[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2 - Informe

Segundo cuatrimestre del año 2020

Zocchi, Tomás	103391	tzocchi@fi.uba.ar
Bilo, Lucas	103252	lbilo@fi.uba.ar
Leon, Agustina	102208	agleon@fi.uba.ar

Supuestos

- Los movimientos del personaje son de a un paso a la vez y este paso no puede darse en direcciones diagonales.
- Un personaje no puede moverse fuera del tablero de juego.
- Los bloques proporcionados pueden ser utilizados sin seguir un orden específico.
- No pueden crearse bloques invertidos, repetidos o personalizados vacíos.
- El personaje siempre comienza en el centro del tablero.

Diagramas de clases

El diagrama muestra la relación entre todos los objetos presentes en el programa.

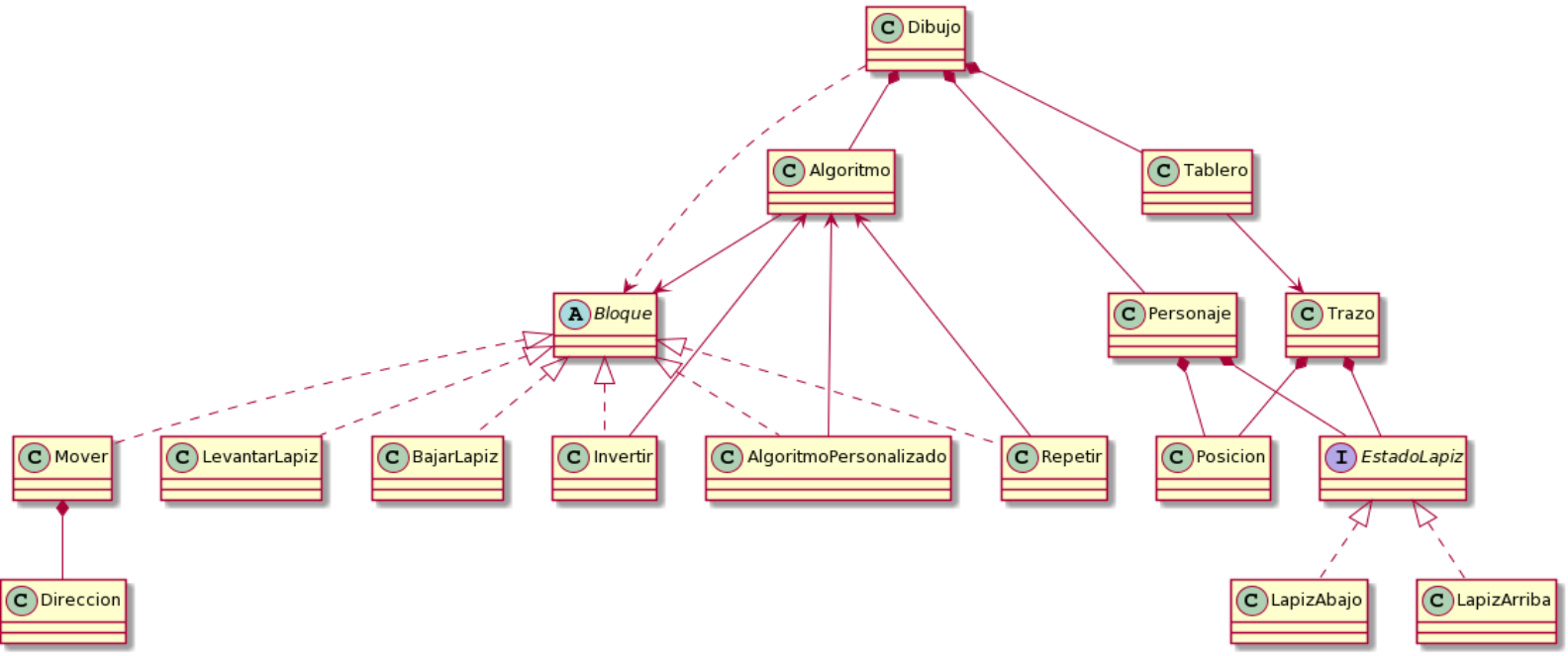

Figura 1: Diagrama general de clase.

El diagrama muestra las relaciones de composición entre los objetos. Un Dibujo no puede existir sin el algoritmo que ejecuta, que a su vez está compuesto por bloques, el personaje que ejecuta los movimientos y el tablero en el cual se reflejan dichos movimientos.

