{Nombre del problema a resolver}

# Análisis del Sistema (30 puntos)

## 1.1 Requisitos funcionales del sistema (5 pts)

Escribir los puntos exactos que debe cumplir tu sistema.

Ejemplo: “El usuario puede crear una nueva solicitud de boletos”, etc.

- El programa debe ser capaz de tener 4 tipos de arrays: 1 de salones pequeños (tamaño de 6 salones), 1 de salones medianos (tamaño de 4 salones), 1 de salones grandes (tamaño de 3 salones), y uno de eventos en espera (tamaño de 10 eventos).

- El usuario debe ser capaz de solicitar la reserva de un salón. Para ello, debe ingresar los datos que el sistema le pida. Una vez se ha ingresado la reserva, esta se almacena en la lista de eventos en espera

- El usuario debe ser capaz de ingresar un nuevo salón al sistema. Para ello, se le pedirá ingresar las características propias del salón. Una vez ha ingresado estas características, el salón se asigna a una lista en base al tamaño que tenga (pequeño, mediano o grande).

- El usuario debe ser capaz de asignar un evento a un salón. Al momento de seleccionar esta opción, el usuario elige que evento quiere asignar. Una vez seleccionado el evento, se despliega la lista de salones disponibles para asignarlo. El usuario elige a que salón quiere asignar el evento, y se verifica que se pueda realizar la asignación. En caso de ser posible, se muestra un mensaje congratulatorio, y se remueve el evento de la lista de espera. En caso de no ser posible, se explica por qué y el usuario debe elegir otro salón para asignar la reserva.

## 1.2 Clases necesarias y su propósito (5 pts)

|  |  |
| --- | --- |
| Clase | Propósito |
| Main | Es la clase de vista, en donde se realizan la mayoría de los mensajes al usuario. Además, desde aquí se manda información que el controlador recibe para hacer las operaciones necesarias. |
| Controlador | Esta clase se encarga de manejar el desarrollo del programa. Tiene objetos de Salon, Reserva y Reglas, y cuenta con diversos métodos que el programa Main usa. |
| Salón | Esta clase representa los salones con los que cuenta el negocio de eventos. Cada salón puede tener múltiples eventos asignados (aunque en diferente hora) |
| Reserva | Esta clase representa las solicitudes de reserva que se tienen en lista de espera. Cada reserva tiene características propias, y en base a ellas se verifica si se puede asignar a un salón o no. |
| Reglas | Esta clase representa las reglas que deben cumplirse para asignar una reserva a un salón. Se tienen 3 reglas diferentes, y todas deben cumplirse para realizar la asignación. Esta clase es estática. |

## 1.3 Atributos de cada clase (10 pts)

Clase: Salón

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Nombre | String | Privada | Representa el nombre del salón, funciona como identificador |
| Tamaño | String | Privada | Representa el tamaño del salón. Debe ser de tipo “pequeño”, “mediano” o “grande” |
| Hora disponible | int | Privada | Representa la ultima hora disponible para asignar un evento a este salón |
| Tiempo de reserva máximo | int | Privada | Representa el tiempo máximo que puede reservarse el salón. No lo asigna el usuario, sino que depende del tamaño del salón. |
| Capacidad máxima | int | Privada | Representa la cantidad máxima de personas que puede mantener el salón |
| Costo de reserva (por hora) | double | Privada | Representa el costo por hora al usar el salón. |

Clase: Reserva

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Nombre del evento | String | Privada | Es el nombre del evento, el cual se usa como identificador |
| Cliente VIP | boolean | Privada | Es el tipo de cliente que hace la reserva. En caso de ser True, el cliente se considera un cliente VIP. En caso de ser False, el cliente se considera un cliente normal |
| Hora de inicio | int | Privada | Es la hora a la que inicia el evento. Debe ser un número positivo |
| Duración de evento | int | Privada | Es la duración del evento, en horas. Debe ser un número positivo |
| Cantidad de personas que atenderán | int | Privada | Es la cantidad de personas que atenderán al evento. Debe ser un número positivo |
| Pago inicial de salon | double | Privada | Es la cantidad de dinero que el cliente pagará inicialmente para usar el salón |

Clase: Reserva

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| N/A |  |  |  |

### 1.4 Métodos de cada clase (10 pts)

Clase: Salón

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros:  Tipo de dato | Resultado:  Tipo de dato | Visibilidad | Propósito |
| Asignar datos | Nueva hora: int, tiempo de reserva: int | void | publica | Este método actualiza la ultima hora en que puede asignarse un salón |

Clase: Reserva

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros:  Tipo de dato | Resultado:  Tipo de dato | Visibilidad | Propósito |
|  |  |  |  |  |

Clase: Reglas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros:  Tipo de datos | Resultado:  Tipo de dato | Visibilidad | Propósito |
| Verificar capacidad | Salón deseado: Salón  Cantidad de asistentes: int | boolean | Pública | Este método verifica si se tiene la cantidad de asistentes necesaria para reservar el salón. Esto depende del tamaño del salón: 3/4 para salones pequeños, 5/8 para salones medianos, y 2/3 para salones grandes. Si no se cubre la cuota, se regresa False. De lo contrario, se regresa True |
| Verificar hora | Salón deseado: Salón  Hora de inicio: int | boolean | Pública | Este método verifica si el salón está disponible a la hora que se necesita. En caso de no estarlo (hora posible > hora de inicio), se regresa False. De lo contrario, se regresa True. |
| Verificar tiempo de reserva | Salón deseado: Salón  Duración de evento: int | boolean | Pública | Este método verifica si las horas que durará el evento es adecuada al salón deseado. Cada salón cuenta con un máximo de duración de evento, por lo que si se excede el número de horas que dura el evento, se regresa False. De lo contrario, se regresa True. |
| Verificar presupuesto | Salón deseado: Salón  Pago inicial: int | boolean | Pública | Este método verifica que el pago inicial de la reserva sea suficiente para reservar el salón. Es necesario pagar al menos la mitad del pago total para poder reservar el salón (pago total = horas de evento \* costo de salón). Si no puede pagarse, se regresa False. De lo contrario, se regresa True. |

## Agregar las necesarias

### 2. Diseño: Diagrama de Clases (30 puntos)

* Asegúrate de mostrar atributos y métodos con visibilidad (+, -).
* Indica relaciones entre clases (asociación, agregación, etc.).
* Incluye el driver program (Main).

Diagrama de clases aquí o adjunto en un archivo aparte.

### 3. Programa (40 puntos)

En cada archivo `.java`, asegurarse de incluir:

* Las clases necesarias.
* Uso adecuado de objetos.

Menú que debe implementar el driver program:

* 1. Nuevo comprador
  2. Nueva solicitud de boletos
  3. …
  4. …
  5. Salir

GitHub: colocar aquí la URL:

<https://github.com/Oscarro11/Manager-de-salones->

# Checklist antes de entregar

Está claro el análisis?

El diagrama tiene los elementos UML correctamente?

Subiste tu código a GitHub con todo lo necesario?