Universidad Técnica Federico Santa María

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

ELO329 - DISEÑO Y PROGRAMACIÓN ORIENTADA A OBJETOS

Tarea 1

Simulando la evolución de una pandemia

Estudiante	ROL
Tamara Carrera	201621010-5
Felipe Contreras	201621002-4
Ricardo Mardones	201621036-9
Nicolás Veneros	201621024-5

Paralelo: 2

Profesor

Patricio Olivares R.

Ayudante

Luis Torres G.

Fecha : 09/05/2021

Índice

Índice de figuras

1. Documentación

1.1. Diagrama UML

La figura 1 representa el diagrama UML de la etapa 4, el cual incluye las 5 clases utilizadas, 4 clases que se implementaron a partir del template y la clase vacunatorio creada para esta última etapa. A partir de este diagrama se puede obtener la relación entre las clases.

De la figura se observa que Stage4 tienen una relación de dependencia (línea discontinuas) con la clase Simulador y Comuna, esto ocurre porque en Stage4 se crea una instancia para cada una de estas clases. También la clase Simulador tiene una relación de dependencia con las clases Individuo y Vacunatorio al crear instancias de ellas.

Por otro lado, Simulador tiene una relación de dependencia con Comuna, Vacunatorio con Comuna, Comuna con Individuo y viceversa, estas relaciones se deben a que la primera clase tiene como atributo un objeto de la segunda (respectivamente en las relaciones mencionadas) y por ende le permite acceder a los métodos respectivos.

Otra de las relaciones presentes en el diagrama es la de asociación (líneas continuas), específicamente de composición (líneas continuas con rombos negros) entre las clases Simulador y Vacunatorio, Simulador y Comuna, Comuna e Individuo y viceversa. Esta relación significa que **la segunda clase solo existe si un objeto de la primera es creado**.

La relación entre Simulador y Vacunatorio junto con la de Comuna e Individuo es de $(1-*)$, lo cual significa que 0 o $+$ instancias de las clases Vacunatorio e Individuo dependen de la creación de un objeto Simulador y un objeto Comuna, respectivamente. En cambio, entre Simulador y Comuna junto con Individuo y Comuna la relación es de $(1-1)$, lo que implica que la existencia de una instancia de la clase Comuna depende de la existencia de la creación de un objeto Individuo y de un objeto Simulador.

