Tecnológico de Monterrey

Modelación de sistemas multiagentes con gráficas computacionales (Gpo 4)

AI. Actividad Integradora Parte 1

David Josué Marcial Quero

Ingeniería en Tecnologías Computacionales (ITC)

Campus Monterrey

Descripción del problema

¡Felicidades! Eres el orgulloso propietario de 5 robots nuevos y un almacén lleno de cajas. El dueño anterior del almacén lo dejó en completo desorden, por lo que depende de tus robots organizar las cajas en algo parecido al orden y convertirlo en un negocio exitoso.

Cada robot está equipado con ruedas omnidireccionales y, por lo tanto, puede conducir en las cuatro direcciones. Pueden recoger cajas en celdas de cuadrícula adyacentes con sus manipuladores, luego llevarlas a otra ubicación e incluso construir pilas de hasta cinco cajas. Todos los robots están equipados con la tecnología de sensores más nueva que les permite recibir datos de sensores de las cuatro celdas adyacentes. Por tanto, es fácil distinguir si un campo está libre, es una pared, contiene una pila de cajas (y cuantas cajas hay en la pila) o está ocupado por otro robot. Los robots también tienen sensores de presión equipados que les indican si llevan una caja en ese momento.

Lamentablemente, tu presupuesto resultó insuficiente para adquirir un software de gestión de agentes múltiples de última generación. Pero eso no debería ser un gran problema ... ¿verdad? Tu tarea es enseñar a sus robots cómo ordenar su almacén. La organización de los agentes depende de ti, siempre que todas las cajas terminen en pilas ordenadas de cinco.

Realiza la siguiente simulación:

- Inicializa las posiciones iniciales de las *K* cajas. Todas las cajas están a nivel de piso, es decir, no hay pilas de cajas.
- Todos los agentes empiezan en posición aleatorias vacías.
- Se ejecuta el tiempo máximo establecido.

Deberás recopilar la siguiente información durante la ejecución:

- Tiempo necesario hasta que todas las cajas están en pilas de máximo 5 cajas.
- Número de movimientos realizados por todos los robots.
- Analiza si existe una estrategia que podría disminuir el tiempo dedicado, así como la cantidad de movimientos realizados. ¿Cómo sería? Descríbela.

Definición del modelo

Puesto que se trata de un problema en el cual hay ciertos elementos no bien definidos en la descripción de este, me tomaré ciertas libertades en la interpretación de algunos de los componentes del problema y su implementación. En primera, creo que es importante destacar que el problema nunca establece claramente que los Robots tienen que estar en la misma celda sobre la cual se encuentra una cierta caja para poder recogerla, por lo cual pueden haber diferentes posibilidades: el Robot puede tomar una caja si esta se encuentra adyacente a él (solo en cuatro direcciones); tiene que estar sobre la caja para poder recogerla; o una combinación de ambas, sin embargo, esta

combinación sería una mejora no tan notoria, ya que en muchos casos se estaría recogiendo la misma cantidad y casi con la misma eficiencia que en la primera opción.

Además, dado que existen tres posibilidades de implementación en el movimiento y toma de cajas, se tienen consecuencias secundarias interesantes. Una de ellas es que, dependiendo de cómo se implemente la toma de cajas, se estará estableciendo por inferencia los siguientes casos: El Robot puede moverse a través de las cajas cuando este sólo las recoge estando sobre la caja en sí; la otra opción es que el Robot no puede pasar a través de dichas cajas cuando este recoge las cajas adyacentes a él, esto siendo así, no porque no se pueda tener el comportamiento primero, sino para apegarse un poco más a la realidad y tener un modelo que claramente refleje dicha realidad.

La segunda área en la que se puede interpretar de distintas formas la descripción del problema es en el requisito de ordenar las cajas en pilas de 5, pues solo se establece que este es el producto final, mas no se nos provee de una coordenada específica en la que estas cajas deben ser apiladas, por lo cual se podría simplemente apilar las cajas sobre la primera que un cierto Robot detecte.

