**Trabajo Practico N°1**

**TAD Sistema de Vacunación**



**Integrantes:**

Barraza, Mauro

Luque Materazzi, Lautaro

**Profesores:**

Bertuchelli Daniel

Escobar Javier

**Programación II - Trabajo Práctico Integrador**

**1er. Cuatrimestre 2021**

**PRIMERA PARTE**

**Fecha de presentación: 6 de mayo 2021**

**Fecha de entrega: 14 de mayo 2021**

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 primera parte**

Se puede trabajar en grupos de 2 personas (recomendado) o individualmente. En ambos casos se debe enviar, para la siguiente clase, si trabajarán en grupo o individualmente y el o los nombres de cada grupo.

El diseño de la solución propuesta debe utilizar, al menos una vez, Herencia de manera adecuada.

**Detalles del problema**

A partir de 2021 la UNGS comenzó a prestar servicios de centro de vacunación.

A tales fines las autoridades nos pidieron ayuda para gestionar la administración de vacunas.

A la UNGS van llegando vacunas (1): pfizer, Sputnik, Sinopharm, Moderna y Astrazeneca.

Las vacunas Pfizer y Sputnik son para mayores de 60 y Sinopharm, Moderna y Astrazeneca para todo público. Pfizer y Moderna se almacenan a -18 grados centígrados mientras que las otras a 3 grados.

Se inscribe a la población mayor de 18 años que soliciten vacunarse. Los datos a registrar son el DNI, la edad, si trabaja en salud y si tiene enfermedades preexistentes que lo convierta en persona de riesgo (tiene prioridad de vacunación).

La asignación de turnos se realiza con la siguiente prioridad:

1. Los trabajadores de la salud.

2. Mayores de 60 años,

3. Personas con enfermedades preexistentes.

4. El resto de la población.

El sistema debe poder generar turnos para vacunación teniendo en cuenta las vacunas disponibles y la prioridad definida anteriormente.

A partir de una fecha se generan los turnos de manera consecutiva según la capacidad de vacunación diaria de la universidad, hasta que se acaben las vacunas disponibles o la población inscripta que aún no fue vacunada. Cada turno debe indicar la persona y la vacuna que le será aplicada.

Se debe generar un listado por fecha con las personas que tengan turno para esa fecha.

También, se espera poder generar un reporte que informe las vacunas aplicadas y a que persona se aplicó cada una.

Por último, es necesario conocer las personas que se encuentran en lista de espera.

(1) Toda esta información no es real, es solo a fines didácticos del TP. Por simplicidad asumimos que solo se aplica 1 dosis.

**TAD CENTRO VACUNACION**

**Atributos:**

nombreCentro: String

capAplicacionDiaria:int

turnos: LinkedList<Turno>

personasVacunadas: Map<Integer, String>

pacientesConTurno: ArrayList<Integer> // Lista de los DNI de los pacientes que ya tienen turno

prio1: Queue<Paciente> // Cola de Pacientes con prioridad asignada en 1

prio2: Queue<Paciente> // Cola de Pacientes con prioridad asignada en 2

prio3: Queue<Paciente> // Cola de Pacientes con prioridad asignada en 3

prio4: Queue<Paciente> // Cola de Pacientes con prioridad asignada en 4

gv: GestionVacunas // Administrador de vacunas

**IREP:**

IREP:

nombreCentroVacunacion tiene que ser “UNGS” o “UNGS 2”

capAplicacionDiaria debe ser estricatemente mayor que 0

**Operaciones:**

Crear: Recibe como parámetro el nombre del centro, y la capacidad de vacunación diaria. En caso de que el nombre del centro no sea valido o la capacidad de vacunación diaria sea menor o igual que cero, se lanza una excepción, caso contrario crea el constructor.

ingresarPaciente: Recibe como parámetros el dni, la fecha de nacimiento, si tiene padecimientos y si es empleado de salud. En caso de que el dni ya esté en la lista de espera o el dni ya se encuentre dentro de la lista de personas vacunadas o la edad del paciente sea menor a 18 se lanza una excepción. Caso contrario se agrega al paciente en la lista de espera y en la prioridad1, prioridad2, prioridad3 o prioridad 4, dependiendo de la prioridad de dicho paciente.