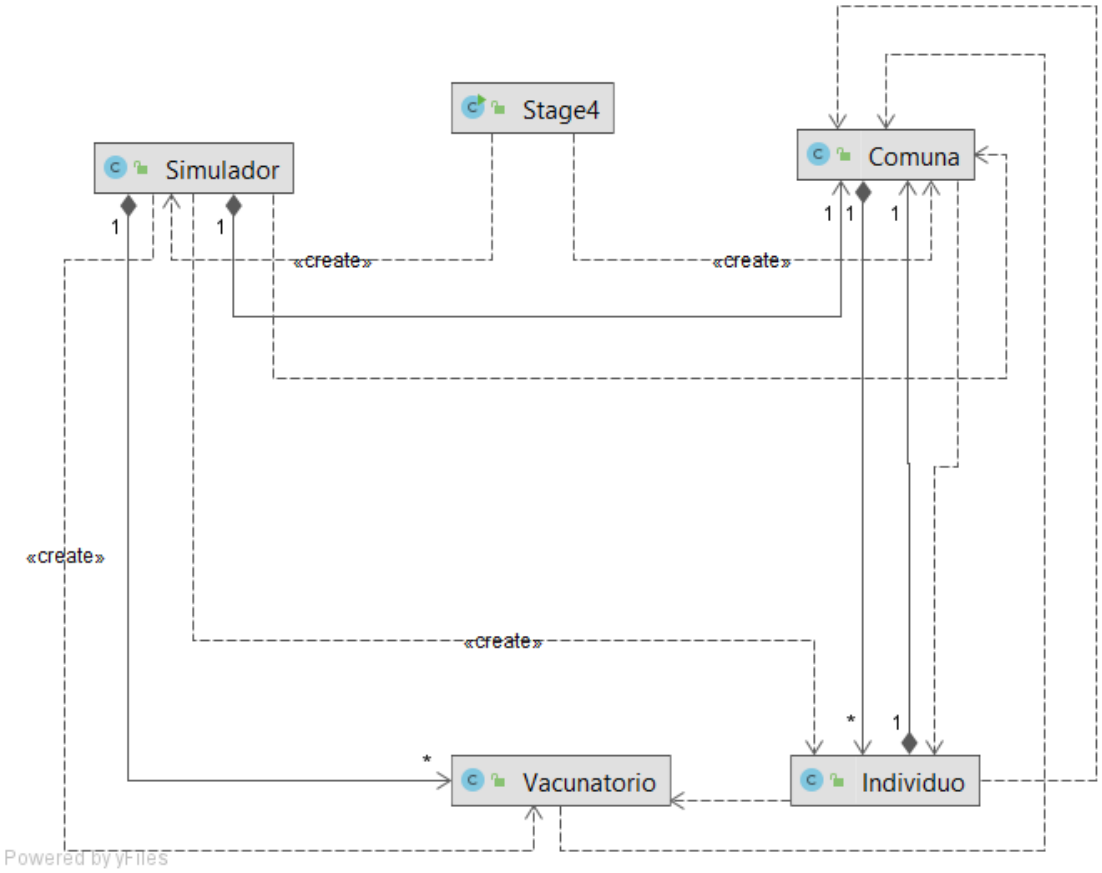

Figura 1: Diagrama UML de Stage4

1.2. Explicación breve

Para el desarrollo de la actividad, se utiliza como base las clases dispuestas por los profesores de la asignatura en el repositorio del ramo. Se ejecuta de manera incremental lo pedido en la tarea.

Esto se logra aplicando las siguientes clases:

- **Individuo**: Posee como atributos y métodos los observados en la figura 2. Los métodos importantes para el desarrollo de la actividad corresponden a: *computeNextState(double)*, el cual recibe como argumento el tiempo *delta_t* en el que se está computando el estado, aquí se mueve al individuo.
  El siguiente método de importancia en la clase es *getSusNewState(Individuo, double, double)* que retorna un valor *boolean* el cual indica si el individuo susceptible cambia su estado a infectado. Cada vez que se compara con otro individuo, calcula la distancia entre ambos individuos y verifica si están dentro de un radio d de infección para calcular su probabilidad.
  verifyVac(ArrayList <Vacunatorio>, double) recibe un *ArrayList* de Vacunatorios y el tamaño de éstos. Luego verifica si el individuo está dentro del área de vacunación de alguno de los vacunatorios, de ser así, deja de buscar en ellos y cambia su estado stateVac a *true* y actualiza los demás estados a *false*. En caso de recorrer toda la lista y no encontrar vacunatorio en su posición actual, deja stateVac como *false* y los demás estados como estaban antes.
  getInfNewState(double, double) recibe como argumentos delta_t e I_time que corresponden al tiempo de simulación y tiempo requerido para pasar de infectado a recuperado respectivamente. Simplemente accede al atributo que indica cuanto tiempo lleva infectado, le suma el tiempo de simulación y si este es mayor a I_time, cambia su estado *new_stateRec = true*.
  Finalmente, *updateState()* actualiza los estados y posiciones calculados a los almacenados para el individuo.

- **Comuna**: Posee como atributos y métodos los observados en la figura 2. En esta clase los métodos más importantes son *setPerson(ArrayList<Individuo >)* que añade la lista de individuos entrega como atributo a la comuna, *computeNextState(double)* que recibe como argumento un delta_t y calcula para cada individuo su nueva posición según el método de mismo nombre descrito en la clase Individuo y updateState(), que actualiza los estados de cada individuo de la comuna.

- **Vacunatorio**: Es una clase que solo crea los vacunatorios cuando se le indique, recibiendo como atributo una comuna y definiendo como origen un punto aleatorio entre 0 y la diferencia entre el ancho/largo de la comuna con el tamaño del lado del vacunatorio.

- **Simulador**: En esta clase, se definen atributos propios de la simulación, los cuales se leen en Stage4 desde el documento configurationFile.txt y son guardados en ella para posterior utilización. Algunos métodos utilizados son: *populate_comuna()* que inicializa los individuos de la comuna generando además la cantidad requerida de gente con mascarilla, *update_comuna()* que calcula los estados de los individuos de las personas utilizando los métodos descritos en la clase Individuo calculando

la probabilidad de infección en los distintos casos según descrito en la tarea y *simulate()*, método principal de la simulación que inicializa el modelo utilizando los métodos descritos anteriormente.

- **Stage4**: Esta clase simplemente crea un nuevo objeto de la clase Comuna, un nuevo objeto de la clase Simulador, los inicializa con los datos necesarios según descripción de la tarea, y hace correr la simulación.

