Diseño y Programación Orientados a Objetos
1er. Semestre 2021
Tarea 1: Simulando la Evolución de una Pandemia

Lea detenidamente la tarea. **Si algo no lo entiende, consulte. Si es preciso, se incorporarán aclaraciones al final.**

1.- Objetivos de la tarea

- Modelar objetos reales como objetos de software.
- Ejercitar la creación y extensión de clases dadas para satisfacer nuevos requerimientos.
- Reconocer clases y relaciones entre ellas en códigos fuentes Java.
- Ejercitar la compilación y ejecución de programas en lenguaje Java desde una consola de comandos.
- Ejercitar la configuración de un ambiente de trabajo para desarrollar aplicaciones en lenguaje Java, se pide trabajar con un IDE. Si bien IntelliJ es el sugerido, su grupo puede trabajar con otro IDE.
- Ejercitar la entrada y salida de datos.
- Manejar proyectos vía GIT.
- Conocer el formato .csv y su importación hacia una planilla electrónica.
- Ejercitar la preparación y entrega de resultados de software (creación de makefiles, readme, documentación).
- Familiarización con desarrollos "iterativos" e "incrementales".

1.- Descripción General

Esta tarea busca practicar la orientación a objeto en una situación inspirada en una pandemia tipo Covid-19. De ninguna manera la intención es obtener resultados sobre cómo debería evolucionar una pandemia real.

Se considera un número dado de “N” individuos dispersos que se mueven aleatoriamente y de manera confinada en una región plana rectangular, que llamaremos **comuna**, durante un tiempo “T” (tiempo de simulación). De los N individuos iniciales, “S” es el número de individuos susceptibles a infectarse e “I” es el número de individuos ya infectados. En este ejercicio, los individuos infectados pasan a estado “R”, de recuperado, luego de I_time segundos. Los parámetros N, T, I, e I_time son ingresados vía un archivo de entrada. Individuos recuperados no son susceptibles de ser infectados nuevamente.

La comuna es rectangular de dimensiones (width, length), ambos parámetros ingresados desde el archivo de entrada.
El movimiento aleatorio de los individuos es simulado con movimientos de rapidez constante “Speed”. La dirección θ de la velocidad varían cada Δt a a un valor aleatorio tomado de entre θ-Δθ y θ+Δθ. “Speed”, Δt y Δθ son parámetros de la simulación ingresados vía archivo, y θ inicial un valor aleatorio para cada individuo. Cuando un individuo tiende a salir de la comuna, éste cambia su dirección como si estuvieran rebotando en el borde de ésta.

Cuando un individuo susceptible de infectarse se acerca a una distancia inferior a d [m] de uno infectado, existe una probabilidad P de que adquiera la infección por cada segundo en contacto cercano. Esta probabilidad de contagio depende de si ninguno (P=p0), sólo uno (P=p1) o ambos (P=p2) usan mascarilla. p0, p1 y p2 son parámetros de la simulación también ingresados por archivo. El parámetro M indica la fracción de individuos de cada tipo que usan mascarilla. Por ejemplo, si M=0,2, significa que 1/5 de los individuos infectados usan mascarilla y 1/5 de los susceptibles de infestarse también la usan. M es otro parámetro de la simulación ingresado por archivo.

Finalmente, en la comuna se activan NumVac vacunatorios a los VacTime segundos de iniciada la simulación. Los vacunatorios son representados como zonas cuadradas fijas, de lado VacSize, y

ubicados aleatoriamente en la comuna. Cuando un individuo susceptible de infectarse ingresa al área de un vacunatorio, éste es vacunado, pasa a estado V y deja de ser susceptible de infectarse. Individuos infectados o recuperados no son vacunados en esta simulación.

El formato para el archivo de entrada -que es pasado como un argumento de la ejecución del programa- es rígido:
<T [s] > <N> <I> <I_time [s] >
<width [m]> <length [m] >
<Speed [m/s] > <Δt [s] > <Δθ [rad] >
<d [m] > <M> <p0> <p1> <p2>
<NumVac> <VacSize> <VacTime [s] >

Por ejemplo este archivo puede tener valores como (es posible que haya valores más interesantes de considerar):
360 1 0 5
100 100
1.4 0.2 0.4
2 0.5 0.3 0.2 0.1
2 10 100

Como salida, interesa conocer la evolución de las variables S, I, R y V en el tiempo. Para esto usted enviará la salida a pantalla con el siguiente formato por cada línea (el tiempo avanza en unidades cercanas a un segundo hasta completar un valor cercano a T). La primera línea es de encabezado para describir cada columna, luego vienen líneas según la duración de la simulación.
Time, Vac, Inf, Rec, Sus
<t>, <V>, <I>, <R>, <S>

Archivos con este formato se conocen con el nombre de “archivos csv” (comma-separated values file). Éstos pueden ser fácilmente importados a una planilla electrónica. Se pide que usted haga eso y manipule la planilla para generar un gráfico de áreas apiladas donde la primera área es el número de individuos vacunados, luego el número de infectados, recuperados y finalmente los susceptibles de infectarse, todos como función del paso del tiempo desde 0 a T .

