Universidad del Valle
Nathalie Valle

Análisis – The Eras Tour

El objetivo es modelar el problema usando programación orientada a objetos, identificando las clases necesarias, sus atributos, métodos y responsabilidades. A través de este enfoque se busca construir un sistema modular claro y funcional que refleje fielmente los requisitos planteados.

Como primero paso en el análisis, es necesario identificar las entidades principales que intervienen en el sistema y definir su estructura interna.

Por esta razón, se identificaron tres clases principales que permiten organizar la lógica del programa: cliente, representa al comprador que desea adquirir lo boletos; ticket, encapsula la lógica para determinar si un ticket es válido y localidad, representa una zona del concierto con capacidad y precio definidos.

Es así como se responde a las siguientes preguntas:

1. **¿Qué propiedades y métodos tendrá cada clase?**

Clase	**Propiedades**	**Métodos**
Cliente	• Nombre: String (esto permite identificar al comprador con su nombre.) • Email: String (esto sirve como medio de contacto y también como identificador único si fuera necesario.) • CantidadBoletos: int (esto representa cuántos boletos desea comprar. Parra verificar si hay suficiente disponibilidad.) • Presupuesto: double (esto es la cantidad máxima de dinero que el cliente puede gastar. Para verificar si el cliente puede pagar los boletos asignados.) • TicketID: int (esto es el número de tictek aleatorio que determina si el cliente puede comprar o no.)	• Cliente(…) (constructor): inicializa los datos del cliente al momento de registrase o iniciar solicitud. • generarTicket(): void, asigna autmaticamente un número aleatorio de 1 a 15,000 para participar en la lotería de compra. • mostrarInfo(): void, impreme los datos del cliente, útil para confirmaciones o depuración. • Getters: permite acceder o modificar las propiedades de forma segura respetando el principio de encapsulamiento.
Ticket	• tikectID: int, guarda el número del cliente que participará en la validación. • a: int, b:int : son los dos valores aleatorios que definin	• Ticket (int ticketID) (constructor), crea un objeto ticket con el número del clientte para poder ser evaluado.

	el rango váido para aceptar un ticket.	• generarRango(): void, genera dos valores aleatorios entre 1 y 15,000, formando el rango dentro del cual un ticket será válido. • esValido: void, este devuelve true si el ticketID del cliente está dentro del rango generado (a a b). Esto permite saber si el cliente puede continuar con la compra.
Localidad	• nombre: String, identifica la zona del concierto (ej. Localidad 1, 5 o 10). • precio: double, define el costo por boleto en esa zona. Este se usa para validar si el cliente puede pagar su compra. • capacidadMaxima: int, indica cuántos boletos puede vender esta localidad (siempre 20). Sirve para validar disponibilidad. • boletosVendidos: int, lleva el control de cuántos boletos se han vendido en esta zona. Se actualiza cada vez que hay una venta.	• Localidad(String nombre, double precio) (constructor), permite crear la localidad con su nombre y precio correspondiente. • disponibles(): int, devuelve cuántos boletos quedan por vender en la localidad. • hayespacio(int cantidadDeseada): boolean, verifica si hay boletos suficientes para satisfacer la solicitud del cliente. • puedepagar(double presupuesto, int cantidad): boolean, verifica si el presupuesto del cliente alcanza para pagar los boletos al precio de esta localidad. • venderBoletos(int cantidad): int, vende la cantidad solicitad, actualiza el contador y devuelve cuántos se vendieron realmente.

2. **¿Qué tipo deben tener las propiedades y métodos de cada clase?**
   Clase Cliente
   Propiedades:
   - **nombre y email:** tipo String, ya que almacenan texto identificativo del usuario.
   - **cantidadBoletos:** tipo int, representa una cantidad entera de boletos.

- **presupuesto:** tipo double, representa una cantidad monetaria con decimales.
- **ticketID:** tipo int, es un número aleatorio entero asignado al cliente.