Ingresar vacuna: Recibe como parámetro el nombre de la vacuna, la cantidad, y la fecha de ingreso. En caso de que la cantidad pasada por parámetro sea menor o igual a 0 se lanza una expecion. Caso contrario se ingresa la vacuna al stock.

generarTurnos: Recibe como parámetro una fecha. En caso de que la fecha pasada por parámetro sea una fecha “menor” que la fecha actual, se lanza una excepción. Verifica si hay vacunas vencidas y las quita del sistema. Regresa las vacunas de los turnos vencidos y borra a las personas que no asistieron al turno. Luego se asigna turnos por prioridad, quitando al paciente de la lista de espera y agregándolos a la lista de pacientes con turno, respetando siempre la capacidad diaria de vacunación, dado que, si dicho atributo llega a 0, se avanza un día. En caso de que no haya más vacunas o no haya más pacientes en la lista de espera se termina la generación de turnos.

vacunarInscripto: Recibe por parámetro un dni y una fecha de vacunación. Si rl dni pasado por parámetro no se encuentra en la lista de pacientes con turno se lanza una excepción. En caso de que sí esté, consulta si la fecha turno del paciente tiene asignado la misma fecha que la pasada por parámetro y lo agrega a la lista de personas vacunas y concluye el turno.

vacunasDisponibles: retorna la cantidad de vacunas disponible en stock

vacunasDisponibles: recibe por parámetro el nombre de una vacuna y devuelve la cantidad de vacunas de la vacuna pasada por parámetro.

listaDeEspera: devuelve la lista de espera.

TurnosConFecha: recibe como parámetro una fecha. Devuelve un arreglo nuevo de enteros con los dni de los pacientes que tienen turno la misma fecha pasada por parámetro.

reporteVacunacion: devuelve la lista de personas vacunadas.

reporteVacunaVencidas: devuelve las vacunas vencidas.

**Firmas:**

public CentroVacunacion(String nombre, int capacidadVacunacionDiaria);

public void inscribirPersona(int dni, Fecha nacimiento, boolean tienePadecimientos, boolean esEmpleadoSalud);

public void generarTurnos(Fecha fechaInicial);

public void vacunarInscripto(int dni, Fecha fechaVacunacion);

public int vacunasDisponibles();

public int vacunasDisponibles(String nombre);

public List<Integer> listaDeEspera();

public List<Integer> turnosConFecha(Fecha fecha);

public Map<Integer, String> reporteVacunacion();

public Map<String, Integer> reporteVacunasVencidas();

**TAD PACIENTE**

**Atributos:**

DNI: Integer

nacimiento: Fecha

esEmpleadoSalud: boolean

tienePadecimientos: boolean

prioridad: int

**IREP:**

DNI: debe ser univoco

edad: Debe tener Valores mayores a 0 (cero).

prioridad debe ser 1,2,3 o 4

**Operaciones:**

Crear: recibe como parámetros los atributos DNI, tienePadecimientos, esEmpleadoSalud y crea al paciente. Dentro del constructor, además, se le asigna la prioridad mediante el método definirPrioridad().

definirPrioridad: Devuelve un int dependiendo los atributos edad, tienePadecimientos y esEmpleadoSalud.

getPrioridad: devuelve el atributo prioridad de paciente

getDNI: devuelve el atributo DNI de paciente

getEdad: devuelve el atributo edad de paciente

**Firmas:**

public Paciente(int dni, Fecha nac, boolean tienePadecimientos,boolean esEmpleadoSalud);

private int definirPrioridad ();

public int getPrioridad();

public int getDNI();

public int getEdad();

**TAD VACUNA**

**Atributos:**

nombre: String

edadMinimaAplicable: Int

tempAlmacenamiento: int

cantidad: Int

aplicadas: Int

fechaIngreso: Fecha

**IREP:**

edadMinimaAplicable debe ser mayor a cero.

cantidad debe ser mayor a cero

nombre debe ser “Pfizer”, “Sputnik”, “Moderna”, “AstraZeneca”, “Sinopharm”.

**Operaciones:**

crear: crea una vacuna que recibe los atributos nombreVacuna.

sumar: suma la cantidad de vacunas pasada por parámetro.

restar: resta la cantidad de vacunas pasada por parámetro.

verCantidad: Devuelve cantidad de vacunas disponibles.

verAplicadas: devuelve la cantidad de vacunas aplicadas

getNombre: devuelve el nombre de la vacuna

getEdadMinima: devuelve la edad minima aplicable

getTemp: devuelve la temperatura en la que se almacena la vacuna

getFechaIn: devuelve la fecha de ingreso de la vacuna.

editarCant: redefine la cantidad de vacunas a través de un entero recibido como parámetro.

ToString: devuelve un String de los atributos nombre y cantidad.

**Firmas:**

public Vacuna (String nombre, int edadMinima, int temp, Fecha fechaIngreso, int cantidad);

public void sumar(int num);

public void restar(int num);

public int verCantidad();

public int verAplicadas();

public String getNombre();

public int getEdadMinima();

public int getTemp();

public Fecha getFechaIn();

public void editarCant(int cant);

**TAD VacunaConVencimiento (Pfizer** **y Moderna)**

**Atributos:**

diasVencimieto: Int

contadorVencidas int

fVencimiento: Fecha

**IREP:**

ContadorVencidas no puede ser menor que cero.

diasVencimiento no puede ser menor que cero.

