Programación II - Trabajo Práctico Integrador
1er. Cuatrimestre 2025
SEGUNDA PARTE

Fecha de presentación: 12/05/2025(subido al Campus)

Fecha límite de entrega: 26/05/2025 (por el Campus)

En esta segunda parte deben entregar la implementación, el diagrama de clases actualizado, el análisis de complejidad en donde se pida, y el IREP para cada tipo de datos modelado en su solución. Para poder empezar con la segunda parte deben tener aprobado el diseño presentado en la primera parte.

Requerimientos técnicos:

i- Grupos: **El mismo grupo de la primera parte del TP.** Si hay alguna modificación debe ser aprobada por sus docentes de la comisión.

ii- Se deben utilizar donde sea conveniente las herramientas de Tecnologías Java que se vieron en la materia. Al menos una vez deben usarse:

- *Stringbuilder,* cuyo uso debe basarse en la necesidad de modificar el string.
- *Iteradores y Foreach* para recorrer las colecciones de Java

iii- Se deberá utilizar en el desarrollo del trabajo **herencia y polimorfismo,** y al menos 2 de estos conceptos: **sobreescritura, sobrecarga e interfaces**. Como también, en los casos que corresponda, se deberá implementar **clases/métodos abstractos**.

iv- **En el informe (documento) se debe explicar donde utilizaron estos conceptos.**

v- **Escribir el *IREP* de la representación elegida para la implementación de cada TAD**. Debe ser parte de la documentación.

Por otro lado, desde la materia se proveerá

a) Una Interfaz para que se utilice como base para la implementación de la clase principal Ticketek, con la explicación de cada método. **NO SE DEBE MODIFICAR**.
b) Un código cliente con datos para crear los objetos, **NO SE DEBE MODIFICAR**.
c) Una clase de testeo (junit). Será condición necesaria para aprobar esta parte del trabajo que tanto el código cliente como el test se ejecuten sin errores.

- Además de pasar el test de *junit* suministrado junto con el TP, en la corrección se testean los ejercicios con otro junit adicional, por lo que se recomienda que el grupo arme un conjunto propio de testeo acorde a su implementación, antes de entregar el TP. Puede entregarlo también si lo desea.

La entrega se realiza subiendo al Campus el proyecto de Java con su implementación **(Seleccionar solamente los archivos .java)** Se debe subir la documentación con los puntos pedidos previamente por escrito en un archivo Word en lo posible o PDF, junto con el proyecto.

Consideraciones importantes para la implementación y la documentación del trabajo:

La implementación de los TADs debe responder a su diseño presentado en la Primera Parte ***teniendo en cuenta las correcciones que se indicaron/indicarán a cada grupo*. Además, será condición necesaria para aprobar que se cumpla con:**

- Deberá correr satisfactoriamente con el código cliente entregado
- Deberá pasar satisfactoriamente el test junit proporcionado.
- Deberá aprovechar correctamente las estructuras de datos elegidas.
- El código deberá tener implementado el método ***toString*** del TAD principal, lo que implica que se deban implementar los toString de los TADs relacionados.
- Se deberá usar herencia, polimorfismo y abstracción.

Algunas definiciones, modificaciones y aclaraciones al enunciado de la primera parte.

Definiciones:

- Los nombres de las **sedes** son únicos.
- Los **estadios** tienen un único Sector llamado **CAMPO**.
- Los nombres de los **espectáculos** no se repiten.
- Las funciones de los espectáculos se identifican por fecha. Cada espectáculo sólo puede tener una función por fecha.

Modificaciones y aclaraciones al enunciado de la primera parte:

- Punto 3. Al registrar un espectáculo solo se indica el nombre que no puede repetirse. Este nombre se usará para identificar el espectáculo. Para definir las funciones del espectáculo se agregó el punto 14.
- Punto 4. Vender entradas a un usuario debe recibir el nombre del espectáculo, la fecha y la cantidad o el sector y números de asientos, dependiendo del tipo de sede de la función. Además de la contraseña para autenticar al usuario. Devolverá una lista de objetos de tipo IEntrada.
- Punto 5. Donde dice listar sedes cambia a listar funciones. devuelve un String con el listado de funciones con un formato en particular (ver ejemplos en la interfaz).
    - Si es estadio:
      "({FECHA}) {NOMBRE SEDE} - {ENTRADAS VENDIDAS} / {CAPACIDAD SEDE}
    - si no es estadio:
      "({FECHA}) {NOMBRE SEDE} - {NOMBRE SECTOR 1}: {ENTRADAS VENDIDAS 1} / {CAPACIDAD SECTOR} | {NOMBRE SECTOR 2}: {ENTRADAS VENDIDAS 2} / {CAPACIDAD SECTOR 2} ..."
- Punto 8. Anular una entrada, dada la entrada (de tipo IEntrada) y la contraseña del usuario que la compró. **Debe resolverse en O(1).**

- Punto 9. Cambiar una entrada para otra función del mismo espectáculo. Además de la entrada, se recibe la fecha de la función y si corresponde, el sector y número de asiento. Y también, la contraseña para autenticar al usuario. Devuelve una entrada nueva para la función solicitada.
- Punto 10. Calcular el costo de una entrada ya no es necesario porque se dispone del objeto IEntrada, a quien se le pregunta el precio. **Se debe resolver en O(1)**
- Punto 11. Al consultar el valor de la entrada para una función **se debe resolver en O(1)**.

Se agregan las siguientes funcionalidades:

- 13. Consultar lo recaudado por un espectáculo en una sede en particular **en O(1)**.
- 14. Agregar función, luego de registrar un espectáculo, se le deben poder agregar las funciones de a una. Para identificar al espectáculo se usa el nombre del mismo que es único. y para identificar una función se usa el nombre del espectáculo y la fecha.
- 15. Listar todas las entradas vendidas de un espectáculo, es decir, de todas sus funciones. Devuelve una lista de IEntrada.

Con las observaciones anteriores, se deberá implementar el TP, teniendo en cuenta las siguientes condiciones:

- Se debe poder imprimir a Ticketek mostrando sus datos en formato adecuado para poder comprenderlo. Se espera visualizar los usuarios, las sedes y una lista de espectáculos con la cantidad de funciones y la recaudación de cada uno. (condiciones para el toString() de Ticketek).

También se deberá entregar en el documento el siguiente análisis de la complejidad:
Explicar la complejidad lograda y justificar por medio de Álgebra de Órdenes para el punto:
- *8. Anular una entrada.*

Test (JUnit):

Se habilitará el archivo de test en el Moodle, junto a este enunciado, de donde deberán descargarlo. **Se avisará en cuanto esté disponible**.

Código Cliente:

Ya está habilitado junto con este enunciado, en el espacio del Moodle para el TP, de donde deberán descargarlo.

Interfaz:

Ya está habilitado junto con este enunciado. Se utilizará como Interfaz para crear el TAD Ticketek. También se les comparte la interfaz IEntrada que deberán implementar para poder usar correctamente la Interfaz ITicketek.