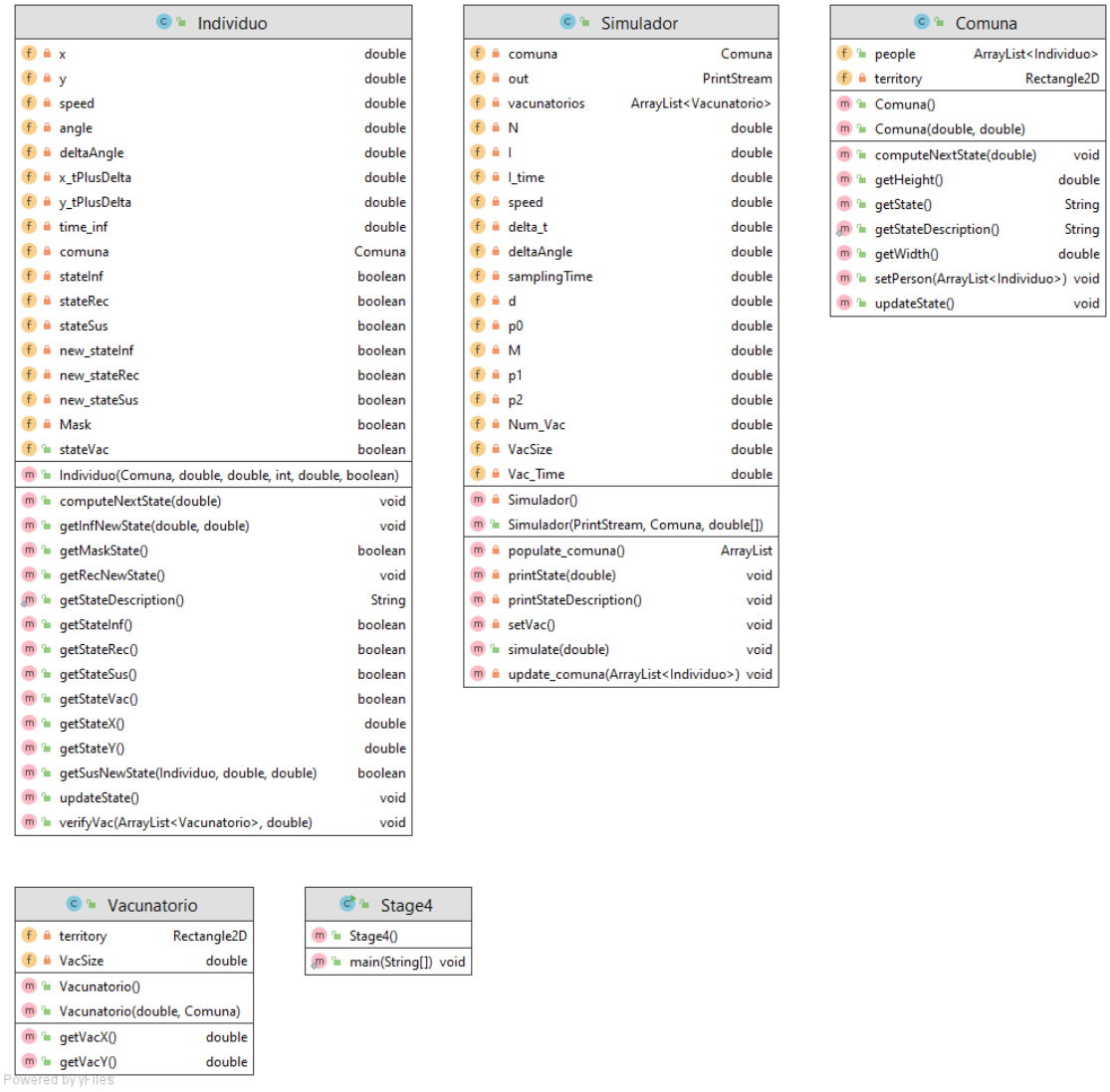

Figura 2: Diagrama de clases, con sus respectivos atributos y métodos para la Etapa 4.

Con lo descrito anteriormente, se obtiene para un configurationFile.txt como el que sigue a continuación:
360 100 30 40
50 100
5 1 0.4
2 0.5 0.5 0.3 0.1
2 10 60
el gráfico de la figura 3.