A partir de todos estos elementos interpretativos, podemos llegar a diferentes modelados del mismo problema. Al final, terminé implementando tres modelos bastante similares entre sí, pero con unas claras diferencias en los elementos que acabo de discutir, por lo cual la eficiencia de cada uno será diferente a las demás. La primera implementación que realicé fue la más “realista”, pues en esta los Robots solo pueden tomar las cajas adyacentes y no pueden pasar encima de estas. La segunda modelación establece un Robot que solo toma las cajas que están debajo de él y, por lo tanto, puede pasar “a través” de ellas. Finalmente, la última modelación que realicé fue una en la que los Robots puede recoger cajas adyacentes a su posición y pasar a través de cajas.

Cabe mencionar que, en todas estas modelaciones, los Robots no tienen una coordenada específica en la cual colocar las cajas, por lo cual, estos, al encontrarse con una caja que no tenga apilada 4 cajas adicionales, tomará esta caja el lugar sobre el cual colocará las que encuentre en su deambular aleatorio.

Tipos de agentes y forma de interacción entre ellos

Para el caso que se está intentando modelar, es evidente que se requerirá utilizar agentes racionales, pues estos son aquellos que hacen lo correcto, es decir, para cada secuencia de percepciones, la función del agente devuelve la acción correcta. Además, para hacer una distinción especial entre cada agente que se estará detallando más adelante, la siguiente clasificación es posible:

- Robot: Es un agente reactivo simple, pues tomará decisiones con base en el input actual y se realiza una actividad (avanzar, tomar, etc.).

- Las cajas: serán agentes reactivos basados en modelos. Pues estos realizan un seguimiento del entorno a través de un modelo interno con el cual conoce la evolución de este de acuerdo con las acciones realizadas por los robots.

Hablando de la interacción que los agentes tendrán entre el uno y el otro, es claro que, antes que nada, para poder comunicarse, los agentes deben tener un conjunto de términos comunes acordados. Por lo tanto, se puede utilizar una pequeña modificación del lenguaje estándar para la comunicación entre agentes denominada FIPA-ACL. De esta forma, se podrá “Informar” y “Solicitar” información pertinente del estado de cada uno de los agentes, de forma tal que puedan actuar con inteligencia y conciencia de los demás agentes. Para esto, es importante mencionar que los agentes serán benevolentes, pues estos se ayudarán mutuamente para tener el apilamiento de cajas más eficiente posible.

De igual forma, creo que es evidente que los agentes tendrán una resolución cooperativa distribuida de problemas, pues ninguno de los agentes estará encargado de almacenar toda la información suficiente para resolver el objetivo de organizar el almacén. Por lo tanto, se tendrá una red de contratos, con lo cual habrá distribución de tareas que lleven a tener un almacén ordenado.

Definición de agentes del modelo

Se tienen dos agentes principales en las tres modelaciones que mencioné en el apartado anterior, y con comportamientos semejantes, por lo cual considero innecesario realizar tres definiciones de agentes. Entonces, los dos agentes que se tienen para llevar a cabo esta modelación son: Caja y Robot. La caja solo será un agente que lleve la cuenta de cuantas cajas estás apiladas sobre sí. El Robot, será el agente que se mueva a través del almacén, encontrando cajas y ordenándolas en conjuntos de 5.

- Caja:

  Este agente, es el que ocupará todas las casillas del modelado. Evidentemente, seguirá los lineamientos establecidos en la definición del problema, es decir, que habrá una cantidad *K* de Cajas colocadas aleatoriamente en un almacén. Por lo tanto, para hacer esto, aquellas casillas que no tengan una Caja sí tendrán el agente Caja, pero con una cantidad de cajas igual a cero.

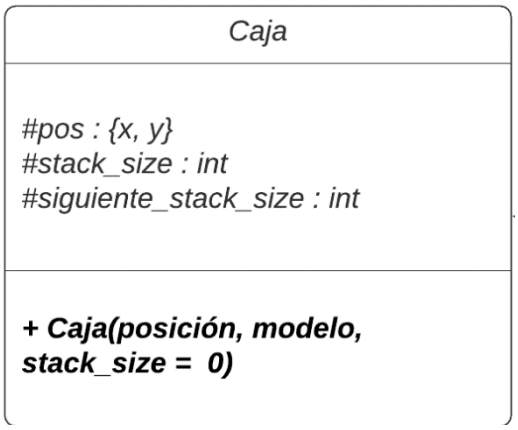
  Los atributos de una Caja son los siguientes:

  - Posición: coordenadas *x* y *y*.
  - Tamaño de la pila de cajas: cantidad de Cajas que se encuentran sobre una cierta celda.

