Proyecto Semestral

primavera 2016

Miguel Romero V. (sección 1)
Juan Carlos Figueroa (sección 2)

1 Enunciado

1.1 Aspectos generales

Una ciudad de nuestro país cuenta con un Teatro Municipal, que es el lugar donde se realizan diversas actividades artísticas, culturales y recreativas. Para su administración se cuenta con un sistema automatizado, que se encarga, entre otros, de registrar la información respecto de los eventos desarrollados en el teatro, la venta de entradas y los espectadores que han asistido a cada evento.

La alcaldía de la ciudad, quiere incentivar a los ciudadanos a asistir al Teatro y a los distintos eventos que allí se desarrollan. Para esto, ha decidido realizar un análisis del uso que se le dio al Teatro durante este año y le encarga a Ud. que desarrolle de una aplicación que se encargue de procesar la información disponible.

Para la construcción de esta aplicación Ud. dispone de los siguientes antecedentes:

1. El teatro cuenta con 3 tipos de localidades (asientos), platea alta, platea baja y balcón. Cada asiento es identificado de manera única mediante la fila (letras del alfabeto) y su ubicación dentro de la fila (números del 1 al 20). El teatro cuenta con 7 filas en platea baja (letras A-G), 10 filas en platea alta (letras H-P), y 10 filas en el balcón (letras Q-Z). Los asientos dentro de cada fila se numeran de izquierda a derecha.

2. Los espectadores se clasifican en 3 categorías distintas, las cuales son:

   Anónimo, corresponde al espectador que se le asigna un asiento cuando adquiere su entrada.

   Registrado, es aquel espectador que se encuentra identificado como cliente habitual del teatro, por lo que se conoce su nombre, rut, correo electrónico y dirección postal.

   Preferencial, que es un espectador registrado, pero además compra de manera anticipada un abono anual para asistir a todos los eventos del año, para lo cual tiene asignado un asiento fijo.

   Anónimo, corresponde al espectador que se le asigna un asiento cuando adquiere su entrada. Registrado, es aquel espectador que se encuentra identificado como cliente habitual del teatro, por lo que se conoce su nombre, rut, correo electrónico y dirección postal. Preferencial, que es un espectador registrado, pero además compra de manera anticipada un abono anual para asistir a todos los eventos del año, para lo cual tiene asignado un asiento fijo.

3. Por cada evento que se desarrolla en el teatro se registra su fecha, nombre y tipo (por ejemplo: obra de teatro, ballet, concierto de jazz, ópera, etc.).

Como información de entrada para la aplicación que debe desarrollar, Ud. dispone de tres archivos de texto. El primero, que almacena los clientes registrados y preferenciales. El segundo, que registra los eventos que se han desarrollado durante 2016. Por último, el tercero que almacena las ventas de asientos por cada evento. A partir de estos datos, la autoridad municipal espera que se le entreguen algunas estadísticas que ayuden al municipio a programar la siguiente temporada de eventos y a motivar a los ciudadanos a ir al teatro. La estructura de los archivos mencionados es la siguiente:

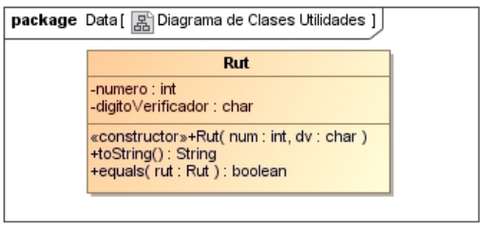

Figure 1: Diagrama de la clase Rut, en UML 2.0

Archivo de clientes (`Clientes.txt`), cada línea de este archivo tiene la siguiente estructura:
nombre, rut, dirección postal, tipo de cliente [, asiento asignado]
El tipo de cliente es *1=registrado y 2=preferencial.* Sólo cuando se trata de un cliente preferencial, se incluye el campo asiento asignado.

Archivo de eventos (`Eventos.txt`), que almacena una línea por cada evento, con la siguiente estructura: Nombre evento, día, mes, año, tipo, precio base El día, mes, año y tipo son enteros, el resto de los campos son de tipo `String`.

