Documentación
Tarea 1 - ELO329: Simulando la Evolución de una pandemia. (Versión 1.0)

Escrito por: **Miguel Andrade**

Manuel Cruces

Kevin Pizarro

Pablo Troncoso

Abril 10, 2021

Explicación breve de la solución:

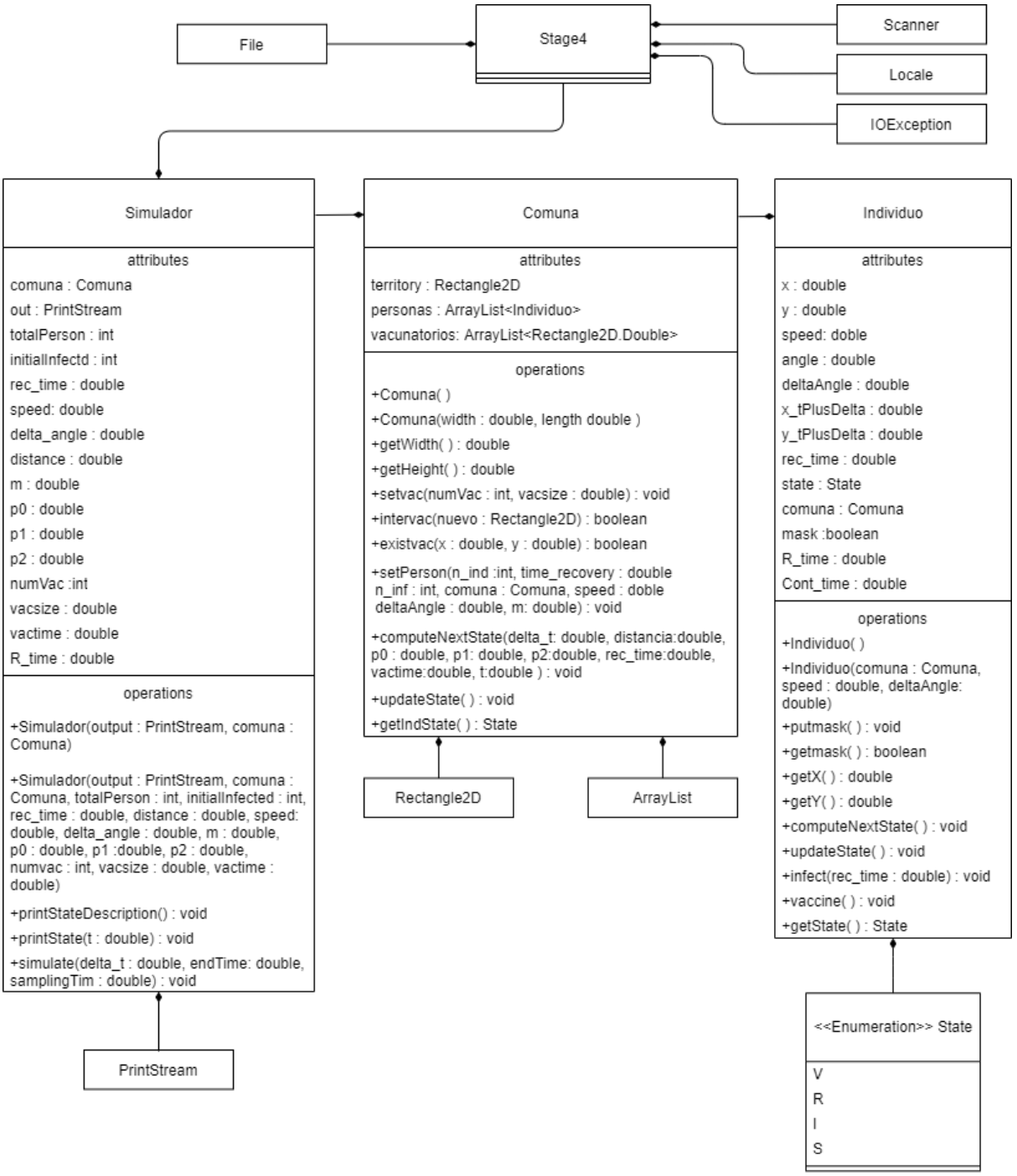
Se nos solicita implementar un simulador para una pandemia, el cual pueda recibir parámetros a través de un archivo de configuración, y pueda entregar los resultados pertinentes en un archivo de salida. Para esto (utilizando el código de ayuda), se tienen 4 clases: StageX, Simulador, Comuna e Individuo.