- Siguiente tamaño de la pila: variable que servirá para tener un mayor orden en la actualización de la cantidad de cajas que una cierta celda tiene apiladas.

Los métodos que tendrá una determinada Caja serán los siguientes:

- Constructor: Método con el cual se inicializará una instancia de esta clase. Se inicia una determinada Caja con una posición, tamaño de pila igual a cero y tamaño siguiente igual a None.



- Robot:

Este agente, como mencioné en el apartado anterior, tendrá diferentes comportamientos y limitaciones dependiendo de cómo se interprete la definición del problema. En lo particular, para la resolución de esta Actividad Integradora, he realizado tres versiones de este agente y a continuación detallo sus diferencias:

- Versión 1:
  - Solo recoge cajas adyacentes a él.
  - No puede pasar “sobre” una caja.
- Versión 2:
  - Solo recoge cajas debajo de él.
  - Puede pasar “sobre” una caja.
- Versión 3:
  - Solo recoge cajas adyacentes a él.
  - Puede pasar “sobre” uina caja.

Todas estas versiones tendrán como fundamento que el Robot solo puede moverse en cuatro direcciones (izquierda, derecha, arriba y abajo) y esta dirección será elegida de forma aleatoria.

En cada turno, el Robot, si no se encuentar cargando alguna caja, se moverá aleatoriamente por el almacén hasta encontrar una caja. En el momento en el que

detecte una caja adyacente (o una caja debajo de él para la versión 2) y no esté carganodo una, establecerá esta caja como la caja sobre la cual comenzará a apilar otras cajas. Ahora seguirá deambulando por el almacén hasta toparse con otra caja o pila de cajas menor a 5, una vez encontrada una o varias, solo tomará una de estas cajas y, evidentemente, cambiará la cantidad de cajas que esa celda tenía.

Ahora, el Robot intentará regresar a la primera caja que se encontro y que estableció como la caja sobre la que empilará. Tomé en consideración varios escenarios en los que podría toparse con otras cajas, y por lo tanto, no poder continuar su camino (este caso corresponde a la versión 1 del problema, ya que en esta versión el Robot no puede pasar sobre una caja), de forma tal que un Robot llegará a su objetivo, siempre y cuando esta pila de cajas no esté totalmente inaccesible.

Para evitar la colisión entre Robots, en lugar de utilizar SimultaneousActivation, utilicé el scheduler RandomActivation, con lo cual, siempre una Robot se moverá antes que otro (lo cual es practicamente imperceptible para el ser humano) y de esta forma, con el simple hecho de elegir entre las direcciones que no tienen otro Robot en esas casillas adyecentes, se puede evitar la colisión.

Cuando un Robot consiga apilar 5 cajas, ningún otro Robot ni él podrá seguir apilando cajas en esta casilla. Ahora ya no tendrá coordenadas de una caja sobre la que apilar, por lo que todo el ciclo se vuelve a repetir.

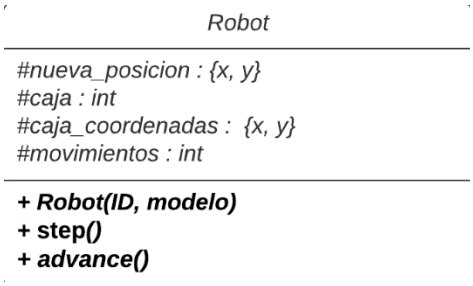
Los atributos de un Robot son los siguientes:

- Nueva posición: coordenadas *x* y *y*.
- Caja: cantidad de cajas que carga el Robot (solo puede ser una así que funciona como True o False).
- Coordenadas de la caja: Coordenadas de la caja sobre la que se apilarán más cajas hasta llegar a una pila de tamaño 5.
- Movimientos: cantidad de movimientos realizados por el Robot.

Los métodos que tendrá un determinado Robot serán los siguientes:

- Constructor: Método con el cual se inicializará una instancia de esta clase. Se inicia un determinado Robot con una posición, cantidad de cajas cargadas igual a cero, coordenadas de caja igual a None y tamaño siguiente igual a None.
- Step: Método en el cual se realiza la comprobación de si trae una caja el Robot y actúa de acuerdo con esto. También, en caso de ser posible, elige una dirección en la que moverse.
- Advance: Método que lleva a cabo el cambio de las variables que se acaban de modificar, tal como la posición del Robot, el tamaño de la pila de cajas

en la que acaba de colocar o tomar, así como aumentar el número de movimientos si es que el Robot se movió.



