Programación II - Trabajo Práctico Integrador
2do. Cuatrimestre 2021
SEGUNDA PARTE

Fecha de presentación: 14 de octubre
Fecha de entrega: viernes 29 de octubre

Este Trabajo Práctico consta de dos etapas. La primera requerirá la entrega del análisis del problema y el diseño de la solución propuesta, o sea la especificación de los TADs necesarios, diagrama de clases y la interfaz de la solución. En la segunda etapa se deberá entregar la implementación, cuyas condiciones de entrega se darán posteriormente en un segundo enunciado. El diseño se hará utilizando los conceptos de programación orientada a objetos, que incluyen herencia y polimorfismo.

Requerimientos técnicos para la 2da parte

- Se deben realizar las correcciones del diseño de la 1ra parte antes de comenzar la 2da.
- Se debe hacer el diseño de la estructura de datos y el invariante de representación (IREP) que soporte el diseño.

-El **IREP** tiene que considerar la consistencia interna, o sea, cómo se relacionan entre sí las estructuras de datos de los TADs diseñados. Por ejemplo, cómo se relaciona el registro de los votantes con y sin turnos, los votantes que se presentaron a votar o no lo hicieron, o las capacidades de las mesas de cada tipo con los votantes que se asignaron a esas mesas y otras relaciones que surgen de la implementación realizada.

- Por último, se debe hacer la implementación, la cual debe respetar la interfaz que se entrega con este enunciado y cumplir satisfactoriamente el junit otorgado por la cátedra.

Respecto a la implementación:

- Se deben utilizar al menos 2 tecnologías java: (StringBuilder, For each, Iteradores), además de Junit en sí, en alguna parte del TP.
- Se debe implementar(sobreescribir) el **toString** de las clases principales. Como mínimo se debe mostrar un título (Sistema de Turnos para Votación - UNGS), los votantes en espera para un turno, los votantes con turnos asignados mostrando sus respectivos turnos (número de mesa y franja horaria) y si votó o no. Las mesas habilitadas en el Sistema, mostrando de qué clase son y el nombre de su presidente.
- Se debe implementar(sobreescribir) el equals de Mesa y sus clases derivadas.

Aclaraciones y agregados del problema

- Se quiere conocer **lo más rápido posible** el turno de un votante dado su DNI.
- Se simplifica la asignación de turnos en caso de tener más de una característica
    - Si es trabajador, solo puede votar en una mesa para trabajadores
    - Si es mayor con enfermedad preexistente, se asigna la primera mesa que tenga lugar disponible, pudiendo ser una mesa para mayores o para quienes padecen enfermedades.
- Dada una mesa se quiere conocer los votantes asignados por franja horaria.

NOTA: Las franjas horarias se representan con un número entero que puede ser la hora de inicio de cada una: 8, 9, 10, …, 17.

Interfaz

Para la implementación se debe respetar la siguiente interfaz

```
/* Constructor del sistema de asignación de turnos.
 * Si el parámetro nombre fuera null, debe generar una excepción.
 */
public SistemaDeTurnos(String nombreSistema);

/* Registrar a los votantes. Antes de asignar un turno el votante debe estar registrado
 * en el sistema.
 * Si no lo está, se genera una excepción.
 * Si la edad es menor a 16, se genera una excepción
 */
public void registrarVotante(int dni, String nombre, int edad, boolean enfPrevia, boolean
trabaja);

/* Agregar una nueva mesa del tipo dado en el parámetro y asignar el presidente
 *  de cada una, el cual deberá estar entre los votantes registrados y sin turno asignado.
 *  -  Devuelve el número de mesa  creada.
 *  si el president es un votante que no está registrado debe generar una excepción
 *  si el tipo de mesa no es válido debe generar una excepción
 *  Los tipos válidos son:  "Enf_Preex", "Mayor65", "General" y "Trabajador"
 */
public int agregarMesa(String tipoMesa, int dni);

/* Asigna un turno a un votante determinado.
 *   -  Si el DNI no pertenece a un votante registrado debe generar una excepción.
 *   -  Si el votante ya tiene turno asignado se devuelve el turno como: Número de
 *       Mesa y Franja Horaria.
 *  -  Si aún no tiene turno asignado se busca una franja horaria disponible en una
 *      mesa del tipo correspondiente al votante  y se devuelve el turno asignado, como
 *      Número de Mesa y Franja Horaria.
 *  -  Si no hay mesas con horarios disponibles no modifica nada y devuelve null.
 *  (Se supone que el turno permitirá conocer la mesa y la franja horaria asignada)
 */
public Tupla<Integer, Integer>  asignarTurno(int dni);

/* Asigna turnos automáticamente a los votantes sin turno.
 *  El sistema busca si hay alguna mesa y franja horaria factible en la que haya disponibilidad.
 *  Devuelve la cantidad de turnos que pudo asignar.
 */
public int asignarTurno();

/* Hace efectivo el voto del votante determinado por su DNI.
 *  Si el DNI no está registrado entre los votantes debe generar una excepción
 *  Si ya había votado devuelve false.
```

```
 *  Si no, efectúa el voto y devuelve true.
 */
public boolean votar(int dni);

/*
 * Cantidad de votantes con Turno asignados al tipo de mesa que se pide.
 * - Permite conocer cuántos votantes se asignaron hasta el momento a alguno
 *     de los tipos de mesa que componen el sistema de votación.
 * - Si la clase de mesa solicitada no es válida debe generar una excepción
 */
public int votantesConTurno(String tipoMesa);

/ * Consulta el turno de un votante dado su DNI. Devuelve Mesa y franja horaria.
 * - Si el DNI no pertenece a un votante genera una excepción.
 * - Si el votante no tiene turno devuelve null.
 */
public Tupla<Integer, Integer> consultaTurno(int dni);

/* Dado un número de mesa, devuelve una Map cuya clave es la franja horaria y
 *  el valor es una lista con los DNI de los votantes asignados a esa franja.
 *  Sin importar si se presentaron o no a votar.
 *  - Si el número de mesa no es válido genera una excepción.
 *  - Si no hay asignados devuelve null.
 */
public Map<Integer,List< Integer>> asignadosAMesa(int numMesa);

/*
 *  Consultar la cantidad de votantes sin turno asignados a cada tipo de mesa.
 *  Devuelve una Lista de Tuplas donde se vincula el tipo de mesa con la cantidad
 *  de votantes sin turno que esperan ser asignados a ese tipo de mesa.
 *  La lista no puede tener 2 elementos para el mismo tipo de mesa.
 */
public List<Tupla<String, Integer>> sinTurnoSegunTipoMesa();
```