Archivo de ventas (`Ventas.txt`), donde cada línea del archivo tiene la siguiente estructura: **Fecha venta, nombre evento, rut del cliente, Id. del asiento, rut del cliente, Id. del asiento, rut del cliente, Id. del asiento, ...**
Como se puede observar, este archivo puede contener varias líneas (extensas) por cada evento. La identificación de un asiento corresponde a la letra de la fila-número asiento, por ejemplo: A-5, significa fila A asiento 5. Cuando un cliente es anónimo, en el lugar del rut del cliente viene un caracter *. El archivo contiene sólo los datos de los asientos vendidos, por lo cual aquellos asientos que no se han vendido no aparecen en este archivo. Un cliente (registrado o preferencial) aparecerá una única vez asociado a la venta de un cierto evento, ya que sólo se le vende una entrada.

Los informes que se desean obtener de la aplicación, se especifican a continuación:

1. **Eventos con poca audiencia**. El informe deberá mostrar una línea por cada evento (todos los datos del evento) en el cual la asistencia de espectadores fue inferior al 30

2. **Tipos de Eventos con mayor audiencia**. El informe deberá mostrar una línea por cada tipo de evento en el cual la asistencia de espectadores de al menos el 50

3. **Gráfica de la ocupación de asientos para un evento específico**. Se deberá mostrar por pantalla la distribución de asientos del Teatro (se sugiere un formato tipo matriz), para un evento específico. Esta distribución deberá incluir todas las filas del teatro y cómo dentro de ellas se distribuyeron los asistentes según su tipo. Los asistentes se indicarán mediante la letra inicial de su tipo: P (Preferencial), R (registrado) y A (Anónimo).

4. **Número de eventos de un cierto tipo a los que ha asistido un cliente en un periodo de tiempo determinado**. Dado el rut de un cliente Registrado o Preferencial, un tipo de evento y un cierto periodo de tiempo (fecha inicio y fecha de término), el informe deberá mostrar el nombre y dirección postal del cliente, así como el número de eventos del tipo indicado al que ha asistido en el periodo señalado.

5. **Recaudación de dinero para un evento específico**. Dado el nombre de un evento mostrar el total recaudado en dicho evento por concepto de venta de entradas.

Finalmente, con respecto al diseño de la aplicación que deberá desarrollar, ya se cuenta con un diagrama de clases, que se presenta en la Figura 2. La Figura 1 muestra la estructura de la clase Rut que debe ser utilizada en la solución.

Es importante destacar que la aplicación debe ser construida basándose estrictamente en estos diagramas. Es decir crear todas clases, relaciones y métodos presentes, asegurándose de utilizar todos y cada uno de estos últimos.