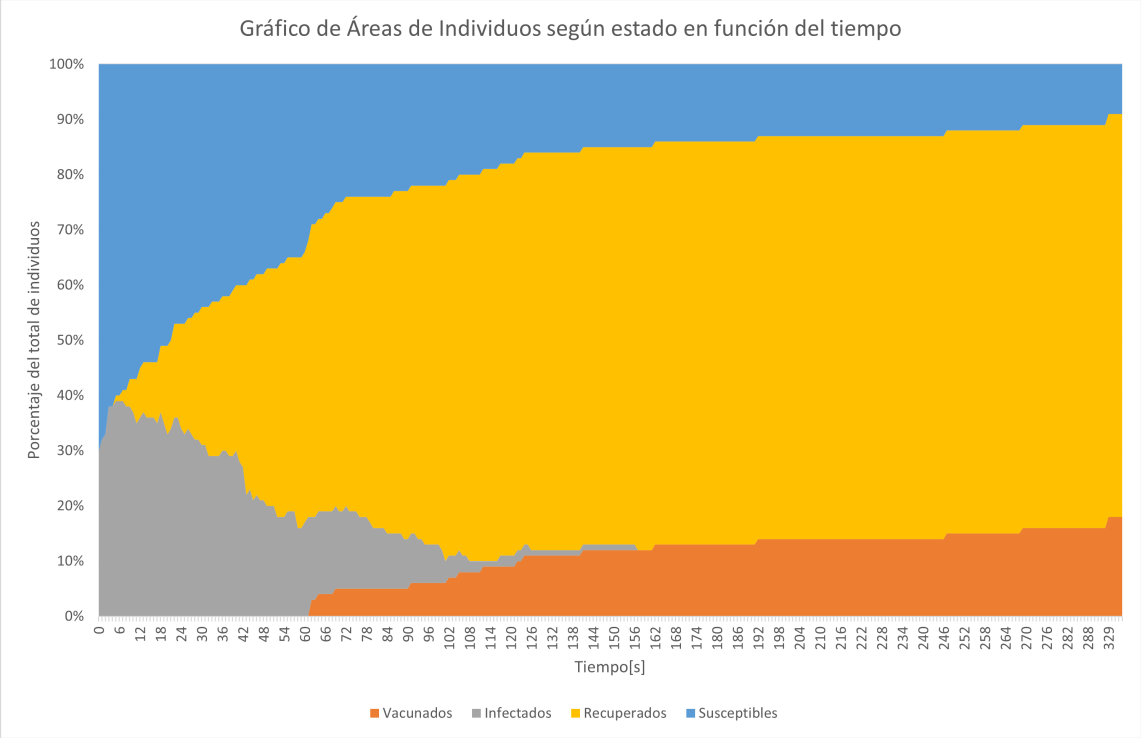

Figura 3: Gráfico de áreas que muestra resultado de simulación para los parámetros dados.

Como se puede ver en el gráfico anterior, se observa que inicialmente los individuos que se encuentran en la comuna se pueden clasificar solo en dos estados, estos son *infectados* y *susceptibles* (áreas gris y celeste respectivamente). Cerca de los $12\ s$ ya comienza la transición de algunos de los individuos que estaban infectados al estado *recuperados* (área amarilla), esta cantidad comienza a aumentar en la medida que los individuos que mantiene su estado anterior disminuyen.
Tal como se definió, el área naranja que representa a los individuos vacunados aparece cerca de los $60\ s$ de simulación, entregado como parámetro $<$`VacTime[s]` $>$ en el archivo `configurationFile.txt`, la cantidad de individuos vacunados aumenta lentamente debido al bajo numero de vacunatorios ingresados en la simulación probada.
Se puede notar como aproximadamente en el segundo 156 de la simulación, el porcentaje de individuos clasificados en estado de recuperados se estaciona mientras que el porcentaje de infectados tiende a cero.

1.3. Dificultades

Las dificultades más grandes que se encontraron durante el desarrollo de la tarea fueron, en primer lugar, lograr que los individuos 'rebotaran' en las paredes. En un principio se pensó que si al calcular la nueva posición esta estuviese por fuera de la comuna entonces se debía recalcular la nueva posición acorde al ángulo de rebote y distancia hasta el borde. Sin embargo, dadas las especificaciones de la tarea se decide simplificar el ejercicio, y en caso de moverse fuera de la comuna, la nueva posición se define como el punto de choque con el borde. Luego el problema fue asegurar que el ángulo no mantuviese al individuo en la pared, sin embargo, se solucionó al asignar como ángulo del siguiente estado el negativo del ángulo pasado.

En segundo lugar, al llegar a la etapa 4 se consideró generar los centros de vacunación con posiciones que no permitieran superposición, dado que en base al contexto del ejercicio esto sería parte de una solución más realista. No obstante dado que esto incide en la eficiencia del código y no tiene implicancia en el cumplimiento del objetivo de la tarea, se optó por omitirlo.

En ultimo lugar, dado que para las primeras 3 etapas se decidió trabajar con valores booleanos que representaran los 3 estados posibles de las personas (susceptible, infectado, recuperado), ya se había logrado un sistema de métodos donde siempre se mantuviese solo 1 de los valores en $true$ para cada individuo. Consecuencia de ello, al tener que agregar un nuevo estado por sobre lo que ya se había generado, fue difícil de implementar sin modificar grandes partes de código o caer en sentencias redundantes.