Modelo Mesa

Habrá una clase que herede los métodos de Model de Mesa, por lo cual, esta clase será la encargada de inicializar todos los demás componentes de la modelación descrita previamente. A su método constructor se le pasarán, desde una función inicializadora, los siguientes atributos:

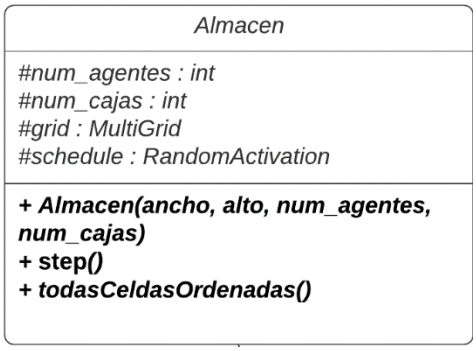
- El ancho del modelo (cantidad de celdas en la orientación horizontal).
- La altura del modelo (cantidad de celdas en la orientación vertical).
- Cantidad de robots.
- Cantidad de cajas.

En este método constructor se hará la inicialización de las instancias de los agentes Caja (algunas vacías, es decir, una celda sin cajas y otras con una caja), de acuerdo con lo estipulado en la definición del problema: una cantidad específica de cajas con una posición inicial aleatoria., De igual forma, se poblarán las celdas de forma aleatoria, con robots, para lo cual es necesario tener Multigrid, pues así se les permite a los agentes compartir el mismo espacio (Robot sobre Caja).

Los métodos que tendrá esta clase serán los siguientes:

- Constructor: Método con el cual se inicializará una instancia de esta clase.
- Step: Método con el cual se inicializa un nuevo turno.
- todasCeldasOrdenadas: Método con el cual se puede verificar si todas las cajas han sido ordenadas, es decir, puestas en pilas de 5.

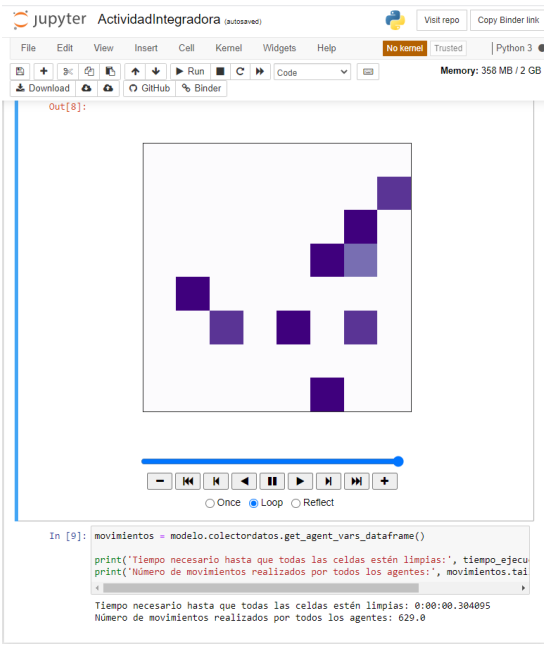
Además de todos los métodos correspondientes a la clase del modelo, también se tendrá una función que ayudará a guardar la información sobre la posición de los robots y cajas en cada uno de los turnos, con lo cual se podrá graficar más adelante todo el transcurso de la simulación.



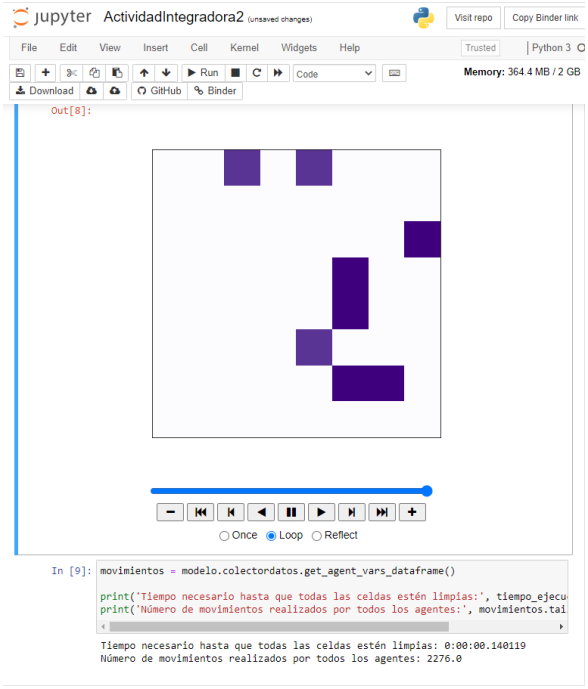
Conclusiones, análisis y mejoras del modelo

Despues de llevar a cabo pruebas con cada uno de los diferentes modelos propuestos en este documento, considero que el más eficiente de ellos es el último. Este tiene la funcionalidad de que cualquire Robot puede tomar cajas que esten adyacentes a él, además de que puede moverse libremente a través de las cajas e incluso apilar en cajas sobre las cuales se encuentra en un dado momento . Aunque no es la versión más realista, sí es aquella que logra tener un desempeño muy bueno, pues el acto de apilar cajas no tiene prácticamente ningún obstaculo, más que el que se presente otro Robot en el camino, pero esto no es catastrófico, ya que uno de estos se moverá y solo se perdería un turno en el peor de los casos.

Cuando se comparan los resultados de la ejecución de las tres versiones del problema, nos damos cuenta de las ventajas de cada uno. La primera versión (la más realista), tiene uno de los peores tiempos de ejecución, debido a la limitación de no poder pasar a través de las cajas, por lo cual su movimiento es bastante limitado, no obstante, tiene la segunda mejor cantidad de movimientos, pues al poder tomar cajas adyacentes a él, en teoría podría apilar varias cajas sin necesidad de moverse por varios turnos.

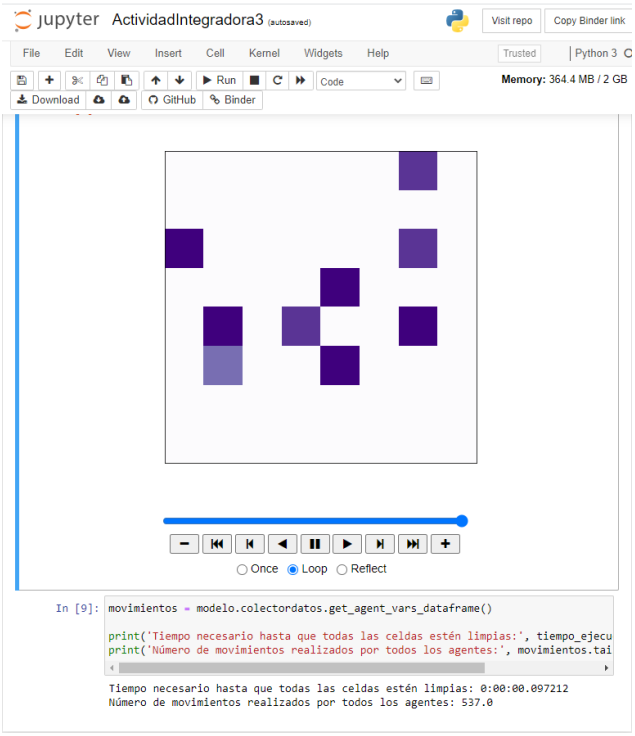


En la segunda versión se tiene un mejor tiempo de ejecución al anterior, aunque no es una mejora tan sobresaliente. Sin embargo, puesto que no puede recoger cajas adyacentes, la cantidad de movimientos es la más grande de todas las versiones, como era de esperarse, por lo cual considero esta versión una de las peores y realmente fue realizada solo para demostrar qué elecciones en la interpretación del problema tendrían como resultado una peor eficiencia.



Finalmente, como era de esperarse, la versión del problema con la mayor eficiencia tanto en el tiempo de ejecución como en la cantidad de movimientos es la tercera, pues a estos

robots se les permite moverse a través de cajas y tomar cajas adyacentes. Sin embargo, es un poco inconsistente en la cantidad de movimientos, pues en algunos casos se moverá exorbitante cantidad de veces, pues, al igual que en la versión dos, no tiene muchos impedimentos para moverse y esto provoca que casi en todos los turnos se muevan casi todos los robots.