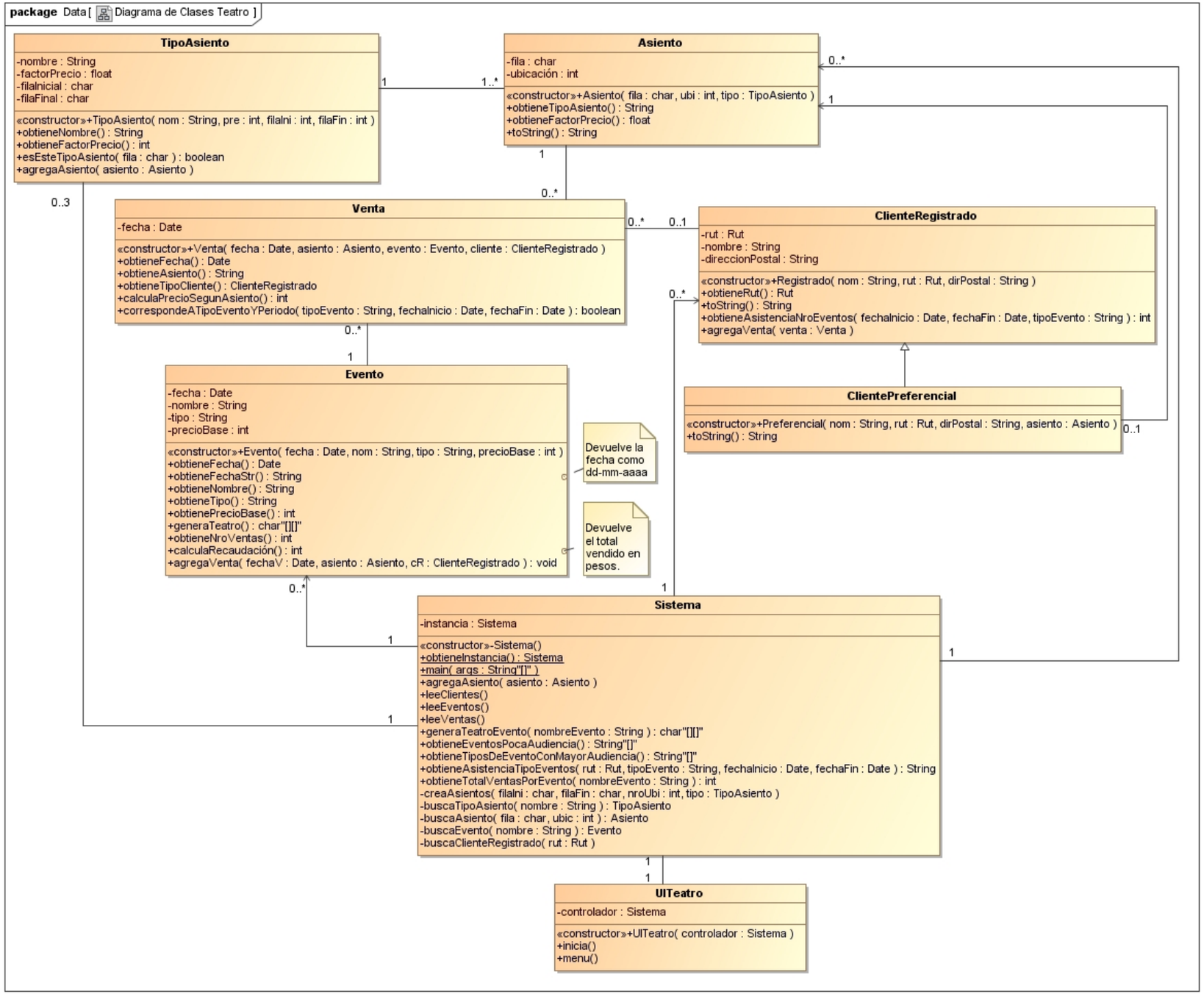

Figure 2: Diagrama de clases de la aplicación requerida, en UML 2.0.

2 Descripción de clases de la aplicación

A continuación se presentan antecedentes importantes de algunas de las clases que conformarán la aplicación que se le solicita implemente y que están presentes en el diagrama de clases de la Figura 1.

2.1 Clase `UITeatro`

Se trata de una clase que permite crear un objeto que maneja la interacción con el usuario, dicha interacción será de naturaleza ASCII, no gráfica. Dicho objeto sólo se encarga de la lectura de los datos que ingrese el usuario y de desplegar los resultados solicitados por este, siendo el único objeto encargado de ello en toda la aplicación. Para realizar las operaciones que escoja el usuario, se presentará un menú de opciones, el usuario escogerá la opción que desee y una vez obtenidos los datos que se pudieran requerir se invocarán los métodos del controlador (objeto de la clase Sistema), objeto que realmente realiza las operaciones requeridas por cada opción usando los datos que se le pasarán como parámetro. Si el método invocado retorna alguna información esta será desplegada convenientemente en pantalla por el objeto de la clase aquí descrita (`UITeatro`).

El detalle de la clase `UITeatro` se especifica a continuación.

Atributos privados

Nombre atributo	Tipo	Descripción
`controlador`	`Sistema`	Almacena la referencia al objeto de tipo Sistema al cual se le invocarán los métodos que realizan las acciones que se requieren.

Constructor público

`UITeatro(Sistema controlador)` Sólo asigna el controlador al atributo respectivo.

Métodos públicos

`void inicia()` Tiene como propósito leer los archivos de Clientes, Eventos y Ventas. Esta acción debe realizarla invocando los métodos correspondientes del controlador. En caso de errores al tratar de leer desde los archivos (se reciben excepciones desde los métodos invocados al controlador) debe enviarse un mensaje apropiado a pantalla y concluir la ejecución de la aplicación. Si no se presentan errores, se debe desplegar un mensaje indicando que los datos han sido activados (o leídos) para su uso.

