Programación II - Trabajo Práctico Integrador
1er. Cuatrimestre 2021
SEGUNDA PARTE

Fecha de presentación: 24 de mayo
Fecha de entrega: Viernes 11 de junio

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 1er 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 cosas como la consistencia interna, por ejemplo, cómo se relaciona el stock de una vacuna con la gente vacunada y las vacunas que llegan.

Por último se debe hacer la implementación, la cual debe cumplir satisfactoriamente el junit otorgado por la cátedra.

Se deben utilizar al menos 2 tecnologías java: (StringBuilder, For range, Iteradores), además de junit en si, en alguna parte del tp.

Se debe implementar(sobreescribir) el toString de las clases principales.Como mínimo se debe mostrar el nombre del centro, la capacidad de vacunación diaria, la cantidad de vacunas disponibles, cantidad de personas en lista de espera, cantidad de turnos asignados y cantidad de vacunas aplicadas hasta el momento.

Se debe implementar(sobreescribir) el equals de vacuna y sus clases derivadas.

Aclaraciones y agregados del problema

Las personas con turno asignado se deben presentar el día del turno para vacunarse, de otra manera “se pierde el turno”. Esto significa que la siguiente vez que se asignen turnos:

- Se debe devolver esa vacuna al stock
- Se debe eliminar a la persona del sistema, de manera que tiene que volver a darse de alta si quiere una vacuna.

Se debe poder conocer la cantidad de vacunas disponibles para cada vacuna.

Las vacunas que se almacenan a -18 grados() tienen una fecha de vencimiento que se cuenta a partir de la fecha en que son ingresadas al centro de vacunación:

- Pfizer dura 30 días a partir de su ingreso
- Moderna dura 60 días a partir de su ingreso.

Como hay vacunas con vencimiento, se solicita poder consultar la cantidad de vacunas vencidas agrupadas por tipo en O(1).

Les damos una clase “**Fecha*”*** de la cátedra que debe ser utilizada para modelar las fechas del TP.

Interfaz

Para la implementación se debe respetar la siguiente interfaz

```
/**
* Constructor.
* recibe el nombre del centro y la capacidad de vacunación diaria.
* Si la capacidad de vacunación no es positiva se debe generar una excepción.
* Si el nombre no está definido, se debe generar una excepción.
*/
public CentroVacunacion(String nombreCentro, int capacidadVacunacionDiaria);

/**
* Solo se pueden ingresar los tipos de vacunas planteados en la 1ra parte.
* Si el nombre de la vacuna no coincidiera con los especificados se debe generar
* una excepción.
* También se genera excepción si la cantidad es negativa.
* La cantidad se debe
* sumar al stock existente, tomando en cuenta las vacunas ya utilizadas.
*/
public void ingresarVacunas(String nombreVacuna, int cantidad, Fecha fechaIngreso);

/**
* total de vacunas disponibles no vencidas sin distinción por tipo.
*/
public int vacunasDisponibles();

/**
* total de vacunas disponibles no vencidas que coincida con el nombre de
* vacuna especificado.
*/
public int vacunasDisponibles(String nombreVacuna);

/**
* Se inscribe una persona en lista de espera.
* Si la persona ya se encuentra inscripta o es menor de 18 años, se debe
* generar una excepción.
* Si la persona ya fue vacunada, también debe generar una excepción.
*/
public void inscribirPersona(int dni, Fecha nacimiento,
                     boolean tienePadecimientos, boolean esEmpleadoSalud);

/**
* Devuelve una lista con los DNI de todos los inscriptos que no se vacunaron
* y que no tienen turno asignado.
* Si no quedan inscriptos sin vacunas debe devolver una lista vacía.
*/
public List<Integer> listaDeEspera();

/**
* Primero se verifica si hay turnos vencidos. En caso de haber turnos
* vencidos, la persona que no asistió al turno debe ser borrada del sistema
* y la vacuna reservada debe volver a estar disponible.
*
```

```
* Segundo, se deben verificar si hay vacunas vencidas y quitarlas del sistema.
*
* Por último, se procede a asignar los turnos a partir de la fecha inicial
* recibida según lo especificado en la 1ra parte.
* Cada vez que se registra un nuevo turno, la vacuna destinada a esa persona
* dejará de estar disponible. Dado que estará reservada para ser aplicada
* el día del turno.
*
*
*/
public void generarTurnos(Fecha fechaInicial);

/**
* Devuelve una lista con los dni de las personas que tienen turno asignado
* para la fecha pasada por parámetro.
* Si no hay turnos asignados para ese día, se debe devolver una lista vacía.
* La cantidad de turnos no puede exceder la capacidad por día de la ungs.
*/
public List<Integer> turnosConFecha(Fecha fecha);

/**
* Dado el DNI de la persona y la fecha de vacunación
* se valida que esté inscripto y que tenga turno para ese dia.
*   - Si tiene turno y está inscripto se debe registrar la persona como
*     vacunada y la vacuna se quita del depósito.
*   - Si no está inscripto o no tiene turno ese día, se genera una Excepcion.
*/
public void vacunarInscripto(int dni, Fecha fechaVacunacion);

/**
* Devuelve un Diccionario donde
*   - la clave es el dni de las personas vacunadas
*   - Y, el valor es el nombre de la vacuna aplicada.
*/
public Map<Integer, String> reporteVacunacion();

/**
* Devuelve en O(1) un Diccionario:
* - clave: nombre de la vacuna
* - valor: cantidad de vacunas vencidas conocidas hasta el momento.
*/
public Map<String, Integer> reporteVacunasVencidas();
```