Llamando a su programa Pandemia.java, la ejecución de su tarea sería:
$ java Pandemia “archivo de entrada.txt”

Para enviar la salida de su programa a un archivo, usar:
$ java Pandemia “archivo de entrada.txt” > salida.csv

2.- Desarrollo en Etapas
Para llegar al resultado final de esta tarea usted debe aplicar una metodología de tipo "Iterativa e Incremental" para desarrollo de software. Usted y su equipo irán desarrollando etapas donde los requerimientos son abordados gradualmente. En cada etapa usted y su equipo obtendrá una solución que funciona para un subconjunto de los requerimientos finales o bien es un avance hacia ellos. **Su equipo deberá entregar una solución para cada una de las etapas** aun cuando la última integre las primeras. **El readme y archivo de documentación deben ser preparados solo para la última etapa. Prepare y entregue un makefile para cada una de las etapas.** Esto tiene por finalidad, educar en la metodología iterativa e incremental.

2.1.- Primera Etapa: Un individuo se mueven aleatoriamente
Es esta primera etapa no hay vacunatorios, nadie usa mascarilla y nadie se contagia. El objetivo de esta etapa es verificar el comportamiento de un individuo que se mueve aleatoriamente en la región rectangular.

Use y/o complete las clases Comuna, Individuo, Simulador y Stage1 disponibles aquí. Usted no está obligado(a) a usar estos códigos. Están para su conveniencia y en caso que le resulten útiles. Otras formas de estructurar el resultado para la etapa 4 también son válidos.

La clase Stage1 contiene el método main, crea una instancia de Comuna, una de Individuo y una de Simulador a partir del archivo de entrada. Esta clase es la encargada de correr la simulación.
La clase Simulador se ocupa de ejecutar la simulación siguiendo la recomendación dada al final de esta tarea. Esta clase maneja en avance del tiempo e imprimir los valores de salida del programa.
La clase Comuna define el territorio y contiene al único individuo que se mueve en ella.
La clase Individuo define la ubicación de los individuos dentro de la comuna y el movimiento de éstos dentro de ella.

Como salida esta etapa pide generar un archivo con formato
<t>, <x>, <y>   /* *x, y son las coordenadas del individuo en el tiempo. */*

Un ejemplo para una salida de este tipo una vez que se ha graficado en una planilla electrónica es se muestra en figura 1.