`void menu( )` Presenta en pantalla un menú con las opciones disponibles dando la posibilidad a los usuarios de escoger una de tales opciones. El menú debiera tener el siguiente formato:

```
MENU SISTEMA TEATRO MUNICIPAL

1: Genera gráfica de localidades del teatro para un evento
2: Obtiene eventos con poca audiencia
3: Obtiene tipos de evento con mayor audiencia
4: Lista eventos de un cierto tipo a los que ha asistido un cliente
5: Obtiene total de ventas para un evento
6: Salir

      Ingrese opción:
```

Para llevar a efecto la o las acciones correspondientes a cada opción se deberá invocar el o los métodos apropiados del controlador (objeto de la clase Sistema).

Además se espera que **capture las excepciones que lancen los métodos correspondientes del controlador y despliegue el motivo de éstas, permitiendo continuar con la ejecución de la aplicación. Esta debiera concluir sólo al escoger la opción Salir del menú.**

2.2 Clase `Sistema`

Clase que proporciona todas las operaciones que la aplicación requiere. Esta clase no interactúa con el usuario, es decir, no lee desde teclado, ni despliega mensajes en pantalla.

Atributos privados

Nombre atributo	Tipo	Descripción
instancia	Sistema	Almacena la referencia al objeto de tipo Sistema al cual se le invocarán los métodos que realizan las acciones que se requieren. Este atributo surge como resultado de implementar un patrón de diseño llamado Singleton que tiene como finalidad asegurar que se cree un único objeto de la clase.

Constructor público

`Sistema( )` Crea los objetos de tipo colección que implementan las relaciones con algunas de las otras clases de la aplicación (ver Figura 1). Crea los objetos `TipoAsiento` y los almacena en el objeto tipo colección respectivo. Crea asimismo todos los asientos que correspondan por cada tipo de asiento, para esto el constructor utiliza el método privado `creaAsientos`.

Métodos públicos

`Sistema obtieneInstancia()` Método que forma parte del patrón Singleton. Su contenido es:

```
1  if(instancia == null) {
2    instancia = new Sistema();
3  }
4  return instancia;
```

`static void main(String[] args)` Crea un objeto de la clase `UITeatro` y le invoca los métodos inicia y menu.

`void leeClientes()` Lee los datos del archivo `Clientes.txt`, crea los objetos correspondientes almacenándolos en la colección respectiva. En caso de producirse algún problema en la lectura lanza una excepción del tipo `Exception` (de este modo cubre excepciones `IOException` y `NumberFomatException`).

`void leeEventos( )` Lee los datos del archivo `Eventos.txt`, crea los objetos correspondientes almacenándolos en la colección respectiva. En caso de producirse algún problema en la lectura lanza una excepción del tipo `Exception` (de este modo cubre excepciones `IOException` y `NumberFomatException`).

`void leeVentas()` Lee los datos del archivo Ventas.txt, crea los objetos correspondientes almacenándolos en la colección respectiva. En caso de producirse algún problema en la lectura lanza una excepción del tipo `Exception` (de este modo cubre excepciones `IOException` y `NumberFomatException`).

`char[][] generaTeatroEvento(...)` Retorna una matriz que representa el teatro con las 20*27 localidades o asientos, indicando si este fue ocupado y si lo fue que tipo de cliente lo ocupó (A, R o P, anónimo, registrado o preferencial, respectivamente) para el evento cuyo nombre es pasado como parámetro. En caso de producirse un error, ya sea porque el nombre indicado es no válido o porque no exista un evento con ese nombre, el método lanza una excepción del tipo `DataFormatException` que incluye un mensaje dejando claramente establecido el tipo de error que se ha producido.

`String[] obtieneEventosPocaAudiencia()` Retorna un arreglo de `Strings`, donde en cada posición se indican los datos de los eventos (de aquellos registrados) que han tenido una audiencia inferior al 30%. En caso que no existan eventos cumplan con este criterio retorna un arreglo vacío (cuyo length es cero).

`String[] obtieneTiposDeEventoConMayorAudiencia()` Retorna un arreglo de strings, donde en cada posición se indica los tipos de evento (atributo de la clase `Evento`) con una audiencia superior o igual al 50%. Para ello deberá determinar primero los tipos de eventos distintos que existen en la aplicación y, por cada uno, determinar su audiencia. En caso que no existan eventos cumplan con este criterio retorna un arreglo vacío (cuyo length es cero).

`String obtieneAsistenciaTipoEventos(...)` Retorna un string con el rut, nombre, dirección postal y número de eventos de un cierto tipo (indicado como parámetro) a los que ha asistido un cliente cuyo rut se recibe como parámetro en el rango de fechas específico (indicado como parámetro). En caso de producirse un error, ya sea porque uno o más datos pasados como parámetro es no válido o porque no exista un cliente con el rut indicado, el método lanza una excepción del tipo `DataFormatException`] que incluye un mensaje dejando claramente establecido el tipo de error que se ha producido.

`int obtieneTotalVentasPorEvento(...)` Retorna el total de ventas (valor en pesos producto de las entradas pagadas) realizadas para el evento cuyo nombre se recibe como parámetro. En caso de que el nombre dado sea no válido o no exista un evento con dicho nombre, el método lanza una excepción del tipo `DataFormatException` que incluye un mensaje dejando claramente establecido el tipo de error que se ha producido.

Importante Respecto de los métdos: **`void leeCliente( )`**, **`void leeEventos( )`**, **`void leeVentas( )`**:

- deberá cuidar de crear los objetos correspondientes y las relaciones que se deben establecer a través de la invocación de os métodos adecuados de las restantes clases, de modo que se implemente correctamente el diagrama de clases del problema.
- Suponga que todos los datos de los archivos correspondientes son correctos por lo que no necesita validarlos.

Métodos privados

`TipoAsiento buscaTipoAsiento(...)` Busca entre los objetos de la colección correspondiente el objeto de tipo `TipoAsiento` que coincida con la fila indicada como parámetro. En caso de no hallar un objeto que cumpla con el criterio de búsqueda retorna `null`.

`Asiento buscaAsiento(...)` Busca entre los objetos de la colección correspondiente el objeto de tipo `Asiento` que coincida con la denominación indicada como parámetro (A1, F5 u otro). En caso de no hallar un objeto que cumpla con el criterio de búsqueda retorna `null`.

`Evento buscaEvento(...)` Busca entre los objetos de la colección correspondiente el objeto de tipo `Evento` que coincida con el nombre indicado como parámetro. En caso de no hallar un objeto que cumpla con el criterio de búsqueda retorna `null`.

`ClienteRegistrado buscaClienteRegistrado(...)` Busca entre los objetos de la colección correspondiente el objeto de tipo `ClienteRegistrado` que coincida con el rut indicado como parámetro. En caso de no hallar un objeto que cumpla con el criterio de búsqueda retorna `null`.

`void creaAsientos(...)` Crea los asientos correspondientes a las filas y número de posiciones señaladas como parámetro. Cada objeto creado es almacenado en la colección pertinente de la clase `Sistema`, siendo además agregado al objeto `TipoAsiento` que se indica como parámetro.

2.3 Clase `Venta`

A continuación se describen dos de los métodos de esta clase.

Métodos públicos

`int calculaPrecioSegun Asiento()` Determina el valor de la venta (en pesos) en función del factor precio del asiento vendido (que depende del tipo de asiento) y del precio del evento correspondiente.

`boolean correspondeATipoEventoYPeriodo(...)` Retorna verdadero si la venta está asociada a un evento cuyo tipo se recibe como parámetro y si dicha venta fue realizada en el período indicado, en caso contrario retorna falso

2.4 Clase `Evento`

A continuación se describe uno de los métodos de esta clase.

Métodos públicos

`void agregaVenta(...)` Crea una nueva venta incorporándola a la colección correspondiente.

2.5 Clase `ClienteRegistrado`

A continuación se describen dos de los métodos de esta clase.

`String toString()` Retorna el rut, nombre y dirección postal del cliente (en ese orden) separados únicamente por comas (sin bancos).

`int obtieneAsistenciaNroEventos(...)` Retorna el número de eventos del tipo indicado dentro del período establecido (incluyendo las fechas de inicio y fin) a los que ha asistido el cliente.

Importante El método `obtieneAsistenciaNroEventos(...)` deberá consultar a cada uno de los objetos de tipo `Venta` que tiene asociados el cliente, si el evento correspondiente a esa venta cumple con los criterios dados como parámetros (tipo de evento y período), de manera que sea contabilizado como evento al cual asistió el cliente.

2.6 Clase `ClientePreferencial`

A continuación se describe uno de los métodos de esta clase.

