{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. No lo asigna el usuario, empieza en 0 |
| Tiempo de reserva máximo | int | Privada | Representa el tiempo máximo que puede reservarse el salón. |
| Capacidad máxima | int | Privada | Representa la cantidad máxima de personas que puede mantener el salón. No lo ingresa el usuario, se decide por el tamaño del 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: Reglas

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

Clase: Controlador

|  |  |  |  |
| --- | --- | --- | --- |
| Atributo | Tipo de dato | Visibilidad | Propósito |
| Lista de salones pequeños | Salon[6] | Privada | Esta lista contiene todos los salones pequeños con los que cuenta el establecimiento. |
| Lista de salones medianos | Salon[4] | Privada | Esta lista contiene todos los salones medianos con los que cuenta el establecimiento. |
| Lista de salones grandes | Salon[3] | Privada | Esta lista contiene todos los salones grandes con los que cuenta el establecimiento. |
| Lista de espera | Reserva[10] | Privada | Esta lista contiene todas las reservas que aun no se han asignado a salones. |
| Cont de salones pequeños | int | Privada | Esta variable contiene la última posición disponible para ingresar un salón pequeño a su lista respectiva. Empieza en 0 |
| Cont de salones medianos | int | Privada | Esta variable contiene la última posición disponible para ingresar un salón mediano a su lista respectiva. Empieza en 0 |
| Cont de salones grandes | int | Privada | Esta variable contiene la última posición disponible para ingresar un salón grande a su lista respectiva. Empieza en 0 |
| Cont de lista de espera | int | Privada | Esta variable contiene la última posición disponible para ingresar una reserva a la lista de espera. Empieza en 0 |

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

Clase: Salón

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros:  Tipo de dato | Resultado:  Tipo de dato | Visibilidad | Propósito |
| Actualizar disponibilidad | Nueva hora: int, tiempo de reserva: int | void | publica | Este método actualiza la ultima hora en que puede asignarse un salón |
| Asignar preset | Tamaño: String | void | Privada | Este método se usa junto con el constructor. Su propósito es asignar al salón su capacidad máxima de personas, en base al tamaño que tiene |

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 VIP | Salón deseado: Salón  Reserva VIP: boolean | boolean | Pública | Este método verifica si la reserva puede asignarse a un salon grande. En caso de que el tamaño del salon sea grande, y la reserva no está hecha por una persona VIP, se regresa False. De lo contrario, se regresa True |
| 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: 1/2 para salones pequeños, 2/3 para salones medianos, y 4/3 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  Duracion\_evento: int  Pago inicial: double | 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. |

Clase: Controlador

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Parámetros:  Tipo de datos | Resultado:  Tipo de dato | Visibilidad | Propósito |
| Anadir salón | Nombre: String  Tamaño: String  Costo de reserva: double  Tiempo máximo de reserva: int | boolean | Pública | Este método agrega un nuevo salón a la lista de salones. Antes de añadir el salón, verifica si el contador de su lista correspondiente es igual al tamaño de la lista. En caso de serlo, no se puede añadir el salón al sistema, y se regresa False. De lo contrario, se crea un nuevo objeto de Salón, se añade a la lista correspondiente y se actualiza el contador de la lista. Después de esto, se regresa True |
| Anadir reserva | Nombre: String  Cliente VIP: boolean  Hora de inicio: int  Duración de evento: int  Cantidad de asistentes: int  Pago inicial: double | boolean | Pública | Este método agrega una nueva reserva a la lista de espera. Antes de añadir la reserva, verifica si el contador de la lista de espera es igual al tamaño de la lista. En caso de serlo, no se puede añadir la reserva al sistema, y se regresa False. De lo contrario, se crea un nuevo objeto de Reserva, se añade a la lista de espera y se actualiza el contador correspondiente. Después de esto, se regresa True. |
| Asignar reserva | Nombre de reserva a asignar: String  Nombre de salón deseado: String | int | Pública | Este método asigna una reserva al salón deseado. Se busca la reserva en la lista que coincida con el nombre ingresado, lo mismo con el salon. Se comprueba si la reserva cumple con las reglas necesarias para ser asignado al salón deseado. Si cumple con estas reglas, el salón se actualiza con la nueva hora disponible, se quita la reserva de la lista de espera y se regresa 1. De lo contrario, se regresa un código de 2 a 5, dependiendo de la regla que no se cumplió por parte de la reserva. |

## 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).

<https://lucid.app/lucidchart/b2f9cbea-99f6-4b1a-a100-71eff16c504c/edit?viewport_loc=-207%2C-967%2C2558%2C1408%2C0_0&invitationId=inv_7b6be7cd-6054-4706-b327-c8b6ec71e2eb>

### 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?