**Plan de Proyecto: Simulación de Atención en un Hospital**

**Objetivo:** Desarrollar una aplicación en C# que simule el funcionamiento de un área de atención en un hospital, utilizando pilas, colas y listas para gestionar la atención de los pacientes.

**Componentes Principales:**

1. **Llegada de Pacientes:**
   * Los pacientes llegan en un orden aleatorio y se agregan a una cola de espera.
2. **Clasificación de Pacientes:**
   * Clasificación según nivel de gravedad (urgencia, prioridad normal, baja prioridad) utilizando una pila.
3. **Atención de Pacientes:**
   * Los pacientes son atendidos de acuerdo a su clasificación. Se atiende al paciente que llega al frente de la cola y se retira de la pila de clasificación.
4. **Estadísticas:**
   * Mostrar estadísticas sobre el tiempo de espera promedio de los pacientes.
   * Cantidad de pacientes atendidos por cada nivel de gravedad.

**Estructura del Proyecto**

1. **Clases Principales:**
   * Paciente: Representa a un paciente con atributos como nombre, nivel de gravedad, tiempo de llegada, y tiempo de atención.
   * ColaDePacientes: Gestiona la cola de espera de pacientes.
   * PilaDeClasificacion: Gestiona la pila para clasificar a los pacientes según su nivel de gravedad.
   * Hospital: Controla la llegada, clasificación y atención de los pacientes, además de generar estadísticas.
2. **Interfaces y Enums:**
   * Enum NivelDeGravedad { Urgencia, PrioridadNormal, BajaPrioridad }
   * Interfaz IEstadisticas con métodos para calcular estadísticas relevantes.

**Implementación**

1. **Clase Paciente:**

public class Paciente

{

public string Nombre { get; set; }

public NivelDeGravedad Gravedad { get; set; }

public DateTime TiempoDeLlegada { get; set; }

public DateTime? TiempoDeAtencion { get; set; }

public Paciente(string nombre, NivelDeGravedad gravedad, DateTime tiempoDeLlegada)

{

Nombre = nombre;

Gravedad = gravedad;

TiempoDeLlegada = tiempoDeLlegada;

}

}

Clase ColaDePacientes:

public class ColaDePacientes

{

private Queue<Paciente> cola = new Queue<Paciente>();

public void AgregarPaciente(Paciente paciente)

{

cola.Enqueue(paciente);

}

public Paciente ObtenerSiguientePaciente()

{

return cola.Count > 0 ? cola.Dequeue() : null;

}

public int ObtenerCantidadPacientes()

{

return cola.Count;

}

}

Clase PilaDeClasificacion:

public class PilaDeClasificacion

{

private Stack<Paciente> pila = new Stack<Paciente>();

public void ClasificarPaciente(Paciente paciente)

{

pila.Push(paciente);

}

public Paciente AtenderPaciente()

{

return pila.Count > 0 ? pila.Pop() : null;

}

public int ObtenerCantidadPacientes()

{

return pila.Count;

}

}

Clase Hospital:

public class Hospital : IEstadisticas

{

private ColaDePacientes colaDePacientes = new ColaDePacientes();

private PilaDeClasificacion pilaDeUrgencia = new PilaDeClasificacion();

private PilaDeClasificacion pilaDePrioridadNormal = new PilaDeClasificacion();

private PilaDeClasificacion pilaDeBajaPrioridad = new PilaDeClasificacion();

private List<Paciente> pacientesAtendidos = new List<Paciente>();

public void LlegadaPaciente(string nombre, NivelDeGravedad gravedad)

{

var paciente = new Paciente(nombre, gravedad, DateTime.Now);

colaDePacientes.AgregarPaciente(paciente);

ClasificarPaciente(paciente);

}

private void ClasificarPaciente(Paciente paciente)

{

switch (paciente.Gravedad)

{

case NivelDeGravedad.Urgencia:

pilaDeUrgencia.ClasificarPaciente(paciente);

break;

case NivelDeGravedad.PrioridadNormal:

pilaDePrioridadNormal.ClasificarPaciente(paciente);

break;

case NivelDeGravedad.BajaPrioridad:

pilaDeBajaPrioridad.ClasificarPaciente(paciente);

break;

}

}

public void AtenderPaciente()

{

Paciente paciente = null;

if (pilaDeUrgencia.ObtenerCantidadPacientes() > 0)

paciente = pilaDeUrgencia.AtenderPaciente();

else if (pilaDePrioridadNormal.ObtenerCantidadPacientes() > 0)

paciente = pilaDePrioridadNormal.AtenderPaciente();

else if (pilaDeBajaPrioridad.ObtenerCantidadPacientes() > 0)

paciente = pilaDeBajaPrioridad.AtenderPaciente();

if (paciente != null)

{

paciente.TiempoDeAtencion = DateTime.Now;

pacientesAtendidos.Add(paciente);

}

}

public double CalcularTiempoEsperaPromedio()

{

if (pacientesAtendidos.Count == 0) return 0;

double tiempoTotalEspera = pacientesAtendidos.Sum(p => (p.TiempoDeAtencion - p.TiempoDeLlegada)?.TotalMinutes ?? 0);

return tiempoTotalEspera / pacientesAtendidos.Count;

}

public int ObtenerCantidadPacientesAtendidos(NivelDeGravedad gravedad)

{

return pacientesAtendidos.Count(p => p.Gravedad == gravedad);

}

}

Interfaz IEstadisticas:

public interface IEstadisticas

{

double CalcularTiempoEsperaPromedio();

int ObtenerCantidadPacientesAtendidos(NivelDeGravedad gravedad);

}

Enum NivelDeGravedad:

public enum NivelDeGravedad

{

Urgencia,

PrioridadNormal,

BajaPrioridad

}

**Desarrollo y Pruebas**

* **Desarrollo del Proyecto:** Utilizar Visual Studio para desarrollar y depurar la aplicación.
* **Pruebas:** Crear casos de prueba que simulen la llegada de pacientes en distintos niveles de gravedad, el procesamiento de los mismos y la generación de estadísticas.
* **Consideraciones Adicionales:**
  + Manejar casos de excepción, como la llegada de pacientes cuando el sistema ya está lleno.
  + Proporcionar una interfaz gráfica sencilla si es posible para mejorar la interacción del usuario.

Este plan proporciona una base sólida para el desarrollo de la aplicación, cubriendo todos los aspectos requeridos y utilizando las estructuras de datos mencionadas.