`String toString(...)` Retorna el rut, nombre, dirección postal del cliente e identificación del asiento que tiene asociado (en ese orden) separados únicamente por comas (sin bancos). La identificación del asiento corresponde a `FilaUbicación` (ej. A1, G20, etc).

2.7 Clase `Rut`

A continuación se describen dos de los métodos de esta clase.

`String toString()` Retorna el rut con el formato numero+"-"+digito verificador, por ejemplo 19.800.350-1 (el anterior es un ejemplo que sólo se presenta para explicar el formato del string retornado, no necesariamente es un rut válido).

`boolean equals` Retorna verdadero si los ruts que se comparan tienen igual número y dígito verificador.

3 Interfaz de la aplicación

En esta sección se presentan algunas pantallas para la primera versión del software que es en modo texto. Esta sección tienen como finalidad servir de ejemplo acerca del formato que se espera tengan las interacciones con el usuario (entradas y salidas estándares).

Usted puede variar estas pantallas respetando la claridad de las mismas.

```
clientes del Teatro activados
Eventos del Teatro activados
Ventas del Teatros activados

MENU SISTEMA TEATRO MUNICIPAL

1: Genera gráfica de localidades del teatro para un evento
2: Obtiene eventos con poca audiencia
3: Obtiene tipos de evento con mayor audiencia
4: Lista eventos de un cierto tipo a los que ha asistido un cliente
5: Obtiene total de ventas para un evento
6: Salir

      Ingrese opción:
```

Figure 3: Pantalla principal con el menú de opciones

```
Desplegando gráfica de las localidades
Ingrese nombre del evento: carmen de bizet
A - - - - - - - - - - A - - - - - - - - - -
A - - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A - - - - A - - - - - A - - - - - - - - - -
- - - - - - - - - - - A - - - - - - - - - -
- - - - - - - - - - - A - - - - - - - - - -
A A - - - - A A A A A A - - - - - - - - - -
A A - - - - - - - - - A A - - - - - - - - -
A A - - - - - - - - - A A A - - - - - - - -
- - - - - - - - - - - A - - - - - - - - - -
- - - - - - - - - - - A A A A - - - - - - -
- - - - - - - - - - - A - - - - - A A - - -
- A - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - A
- A - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
A A - - - - - - - - - A - - - - - - - - - -
- - - - - A A - - - - - - - - - A - - - - -
```

Figure 4: Despliegue del teatro con la asistencia al evento indicado.

La Figura 3 presenta la pantalla principal en la que se muestra el menú de opciones y se permite al usuario escoger la acción que desea realizar. Las tres líneas iniciales sólo aparecerán al iniciarse la aplicación para indicar al usuario que se cuenta con los datos requeridos para generar las salidas requeridas.

La Figura 4 corresponde al resultado de la primera opción del menú. En este caso los guiones representan asientos vacíos y las letras a los tipos de clientes que han ocupado los asientos representados. Obsérvese que antes de desplegar la asistencia del teatro se ha leído el nombre del evento para el cual se desea conocer tal asistencia. En este caso particular, al evento Camen Bizet, asistieron mayoritariamente clientes anónimos.

La Figura 5 muestra la salida producto de haber seleccionado la opción 2 del menú, esto es, el listado de eventos cuya asistencia fue inferior al 30%, dicho listado debe incluir los datos de los eventos correspondientes. En este caso, se trata de 5 eventos cuyos datos se pueden observar.

La Figura 6 muestra el resultado de la opción 3. En este caso no se encontró ningún tipo de evento que haya contado con al menos el 50% de sus eventos (respecto del total de eventos de la temporada) cuya asistencia haya sido superior al 80%.

La Figura 7 presenta el resultado de haber escogido la opción 4. Nótese que al inicio se deben solicitar todos los antecedentes necesarios para realizar la operación correspondiente a la opción indicada. En este

```
Eventos con audiencia inferior a 30%

nombre, fecha, tipo evento, precio base
-----------------------------------------------
Concierto de Jazz, 12-01-2016, Musical, 12000
Carmen de Bizet, 18-01-2016, Opera, 17000
Casa de Remolienda, 28-01-2016, Teatro, 10000
La casa de Bernarda Alba, 20-03-2016, Teatro, 10000
```

Figure 5: Despliegue del listado de eventos cuya asistencia ha sido inferior al 30%.

```
Eventos con audiencia inferior a 30%

No se encontraron eventos con estas características
```