Métodos:

- **Cliente(...):** constructor, sin tipo de retorno (void implícito).
- **generarTicket():** tipo void, ya que asigna un número aleatorio sin retornar valor.
- **mostrarInfo():** tipo void, solo imprime datos.
- **Getters:** devuelven o modifican String, int o double según la propiedad.

Clase Ticket

Propiedades:

- **ticketID, a y b:** tipo int, ya que son números enteros que representan identificadores y límites de un rango.

Métodos:

- **Ticket(int ticketID):** constructor, sin tipo de retorno.
- **generarRango():** tipo void, genera y asigna dos valores sin retornar.
- **esValido():** tipo boolean, ya que devuelve true o false si el ticketID está dentro del rango [a, b].

Clase Localidad

Propiedades:

- **nombre:** tipo String, representa el nombre de la zona.
- **precio:** tipo double, representa el costo del boleto.
- **capacidadMaxima, boletosVendidos:** tipo int, son cantidades enteras relacionadas con disponibilidad.

Métodos:

- **Localidad(String nombre, double precio**): constructor, sin tipo de retorno.
- **disponibles():** tipo int, devuelve la cantidad de boletos disponibles.
- **hayEspacio(int cantidadDeseada):** tipo boolean, indica si hay espacio suficiente.
- **puedePagar(double presupuesto, int cantidad**): tipo boolean, indica si el cliente puede pagar la cantidad deseada.
- **venderBoletos(int cantidad):** tipo int, devuelve cuántos boletos fueron efectivamente vendidos.

Universidad del Valle
Nathalie Valle

3. **¿Cuáles deben ser los modificadores de visibilidad de los miembros en cada clase?**

Clase Cliente

Miembro	Visibilidad recomendada	Justificación
nombre, email, cantidadBoletos, presupuesto, ticketID	private	Son datos sensibles del cliente. Deben protegerse del acceso directo.
Cliente(...) (constructor)	public	Se necesita para crear objetos desde fuera de la clase.
generarTicket()	public	Necesario para asignar el ticketID desde fuera.
mostrarInfo()	public	Permite mostrar datos del cliente cuando sea necesario.
Getters/Setters	public	Permiten acceder o modificar propiedades privadas de forma segura.

Clase Ticket

Miembro	**Visibilidad recomendada**	**Justificación**
ticketID, a, b	private	Son internos al sistema de validación. No deben ser modificados desde fuera.
Ticket(int ticketID)	public	Permite crear un ticket con el ID del cliente.
generarRango()	public	Se debe llamar para establecer el rango válido.
esValido()	public	Se necesita acceder al resultado de la validación desde otras clases.

Clase Localidad

Miembro	Visibilidad recomendada	Justificación
nombre, precio, capacidadMaxima, boletosVendidos	private	Protege los datos de localidad contra modificaciones externas directas.
Localidad(...) (constructor)	public	Permite crear nuevas localidades.
disponibles()	public	Se debe conocer la cantidad disponible desde fuera.
hayEspacio(int)	public	Se usa para validar solicitudes externas.
puedePagar(double, int)	public	Se necesita para verificar presupuesto desde otras clases.
venderBoletos(int)	public	Ejecuta la acción de venta desde otra clase.
Getters	public	Se usa si se desea mostrar datos de localidad (nombre, precio, etc.).

4. **¿Qué parámetros serán requeridos por los métodos en sus clases?**
   Clase Cliente

   Constructor Cliente(String nombre, String email, int cantidadDeseada, double presupuesto)

   - nombre (String): Identifica al cliente con su nombre.
   - email (String): Sirve como medio de contacto y posible identificador único.
   - cantidadDeseada (int): Indica cuántos boletos desea comprar el cliente.
   - presupuesto (double): Monto máximo de dinero que puede gastar.

   Uso: Estos parámetros inicializan las propiedades del objeto Cliente al momento de su creación.

   Método generarTicket()

   - Parámetros: Ninguno.
   - Por qué: No requiere parámetros, ya que genera internamente un número aleatorio que se asigna como ticketID.