No obstante, creo que hay un diseño más eficiente que reduciría el tiempo dedicado así como la cantidad de movimientos realizados, el cual no implementé. Para este no se tendría el movimiento de los robots de forma aleatoria y, además, tendría coordenadas predeterminadas en las que apilar las cajas. Es más, se podría tener que los robots realicen una limpieza totalmente organizada, es decir, que cada uno de estos robots se encarge de apilar (ordenar) una fila del grid, otro robot la siguiente, etc. Al tener una coordenada específica sobre la cual apilar cajas, también se logra que no haya tantas incertidumbre en cuanto a las posiciones de las cajas. Sin embargo, para hacer que los robots no recorran grandes distancias para obtener una caja y luego llevarla a la coordenada de apilamiento, se podría tener esta coordenada en una posición central, lo cual reduciría la cantidad de movimientos necesarios.

Diagrama de clases

Diagrama de clase: Actividad Integradora

David Josué Marcial Quero | A00828702

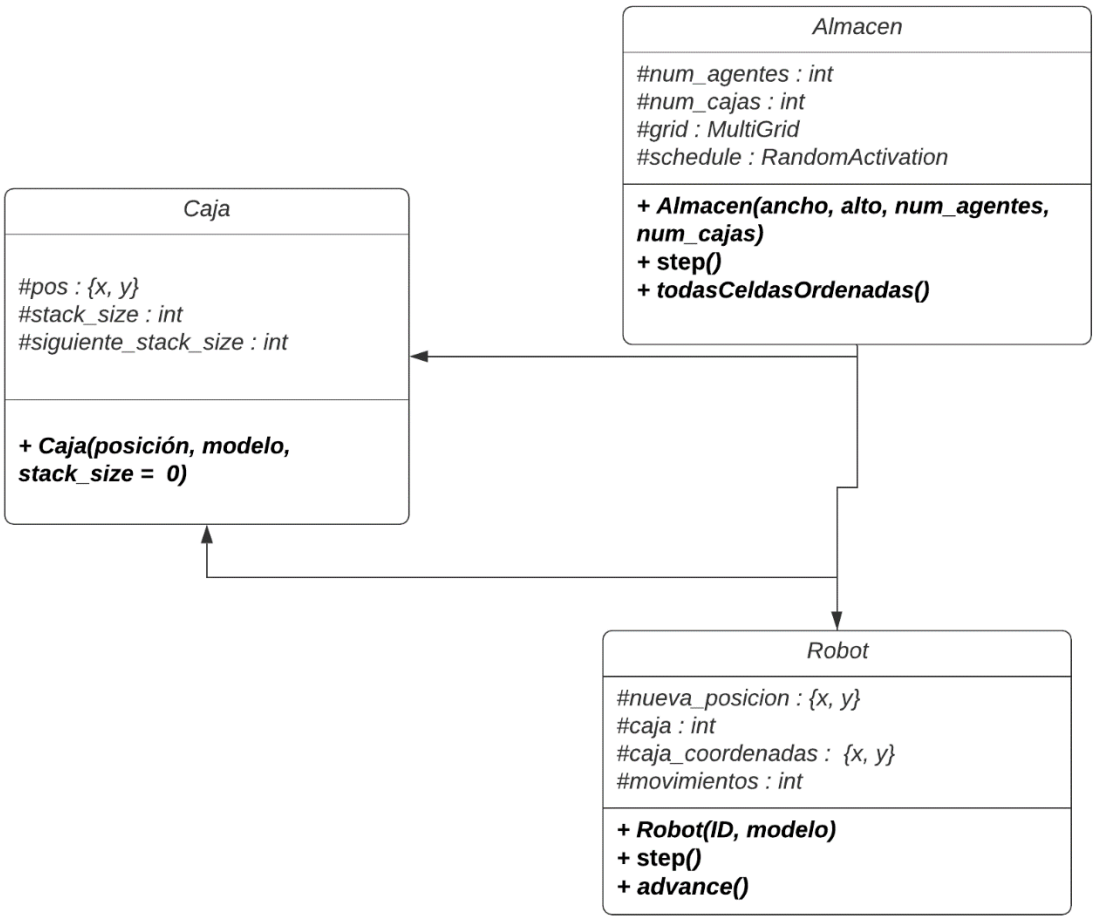


Diagrama de secuencia

Diagrama de secuencia Actividad Integradora

David Josué Marcial Quero | September 6, 2021