lógica para asignar médicos a pacientes (según categoría, especialidad, o turno).
- Qué estructuras usaría (ej. cola circular para turnos rotativos, pila de eventos, etc.).

4.7. Solución del Sistema por pacientes no atendidos

Describa su implementación y explique cómo su solución ayuda a resolver el problema planteado.

5. Informe

5.1. Introducción

En esta sección, explique de forma general el objetivo del laboratorio, el contexto de atención hospitalaria bajo el sistema de categorización C1 a C5, y la motivación de simular un sistema de atención con estructuras eficientes de datos.

5.2. Implementación

Describa de forma resumida cómo realizó la implementación de las clases solicitadas: `Paciente`, `AreaAtencion`, `SalaUrgencia`, `HeapSortHospital` y otras adicionales si las hubiese creado. Destaque:

- Estructuras de datos utilizadas (ej. `PriorityQueue`, `Map`, `Stack`).
- Métodos implementados y si usó métodos auxiliares.
- Para cada método pedido, indicar complejidad temporal (Big-O).
- Justificación de decisiones de diseño.
- Si realizó modificaciones a los requisitos originales, explíquelas y justifique por qué fueron necesarias.

5.3. Experimentación

Documente el proceso completo de simulación:

- Cómo generó los pacientes (cantidad total, parámetros usados, archivo generado).
- Cómo configuró los intervalos de tiempo de llegada y atención.
- Qué pruebas específicas realizó (promedio de atención por categoría, pacientes fuera de tiempo, pruebas de saturación).
- Si utilizó distintas cantidades de pacientes en distintas simulaciones, indíquelo.

Incluya tablas o gráficos.

5.4. Análisis

Esta sección debe abordar detalladamente cada uno de los puntos solicitados en el punto 4 del enunciado del laboratorio:

- Análisis de resultados (cuantitativos y cualitativos).
- Análisis asintótico.
- Decisiones de diseño y sus implicancias.
- Ventajas y desventajas observadas.
- Desafíos enfrentados y soluciones adoptadas.
- Propuesta para la extensión con turnos médicos.

Use subtítulos para organizar esta sección de forma clara.

5.5. Conclusión

Resuma los aprendizajes obtenidos durante el desarrollo del laboratorio. Reflexione sobre la utilidad práctica de aplicar estructuras de datos avanzadas en un escenario real como el de una urgencia hospitalaria. También puede mencionar posibles mejoras o futuras extensiones al sistema.