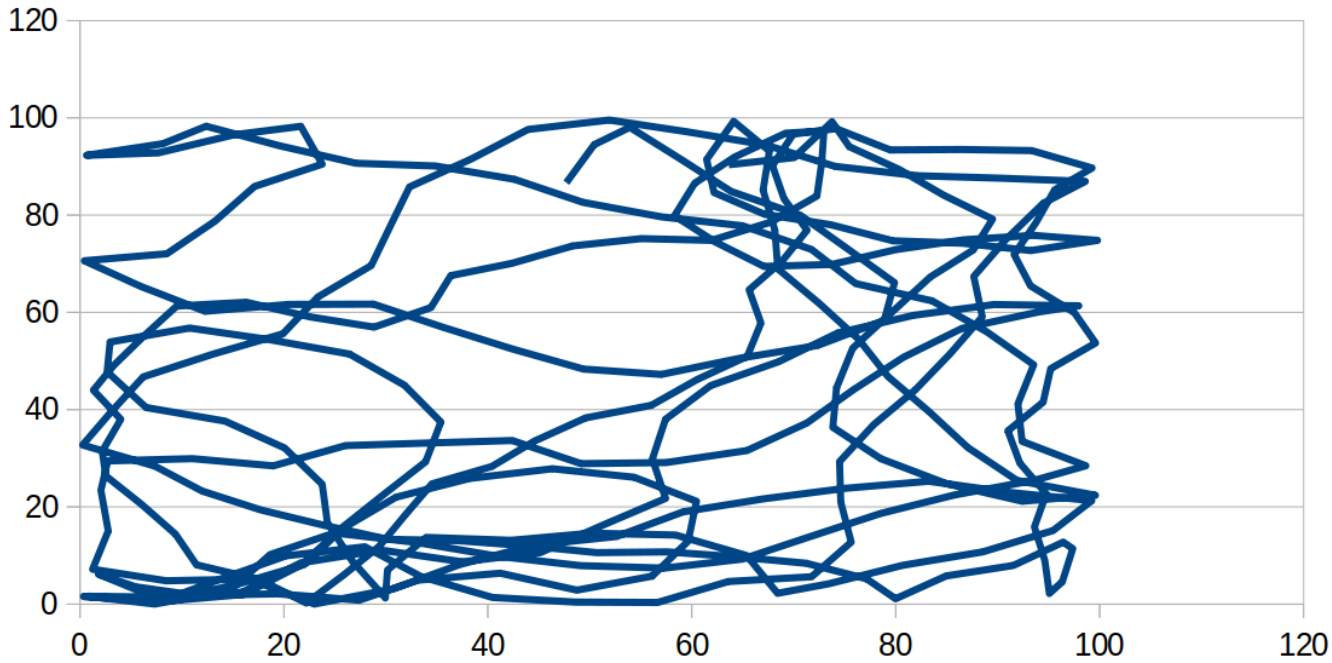

Figura 1: Ejemplo de recorrido aleatorio para individuo en comuna de 100 [m]x100[m].

2.2.- Segunda Etapa: Individuos se mueven aleatoriamente y pueden contagiarse
Esta etapa es similar a la previa (sin vacuna, sin mascarilla), excepto que se crean N individuos los cuales son almacenados en un “ArrayList <individuo>”. La clase simulador crea y ubica I individuos infectados y (N-I) individuos susceptibles de infectarse. La clase Individuo se debe completar para reflejar la interacción entre individuos.
Como salida se espera un archivo de salida similar al indicado en la descripción general de la tarea, excepto que no hay columna V.

2.3.- Tercera Etapa: Individuos se mueven aleatoriamente y algunos usan mascarillas
Esta etapa modifica la forma como los individuos de pueden contagiarse dependiendo del uso de mascarilla. Haga los cambios en la lógica del programa de la etapa previa para reflejar este comportamiento.
Como salida se espera un archivo de salida similar al indicado en la descripción general de la tarea, excepto que no hay columna V.

2.4.- Cuarta Etapa: Individuos se mueven aleatoriamente, algunos usan mascarillas y se inicia la vacunación.
En esta última etapa se espera alcanzar la funcionalidad descrita en la sección 1. Utilice un archivo de entrada con el formato señalado allí y genere un archivo de salida como el indicado en sección 1.
Usando una planilla de cálculo, manipule los datos para generar la gráfica pedida la cual usted incluirá en su documentación de la tarea.

2.5.- Extra crédito: Los individuos recuperados se vuelven susceptible al cabo de un tiempo.
Esta parte es voluntaria, su desarrollo otorga 5 puntos adicionales (la nota final se satura en 100).
Se trata de incluir un parámetro entero (R_time) opcional al final de la ejecución del programa de la Etapa 4. Cuando R_time es ingresado, los individuos recuperados se vuelven susceptibles de enfermar si éstos no han sido vacunados. Suponga que la inmunidad generada por la vacuna no se pierde. En su documentación mencione que desarrolló esta parte. Además del gráfico pedido en etapa 4, agregue un gráfico con la la evolución de la pandemia en este caso.

3.- Elementos a considerar en su documentación
Entregue todo lo indicado en “Normas de Entrega de Tareas” que se dará a conocer oportunamente.
Prepare un archivo makefile para compilar y ejecutar su tarea en aragorn con rótulo “run”. Además, incluya rótulos "clean" para borrar todos los .class generados. Los comandos a usar en cada caso son:
$ make /* *para compilar* */
$ make run /* *para ejecutar la tarea* */
$ make clean /* *para borrar archivos .class* */

En su archivo de documentación (pdf o html) incorpore el diagrama de clases de la aplicación (etapa 4).

4.- Nociones de simulación de fenómenos continuos
La idea básica usada en esta tarea es discretizar el tiempo; para cada instante discreto, congelamos el tiempo y pedimos a cada objeto de la simulación calcular cuál será su estado futuro (delta t más tarde) a partir del estado actual y las condiciones en que está sometido. Es importante congelar el tiempo y hacer este cálculo para cada objeto. Luego pedimos a todos actualizar su estado basado en lo calculado y avanzamos el tiempo en ese delta. Se debe operar así para evitar que un individuo se infecte y en el mismo instante éste pueda infectar a otros.
Esta simple idea trabaja bien aquí y en muchas situaciones. La calidad de la simulación aumenta al usar un Δt lo suficientemente pequeño; sin embargo, si en muy pequeño, la simulación requerirá mucho cálculo y tomará más tiempo de ejecución.

Se puede jugar cambiando los parámetros de entrada y analizar qué pasa si al cabo de un rato los individuos recuperados pueden pasar a ser susceptibles de infección. Lo mismo se puede analizar para aquellos vacunados. Como este ramo no es el único donde usted debe rendir bien, no entusiasmarse mucho.