   Método mostrarInfo()

   - Parámetros: Ninguno.

- Por qué: No requiere parámetros, ya que únicamente imprime la información contenida en el objeto.

Clase Ticket

Constructor Ticket(int ticketID)

- ticketID (int): Número aleatorio generado previamente por el cliente (con generarTicket()).

Uso: Se asigna al objeto Ticket para que pueda evaluarse si está dentro de un rango válido.

Método generarRango()

- Parámetros: Ninguno.
- Por qué: No requiere parámetros, ya que internamente se generan dos valores aleatorios que definen el rango válido

Método esValido()

- Parámetros: Ninguno.
- Por qué: no requiere parámetros, ya que la validación se realiza utilizando las propiedades internas del objeto: ticketID, a y b.

Clase Localidad

Constructor Localidad(String nombre, double precio)

- nombre (String): Nombre de la zona del evento
- precio (double): Costo por boleto en esa localidad.

Uso: Inicializa una nueva localidad con su nombre y precio. La capacidad máxima (20 boletos) y boletos vendidos (0) se pueden establecer por defecto internamente.

Método hayEspacio(int cantidadDeseada)

- cantidadDeseada (int): Cantidad de boletos que el cliente quiere comprar.

Uso: Se compara cantidadDeseada con la cantidad de boletos disponibles (capacidadMaxima - boletosVendidos) para determinar si hay espacio suficiente.

Método puedePagar(double presupuesto, int cantidad)

- presupuesto (double): Dinero disponible del cliente.
- cantidad (int): Número de boletos que el cliente quiere comprar.

Uso: Evalúa si el cliente puede pagar cantidad × precio.

Método venderBoletos(int cantidad)

- cantidad (int): Número de boletos que se intenta vender.

Uso: Verifica disponibilidad, actualiza el contador de boletosVendidos y retorna cuántos se lograron vender efectivamente.

5. **¿Cómo proveerá de valores iniciales a sus objetos? ¿Qué valores iniciales les asignara?**

Clase Cliente

Los objetos de la clase Cliente serán creados utilizando los datos que ingrese el usuario al momento de iniciar su solicitud. Los valores iniciales se proveerán a través del constructor de la clase, el cual recibirá como parámetros:

- **nombre:** valor ingresado por el usuario.
- **email:** valor ingresado por el usuario.
- **cantidadBoletos:** cantidad de boletos deseados, ingresada por el usuario.
- **presupuesto:** presupuesto disponible del cliente, también proporcionado por el usuario.

Posteriormente, al objeto cliente se le asignará un valor aleatorio para su ticketID mediante la ejecución del método generarTicket(), que genera un número entre 1 y 15,000.

Clase Ticket

Un objeto de la clase Ticket se crea a partir del ticketID generado previamente por el cliente. Ese valor se pasa como parámetro al constructor de la clase. Luego, mediante el método generarRango(), se asignan internamente dos valores aleatorios que definen el rango válido para evaluar si el ticket es aceptado.

Valores iniciales asignados:

- **ticketID:** valor proveniente del cliente.
- **a y b:** valores aleatorios generados dentro del método generarRango() (ambos entre 1 y 15,000).

Clase Localidad

Las localidades se inicializan al inicio del programa con valores fijos y preestablecidos. Cada localidad tiene un nombre, un precio por boleto, una capacidad máxima de 20 boletos, y un contador de boletos vendidos que inicia en cero.

Los valores iniciales son los siguientes:

Nombre de la localidad	Precio (double)	Capacidad máxima (int)	Boletos vendidos (int)
Localidad 1	100.0	20	0
Localidad 5	500.0	20	0
Localidad 10	1000.0	20	0

Estos objetos de tipo Localidad pueden crearse al inicio del sistema y almacenarse, por ejemplo, en una lista o arreglo para su posterior uso en la selección y validación de ventas.

Universidad del Valle
Nathalie Valle

Diseño