Apéndice I: Codigo Cliente

```
package centroVacunacion;

public class Principal {

	public static void main(String[] args) {
		Fecha fTurnos = new Fecha(15, 7, 2021);
		CentroVacunacion centro = new CentroVacunacion("UNGS", 5);

		System.out.println("------------ Creacion -------------");
		System.out.println(centro);
		System.out.println("-----------------------------------");
		System.out.println();

		centro.ingresarVacunas("Moderna", 10, new Fecha(15,5,2021));
		centro.ingresarVacunas("Pfizer", 10, new Fecha(15,5,2021));

		centro.inscribirPersona(34701000, new Fecha(1, 5, 1989), false, false);
		centro.inscribirPersona(29959000, new Fecha(20, 11, 1982), false, true);
		centro.inscribirPersona(24780201, new Fecha(1, 6, 1972), true, false);
		centro.inscribirPersona(29223000, new Fecha(2, 5, 1982), false, true);
		centro.inscribirPersona(13000000, new Fecha(1, 5, 1958), true, false);
		centro.inscribirPersona(13000050, new Fecha(20, 6, 1958), false, true);
		centro.generarTurnos(fTurnos);

		System.out.println("-------------- Turnos -------------");
		System.out.println(centro.turnosConFecha(fTurnos));
		System.out.println("-----------------------------------");
		System.out.println();
		centro.vacunarInscripto(24780201, fTurnos);
		centro.vacunarInscripto(13000000, fTurnos);

		System.out.println("------------- Centro --------------");
		System.out.println(centro);
		System.out.println("-----------------------------------");

	}
}
```

Apéndice II: JUnit

```
package centroVacunacion;

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;

public class TestCentroVacunacion {
CentroVacunacion centro;

@Before
public void setUp() throws Exception {
Fecha.setFechaHoy();
centro = new CentroVacunacion("UNGS", 5);
centro.ingresarVacunas("Sputnik", 10,new Fecha(20,3,2021));
centro.ingresarVacunas("AstraZeneca", 10,new Fecha(20,3,2021));

centro.inscribirPersona(34701000, new Fecha(1, 5, 1989), false, false);
centro.inscribirPersona(29959000, new Fecha(20, 11, 1982), false, true);
centro.inscribirPersona(24780201, new Fecha(1, 6, 1972), true, false);
centro.inscribirPersona(29223000, new Fecha(2, 5, 1982), false, true);
centro.inscribirPersona(13000000, new Fecha(1, 5, 1958), true, false);
centro.inscribirPersona(13000050, new Fecha(20, 6, 1958), false, true);
centro.inscribirPersona(14000000, new Fecha(1, 1, 1961), false, false);
centro.inscribirPersona(14005000, new Fecha(20, 12, 1961), true, false);
}

@Test
public void testIngresarVacunas() {
assertEquals(20, centro.vacunasDisponibles());

centro.ingresarVacunas("Pfizer", 10,new Fecha(20,3,2021));
centro.ingresarVacunas("Moderna", 10,new Fecha(20,3,2021));
centro.ingresarVacunas("Sinopharm", 10,new Fecha(20,3,2021));

assertEquals(50, centro.vacunasDisponibles());
}

@Test
public void testInscripcion() {
assertEquals(8, centro.listaDeEspera().size());

centro.inscribirPersona(34780201, new Fecha(1, 7, 1989), false, false);
centro.inscribirPersona(29223959, new Fecha(10, 11, 1982), false, true);

assertEquals(10, centro.listaDeEspera().size());
}

@Test
public void testGenerarYConsultarTurnos() {
Fecha fechaInicial = new Fecha(2, 7, 2021);
Fecha fechaSiguiente = new Fecha(3, 7, 2021);
```

```
Fecha fechaAnteriorSinTurnos = new Fecha(1, 7, 2021);
Fecha fechaPosteriorSinTurnos = new Fecha(4, 7, 2021);

assertEquals(8, centro.listaDeEspera().size());
assertEquals(20, centro.vacunasDisponibles());
centro.generarTurnos(fechaInicial);
assertEquals(0, centro.listaDeEspera().size());
assertEquals(12, centro.vacunasDisponibles());

// son 8 anotados y la capacidad diaria es 5 personas.
assertEquals(5, centro.turnosConFecha(fechaInicial).size());
assertEquals(3, centro.turnosConFecha(fechaSiguiente).size());
assertEquals(0, centro.turnosConFecha(fechaAnteriorSinTurnos).size());
assertEquals(0, centro.turnosConFecha(fechaPosteriorSinTurnos).size());
}

@Test
public void testGenerarTurnosYRegistrarVacunacion() {
int dniAVacunar=29959000;
Fecha fecha = new Fecha(30,6,2021);
assertEquals(20, centro.vacunasDisponibles());
assertTrue(centro.listaDeEspera().contains(dniAVacunar));
assertFalse(centro.reporteVacunacion().containsKey(dniAVacunar));

centro.generarTurnos(fecha);
assertEquals(12, centro.vacunasDisponibles());
assertFalse(centro.listaDeEspera().contains(dniAVacunar));
assertFalse(centro.reporteVacunacion().keySet().contains(dniAVacunar));

centro.vacunarInscripto(dniAVacunar,fecha);

assertTrue(centro.reporteVacunacion().keySet().contains(dniAVacunar));
// Simulo que pasó la fecha del turno y reviso que los turnos no
// cumplidos devuelvan las vacunas al STOCK y no quede gente en
// lista de espera.
Fecha.setFechaHoy(2, 7, 2021);

centro.generarTurnos(new Fecha(5,7,2021));
assertEquals(19, centro.vacunasDisponibles());
assertTrue(centro.listaDeEspera().isEmpty());
}
@Test
public void testReporteVacunasVencidas() {
CentroVacunacion centroConVacunasVencidas = new CentroVacunacion("UNGS 2", 5);
Fecha.setFechaHoy();
centroConVacunasVencidas.ingresarVacunas("Pfizer", 10,new Fecha(20,3,2021));
centroConVacunasVencidas.ingresarVacunas("Pfizer", 10,new Fecha(20,4,2021));

assertEquals(20, centroConVacunasVencidas.vacunasDisponibles("Pfizer"));
// Simulo que hoy es el 19 de mayo
Fecha.setFechaHoy(19,5,2021);
centroConVacunasVencidas.generarTurnos(new Fecha(20,5,2021));
assertEquals(10, centroConVacunasVencidas.vacunasDisponibles("Pfizer"));
assertEquals(10, centroConVacunasVencidas.reporteVacunasVencidas()
                 .get("Pfizer").intValue());
```

```
}

/*****************************************************************************/
/********************* Casos que deben fallar ****************************/
/*****************************************************************************/

@Test
public void testIngresarVacunasConCantidadInvalida() {
try {
centro.ingresarVacunas("AstraZeneca", 0, new Fecha(20,3,2021));
fail("Permitió ingresar una vacuna con cantidad 0");
} catch (RuntimeException e) { }
try {
centro.ingresarVacunas("Moderna", -10, new Fecha(20,3,2021));
fail("Permitió ingresar una vacuna con cantidad negativa");
} catch (RuntimeException e) { }
}

@Test(expected = RuntimeException.class)
public void testGenerarTurnosParaUnaFechaPasada() {
centro.generarTurnos(new Fecha(10,05,2021));
}

@Test(expected = RuntimeException.class)
public void testVacunarPersonaNoRegistrada() {
centro.vacunarInscripto(17000000,new Fecha(31,12,2021));
}

@Test(expected = RuntimeException.class)
public void testVacunarPersonaConTurnoParaOtraFecha() {
int dniAVacunar=29959000;
Fecha fecha = new Fecha(30,6,2021);
Fecha fechaIncorrecta = new Fecha(29,6,2021);
centro.generarTurnos(fecha);
centro.vacunarInscripto(dniAVacunar,fechaIncorrecta);
}
}
```