Stage X es la clase donde se encuentra el main y cuyo trabajo es inicializar simulador y comuna, así como recibir los parámetros del archivo de configuración (utilizando las librerías Scanner y Locale para leer el archivo, IOException para evitar errores y File para trabajar en los archivos) los cuales se entregan al Simulador, que se encargará principalmente de la simulación en sí, es decir, el manejo de los tiempos de la simulación e invocación de los métodos para actualizar e imprimir el estado de los individuos. Esto lo logra trabajando con la librería PrintStream y utilizando tiempos discretos, es decir, por cada instante de tiempo Δt se "detiene" a calcular los cambios en los individuos para luego reportarlos y agregarlos a sus variables correspondientes, además de otorgar los valores necesarios a comuna para que esta última se encargue de la creación de los individuos y vacunatorios.

En Comuna se inicializa el "espacio" y arreglo de los individuos que se mueven en este, haciendo uso de Rectangle2D y ArrayList respectivamente, llamando a las actualizaciones de estado de los mismos según lo indique el simulador. Luego se definen los vacunatorios y el método para ubicarlos en el espacio de la comuna. Por último, en individuo se parametriza el comportamiento y estados de estos; si está infectado, vacunado o es susceptible a infectarse. Luego se tienen los cálculos propios de su movimiento "aleatorio", el cual está determinado por los parámetros entregados por el archivo de configuración.

Se adjunta el diagrama de clases UML en la siguiente página.

Diagrama 1: Diagrama de clases UML para Etapa Extra.



Dificultades encontradas:

1. Lectura de datos “.” en vez “,” como separador decimal en StageX.java

Solución: Investigar respecto al tema de que utiliza java como separador decimal, y ya que depende un poco de la configuración del computador, se utiliza la clase importada “Locale” que permite configurar estos parámetros, permitiendo que tome los valores de forma estandarizada sin importar la configuración del computador.

2. Entendimiento y ejecución de makefile.

Solución: Se realizó una mejor lectura del material dispuesto y se revisaron errores a más detalle que podría haber tenido el archivo.

3. Dificultad: Evitar que los vacunatorios se “intersectan”.

Solución: Crear un set de reglas para generar los vacunatorios, además de un método que verifica si el vacunatorio que se va a colocar intersecta con alguno de los vacunatorios anteriores, esto está dentro de un ciclo while que provoca que el vacunatorio a colocar cambie de posición hasta que no intersecta con ningún vacunatorio anterior, en el caso de que no se pueda colocar el vacunatorio (después de un número fijo de iteraciones de dicho while), genera los vacunatorios desde el inicio, ya que según la cantidad y tamaño de los vacunatorios, y el tamaño de la comuna, se puede dar el caso que no se puedan colocar la cantidad solicitada de vacunatorios si es que se generan en posiciones que “desperdicien” mucho espacio.

Gráficos de Salida:

Para obtener los siguientes gráficos se simuló con la configuración dada por defecto en el repositorio de GitLab.