**Operaciones:**

crear: crea una vacuna con vencimiento sobreescribiendo el constructor de Vacuna, agregándole el atributo diasVencimiento y fVencimiento e inicializando contadorVencida en 0.

Public voidd controlVencidas(): controla y disminuye el contador de diasVencimieto, si está en cero las suma a contadorVencidas

Public int CantVencidas(): Consulta y devuelve el contadorVencidas

**Firmas:**

public VacunaConVencimiento(String nombre, int edadMinima, int temp, Fecha fechaIngreso, int cantidad, int diasVencimiento)

public void controlVencidas()

public int cantVencidas()

**TAD TURNO**

**Atributos:**

paciente: Paciente

vacuna: Vacuna

fecha: Fecha

estado: Int //paciente sin vacunar=0, paciente vacunado=1.

**Operaciones:**

Crear: recibe un Paciente, el tipo de la Vacuna que debe ser provista y una Fecha (provista por TAD Fecha provisto por la catedra) inicializando el estado en 0(paciente sin vacunar)

concluir: Cambia el estado en 1 dando el Turno por concluido exitosamente

getDNI: devuelve el dni del paciente.

getFecha: devuelve la fecha del turno.

getVacuna: devuelve la vacuna del turno.

PacienteVacunado: devuelve True en caso de que el estado sea igual a 1, caso contrario devuelve False.

getEstado: devuelve el estado del turno.

borrar: “borra” el turno asignándole null a los atributos paciente, vacuna, fecha, y 0 al estado.

**Firmas:**

public Turno(Paciente pac, Vacuna vac, Fecha fecha);

public void concluir();

public int getDNI();

public Fecha getFecha();

public Vacuna getVacuna();

public boolean pacienteVacunado();

public int getEstado();

public void borrar();

**TAD GestionVacunas**

**Atributos:**

vacunas: ArrayList<Vacuna>

vacunasAplicadas: int

paciente: Paciente

vacuna: Vacuna

reporteVacunasVencidas: HashMap<String, Integer>

**Operaciones:**

crear: No recibe parámetros y crea GestionVacunas inicializando todos los atributos de la clase.

IngresarVacunas: recibe como parámetros el nombre de la vacuna, la cantidad y la fecha de ingreso y llama al constructor de Vacuna o VacunaConVencimiento dependiendo el nombre de la vacuna pasada por parametro. En caso de que el nombre de la vacuna no sea válido, lanza una excepción.

IngresarVacunas: recibe el nombre de la vacuna que será ingresada, la cantidad de la misma, y la fecha en que fue ingresada. De acuerdo al nombre ingresado, decide si es vacuna o vacuna con vencimiento, y crea el objeto vacuna asignando la cantidad y fecha de vencimiento en caso de ser necesario.

AplicableA: recibe a un Paciente y una Vacuna. Devuelve True en caso de que el Paciente supere la edad mínima aplicable de la Vacuna, caso contrario devuelve False

vacunasCant: devuelve la cantidad de vacunas que hay en stock.

vacunasDisponibles: recibe como parámetro un nombre de vacuna y devuelve la cantidad de vacunas de dicha vacuna.

controlVencVacunas: Recorre las vacunas (es un array) y revisa cuales son los lotes con vencimiento. En cada uno de estos lotes chequea la fecha de vencimiento y si esta vencida pasa la cantidad a otra variable para cantidad vencida.

getVacuna: recibe un paciente, recorre el arreglo de vacuna y devuelve la primera que encuentra que sea compatible según el Enunciado del trabajo practico

reporteVencidas: devuelve el atributo reporteVacunasVencidas

Vacunas: devuelve un String de las vacunas.

IREP:

vacunasAplicadas debe ser mayor o igual a 0

**Firmas:**

public GestionVacunas():

public void ingresarVacunas( String nombreVacuna, int cantidad, Fecha fechaIngreso);

public void ingresarVacunas(Vacuna vac);

public boolean aplicableA(Paciente p, Vacuna v);

public int vacunasCant();

public int vacunasDisponibles(String nombreVacuna);

public void controlVencVacunas();

public Vacuna getVacuna(Paciente pac);

public Map<String, Integer> reporteVencidas();

public String Vacunas();

**TAD FECHA (dado por la cátedra)**

**DIAGRAMA DE CLASES**

A continuación se grafica con el software StarUML, como se relacionan los distintos TAD generados con anterioridad en este Trabajo Practico:

