Programación II

Trabajo práctico Final

Autores:

Elías Espinillo 40.663.777

Federico Farias 36.495.959

Profesores:

Daniel Bertaccini

Javier Escobar

Contenidos

1 Diseño, especificación e IREP (con correcciones)

Hemos arreglado el diseño con las correcciones enviadas por los profesores y extendiendo los TADs con métodos privados que nos ayuden a resolver los problemas de una manera más sencilla

- **Centro de Vacunación:**

  Invariante de Representación:

  La capacidadDiaria debe ser un entero mayor a 0. Cada valor para cada clave en vacunados solo puede ser : Sputnik , Moderna , AstraZeneca , Sinopharm , Pfizer. Cada persona inscripta debe ser mayor de 18 años, el DNI no puede ser menor a 0.000.001 ni mayor a 99.999.999 y no puede haber sido vacunado en el centro. El nombre del centro de vacunación no puede estar vacío. La cantidad de vacunas ingresadas no puede ser menor a 0. No se pueden generar turnos para fechas anteriores al día de la fecha.

  Atributos:

  - Capacidad del centro de vacunación: Integer
  - Nombre del centro de vacunación: String
  - Diccionario de inscriptos: HashMap<Integer, Persona>
  - Diccionario de turnos: LinkedHashMap<Integer, Persona>
  - Diccionario de personas vacunadas: HashMap<Integer, String>
  - Lista de prioridad de personas: LinkedList<Persona>
  - Heladera para vacunas: HeladeraVacunas

Operaciones:

- Constructor del centro de vacunación. 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 void ingresarVacunas(String nombreVacuna, int cantidad, Fecha fechaIngreso):

  Se ingresa una vacuna con su nombre, la cantidad de vacunas y la fecha con la que ingresa al centro.
- public int vacunasDisponibles():

  Muestra el total de vacunas disponibles que no estén vencidas del centro.
- public int vacunasDisponibles(String nombreVacuna):

  Muestra el total de cada vacuna pasada como parámetro que no esté vencida del centro.
- public void inscribirPersona(int dni, Fecha nacimiento, boolean comorbilidad, boolean trabajadorDeSalud):

  Inscribe una persona pasando como parámetros el dni, su edad, si es trabajador de salud y si tiene comorbilidades.
- private void definirPrioridad():

  Método que se utiliza en inscribirPersona() para definir la prioridad de la misma.
- public List<Integer> listaDeEspera():

  Devuelve una lista con los DNI de los inscriptos sin vacunar y que no tienen turno asignado. Si no quedan inscriptos sin vacunas se devuelve la lista vacía.
- public void generarTurnos(Fecha fechaInicial):

  El método verifica si hay turnos vencidos, en caso de que los hubiera se borra a la persona del sistema. Luego verifica si hay vacunas vencidas (Moderna y Pfizer) y las elimina del sistema. Por último se asignan los turnos con la fecha pasada como parámetro.

- private void removerPorfechaInvalida():

  Primer paso para poder generarTurno(). Verificamos si hay fechas validas caso contrario la eliminamos y devolvemos la vacuna que había sido asignada.

- private void inscriptosOrdenados():

  Segundo paso para generarTurno(). Ordena a los inscriptos según su prioridad.

- private void asignarTurnos(Fecha fechaInicial):

  Tercer paso para generarTurno(). Asignamos el turno para cada inscripto chequeando las fechas.

- private void moverConTurnoAsignado():

  Cuarto paso para generarTurno(). Finalmente movemos a la persona de inscriptos al diccionario de turnos.

- private Fecha chequearFecha(Fecha fecha):

  Método que se utiliza en asignarTurno() para comparar la fecha del parámetro con la fecha de hoy.

- private int cantidadDeTurnosPorDia(Fecha fecha):

  Método que utiliza chequearFecha() donde recorre las fechas del diccionario de turnos.

- public List<Integer> turnosConFecha(Fecha fecha):

  Retorna una lista con los DNI de las personas que tienen turno asignado para la fecha pasada como parámetro. Si no hay turnos asignados para ese día devuelve la lista vacía.

- public void vacunarInscripto(int dni, Fecha fechaVacunacion):

  Con el DNI y la fecha de vacunación se valida que la persona esté inscripta y tenga turno para ese día. Si se confirma la persona pasa a tener el estado de vacunado y se elimina a la vacuna del depósito. Caso contrario se genera una excepción, ya sea porque no está inscripto o porque no tiene turno ese día.

- public Map<Integer, String> reporteVacunacion():

  Devuelve un diccionario con el DNI como clave y la vacuna aplicada como valor.

- public Map<String, Integer> reporteVacunasVencidas():

  Retorna un diccionario con el nombre de la vacuna como clave y la cantidad de vacunas vencidas como valor.

- public String toString():

  Con un StringBuilder le damos forma al toString para que muestre los datos necesarios del centro de vacunación cuando se haga un print.

- **HeladeraVacunas:**

Invariante de Representación:

Las claves en la colección vacunasVencidas solo pueden ser : Sputnik , Moderna , AstraZeneca , Sinopharm , Pfizer. La cantidad de vacunas ingresadas no puede ser menor a 0.

Atributos:

- Código de vacuna: Integer
- Diccionario de vacunas: HashMap<Integer, Vacunas>
- Diccionario de vacunas vencidas: HashMap<String, Integer>

Operaciones:

- Constructor para crear una heladera.
- public void ingresarVacunas(String nombre, int cant, Fecha fechaDeEntrada):
  Ingresa la vacuna a la heladera con su nombre, cantidad y fecha de entrada, además le pasa un código para diferenciarla.
- public int vacunasDisponibles(String nombre):
  Muestra la cantidad de vacunas disponibles por tipo en la heladera y verificamos que no esté reservada para una persona.
- public int vacunasDisponibles():
  Indica la cantidad total de vacunas disponibles en la heladera.
- public String asignarVacunaDisponibles(int edad):
  Se ingresa la edad, verifica si hay vacunas disponibles por tipo y si se cumple se asigna la vacuna con el método asignarVacuna().
- private void asignarVacuna(String nombre):
  Setea la vacuna del stock como asignada.

- public void aplicarVacuna(String nombre):

  Setea la vacuna del stock como aplicada.

- public void quitarVacunaAplicada(String nombre):

  Elimina la vacuna aplicada a la persona.

- public void devolverVacuna(String nombre):

  Devuelve la vacuna de la persona que no asistió al turno a la heladera

- public void quitarVacunaVencida():

  Si la vacuna estuviera vencida la elimina

- public void moverVacunasVencidas():

  Mueve la vacuna vencida a un diccionario que reporta vacunas vencidas.

- Map<String, Integer> reporteVacunasVencidas():

  Brinda un reporte de la situación de las vacunas vencidas.

- public String toString():

  Con un StringBuilder le damos forma al toString para que muestre los datos necesarios de las vacunas en general.

- **Persona:**

Invariante de Representación:

El número de DNI no puede ser menor a 0.000.001 ni mayor a 99.999.999. La diferencia de la fecha de nacimiento y la fecha de hoy no puede ser menor a 18.

Atributos:

- Número de DNI: Integer
- Fecha de nacimiento: Fecha
- Fecha de turno: Fecha
- Es trabajador de salud: Boolean
- Tiene comorbilidades: Boolean
- Prioridad de la persona: Integer
- Vacuna asignada a la persona: String

Operaciones:

- Constructor para crear una Persona. Recibe como parámetros el DNI, la fecha de nacimiento, si es trabajador de salud y si tiene comorbilidades.
- Getters y setters de la clase.
- public int edad():

  Calculamos la edad en base a la fecha de nacimiento y la diferencia con la fecha actual.

- **Vacunas (Super clase abstracta):**

  Invariante de Representación:

  La fechaDeEntrada tiene que ser igual a la fecha de hoy y como las clases Astra, Moderna, Pfizer, Sino, Sputnik heredan de esta, el IREP es siempre true.

  Atributos:

  - Fecha de entrada: Fecha
  - Nombre de la vacuna: String
  - Vencida: Boolean
  - Reservada para una persona: Boolean
  - Ya aplicada a una persona: Boolean

  Operaciones:

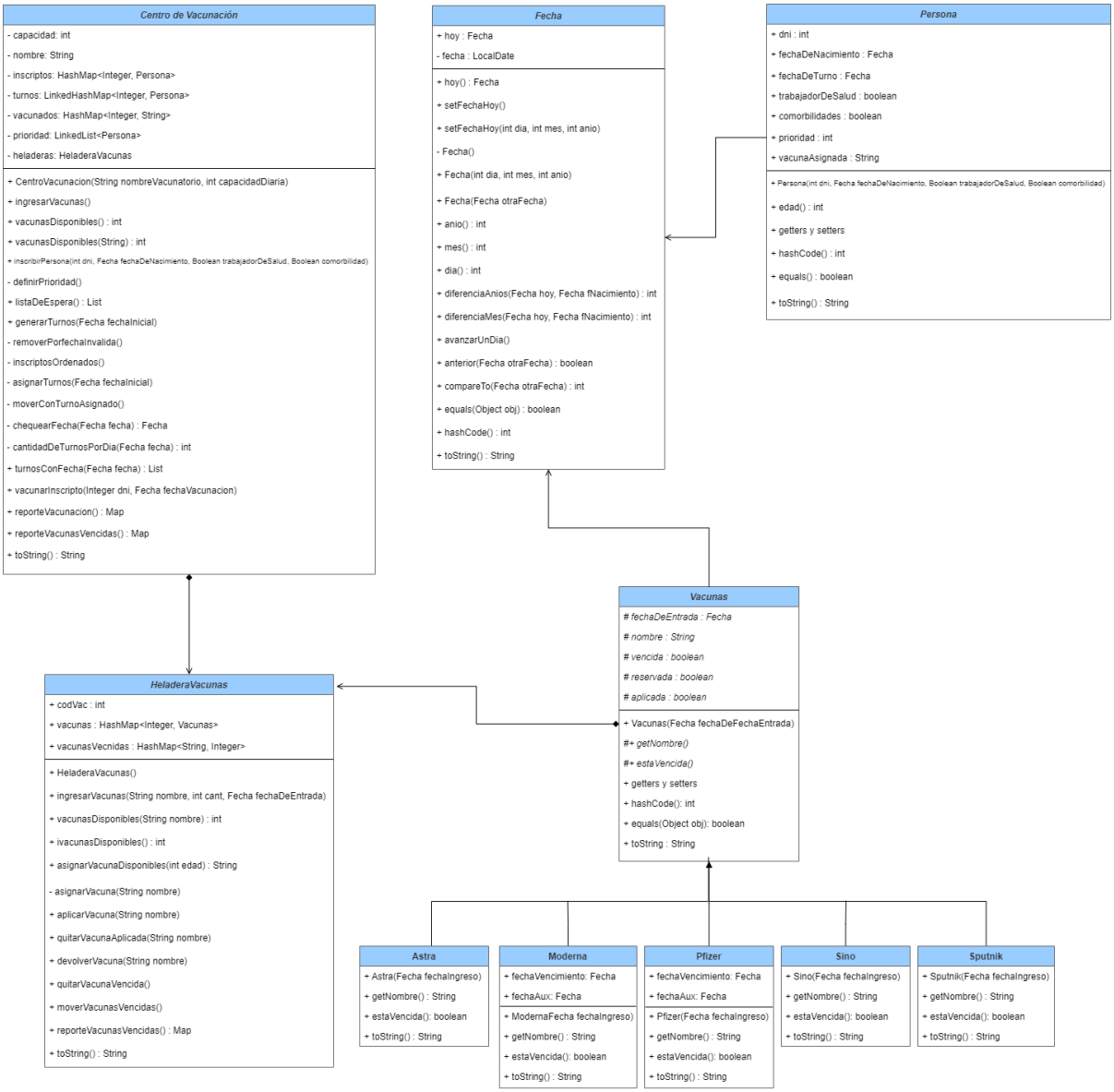
  - Constructor de vacunas. Recibe como parámetro la fecha de entrada.
  - abstract public String getNombre()

    Lo implementan las clases heredadas.
  - abstract public boolean estaVencida()

    Lo implementan las clases heredadas.
  - Getters y setters de la clase.

- **Sputnik, Sinopharm y AstraZeneca (clases heredadas):**
  Heredan de la clase Vacunas los atributos, el método abstracto getNombre() que devuelve el nombre de cada vacuna y el método estaVencida() que retorna false porque no se vencen.
- **Pfizer y Moderna (clases heredadas):**
  Heredan de la clase Vacunas los atributos, el método abstracto getNombre() que devuelve el nombre de cada vacuna y el método estaVencida() que verifica la fechaDeEntrada de la vacuna para ver si esta vencida.

3 Diagrama de Clases



4 Conclusiones

Este trabajo se realizó con el objetivo de diseñar, implementar y simular un centro de vacunación para ayudar a mejorar el sistema de vacunación nacional.

Para lograr el objetivo se consultó a los profesores para la primera parte realizar la especificación y diseño del trabajo. Luego de haber recibido las correcciones se comenzó con la implementación del mismo para proseguir con la segunda parte. A medida que se implementaba fueron surgiendo problemas los cuales solucionamos dividiendo en métodos más sencillos para poder trabajar en equipo. Se consiguió aplicar los conceptos de herencia, polimorfismo y TADs. También se pudieron aplicar tecnologías JAVA como Iterator, For Each y StringBuilider.

La realización del trabajo se desarrolló de la mejor manera posible ya que el equipo se comunicó todo el tiempo que sea necesario ya sea por chat entre todos, inclusive los profesores, y también por reuniones virtuales para poder llevar al día las actualizaciones que pudieran surgir en cuanto al trabajo práctico.

Finalmente, se llegó a la conclusión de que el objetivo del trabajo se cumplió pudiendo hacer el centro de vacunación acorde al test otorgado por la cátedra.