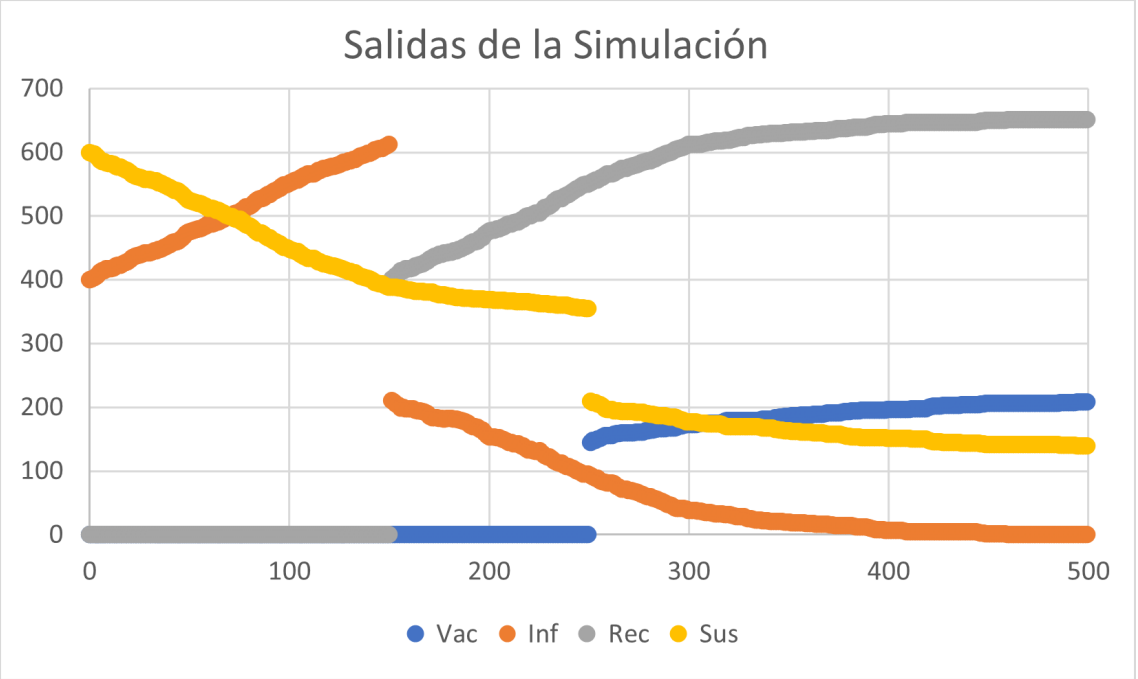


Gráfico 1: Salida de Stage 4.

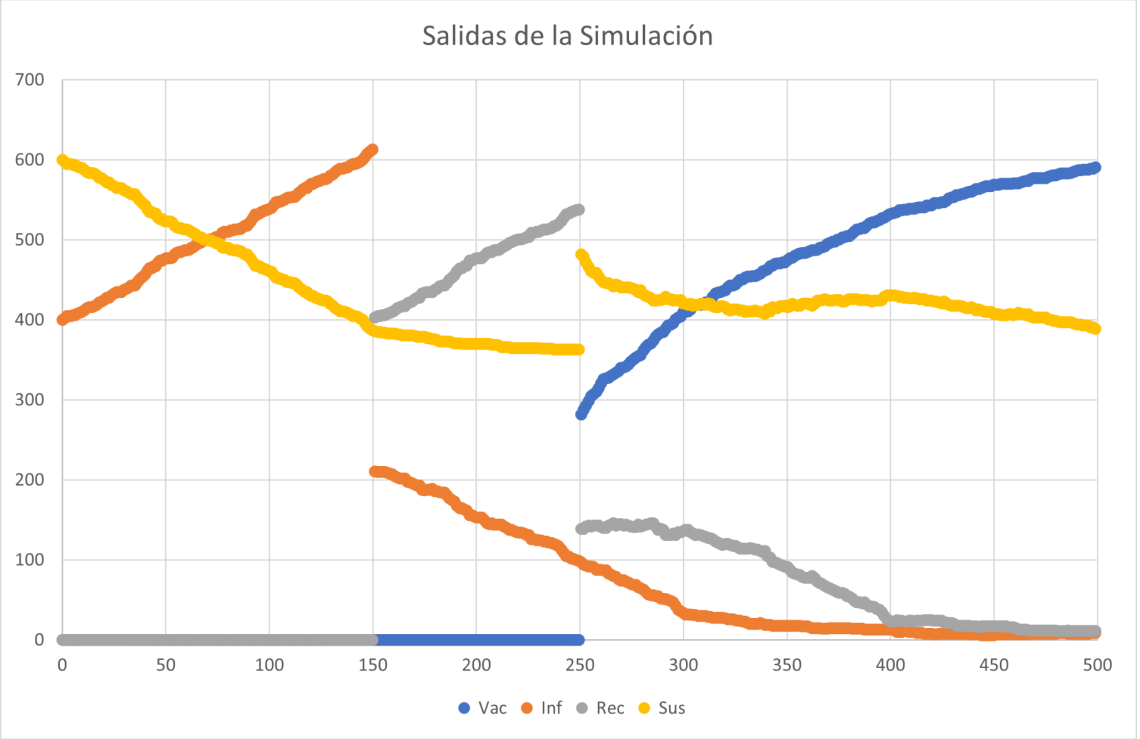


Gráfico 2: Salida de Etapa Extra.