Figure 6: Despliegue de los tipos de evento cuya asistencia ha sido superior al 80%

```
Eventos al que ha asistido un Cliente....

Ingrese rut: 14.785.002-5
Ingrese tipo de evento: Teatro
Ingrese fecha de inicio (dd/mm/aaaa): 01/01/2016
Ingrese fecha de término (dd/mm/aaaa): 2/12/2016

El cliente Luis Torres Jara con dirección postal O'Higgins 362 ha asistido a 2 eventos

Operación realizada exitosamente
```

Figure 7: Número de eventos de un cierto tipo y en cierto período a los que ha asistido un cierto cliente.

```
Obteniendo total de ventas ...

Ingrese nombre del evento: Carmen de Bizet

Las ventas totales del evento indicado ascienden a: 180200 pesos

Operación realizada exitosamente
```

Figure 8: Despliegue del total de ventas en pesos para un evento indicado.

caso, para los datos indicados, se indica que el cliente Luis Torres Jara, cuya dirección postal se despliega, asistió a 2 eventos durante la temporada.

La Figura 8 muestra la salida producida por la opción 5 del menú. Como se observa, una vez ingresado el nombre del evento, el sistema despliega la cantidad en pesos de las entradas vendidas (ventas) de ese evento. En este caso se trata del evento Camen de Bizet que recaudó $180.200.-

4 Plazos

El proyecto semestral tiene 3 entregas, las cuales deberán ser subidas antes de las 23.00 hrs. en plataforma `Moodle`. Las fechas y contenido de las entregas son las siguientes:

1. **viernes 21 de octubre.** Esta primera entrega es parcial en la cual debe dar cuenta del estado de avance del proyecto. En entrega debe tener como mínimo programadas (sin errores de compilación) todas las clases con sus respectivos atributos y todas las variables de instancia que sean necesarias para implementar las asociaciones y herencia. **Esta entrega no contempla la implementación de los métodos**

2. **viernes 4 de noviembre.** Primera versión del sistema completa (interfaz de texto). Esto significa que debe estar toda la funcionalidad requerida en este informe completada sin errores de compilación ni de ejecución.

3. **Lunes 5 de diciembre.** Segunda versión del sistema completamente terminada. Esta segunda versión implementa nuevas características al sistema, así mismo una interfaz gráfica de usuario y mejoras en la persistencia. Los nuevos requisitos serán informados para la segunda entrega (4-nov).

En cada entrega deberá subir a `moodle`:

- código fuente
- informe en formato `PDF`

Las tareas que no cumplan con lo anterior se considerarán como no entregadas.

5 Características del informe a presentar

Se debe entregar un informe del trabajo realizado cuya estructura y contenido es:

1. **Portada** que incluya: Membrete (esquina superior izquierda), Título (centrado vertical y horizontalmente), Nombre de la asignatura, sección, nombre del profesor, nombre de los autores y fecha, con cada dato en una fila diferente (esquina inferior derecha)
2. **Introducción** (mencione propósito del trabajo realizado y descripción breve de la estructura del informe)
3. **Plan de trabajo**. una lista de tareas a realizar para terminar la primera versión del sistema, indicando una breve descripción de la tarea, día de realización, tiempo estimado, tiempo real (solo para las tareas terminadas) y responsable. Esta lista de tareas debe contemplar algunas referentes al control de la calidad del software como realización de pruebas y/o inspecciones de código.
4. **Descripción del problema** versión resumida del enunciado con sus propias palabras.
5. **Descripción general de la solución** explicación de la forma como se resolvió el problema.
6. **Conclusiones** aspectos relevantes de la solución implementada, lecciones aprendidas, dificultades encontradas, lo que quedó pendiente, propuestas de mejora al trabajo realizado.

Además, considere los siguientes aspectos de presentación cuando realice su informe:

- Adecuada redacción y ortografía
- Tamaño de la hoja: carta
- Letra: Times New Roman, 12 pt
- Interlineado: 1,5
- Párrafos separados por una línea en blanco

6 Otros

- Para el desarrollo de la tarea se deberán formar grupos de 2 personas.
- Las copias (parciales o totales) serán evaluadas con nota 1. Esta nota será compartida por todas las tareas involucradas en la copia.
- Si una tarea no se encuentra en `moodle` o se entrega fuera de plazo será evaluada con nota 1.