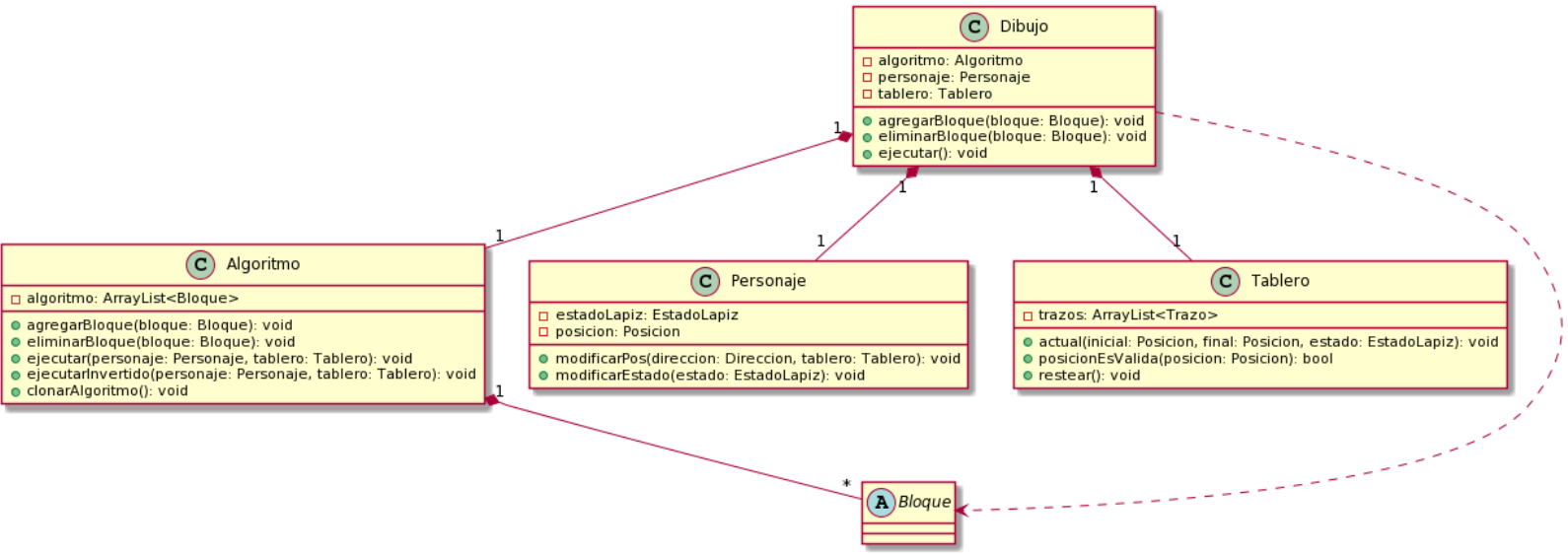

Figura 2: Diagrama de relaciones entre Dibujo, Algoritmo, Personaje y Tablero.

El siguiente diagrama muestra la relación de composición entre Personaje y los tipos de sus atributos. El personaje es creado con el lápiz arriba y siempre posee un estado de ese lápiz al igual que una posición.

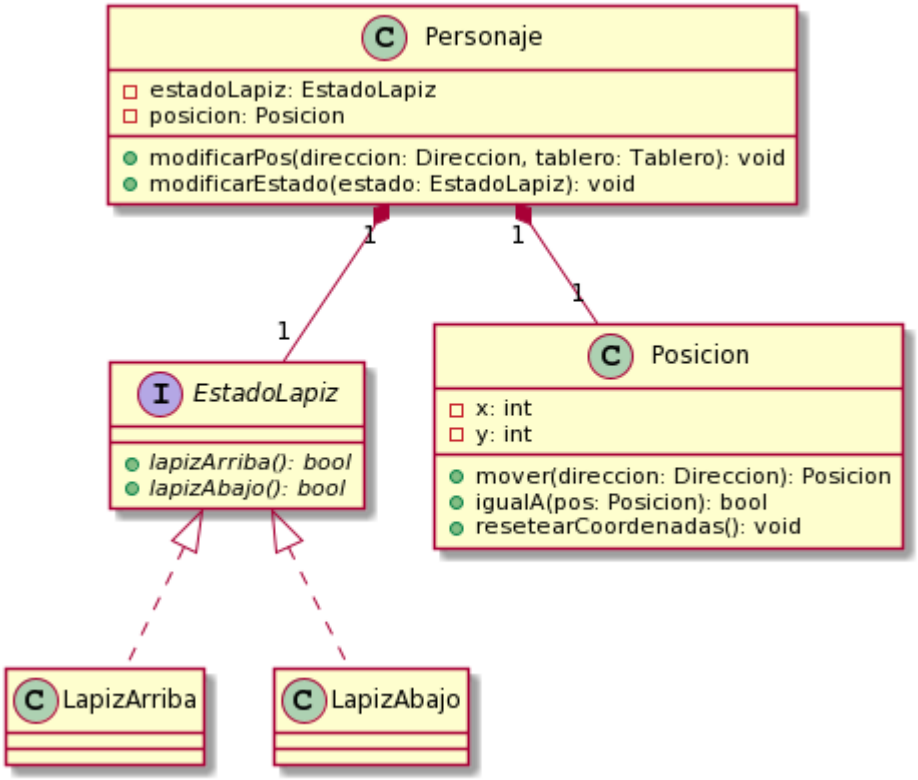

Figura 3: Diagrama de relaciones entre Personaje, Estado Lápiz y Posición

En el siguiente diagrama se muestra la relación entre Tablero, Trazo y Posición. El tablero guarda los trazos realizados por el personaje, siendo estos trazos la posición de donde parte, hacia donde llega y en qué estado se encuentra el lápiz en ese movimiento.

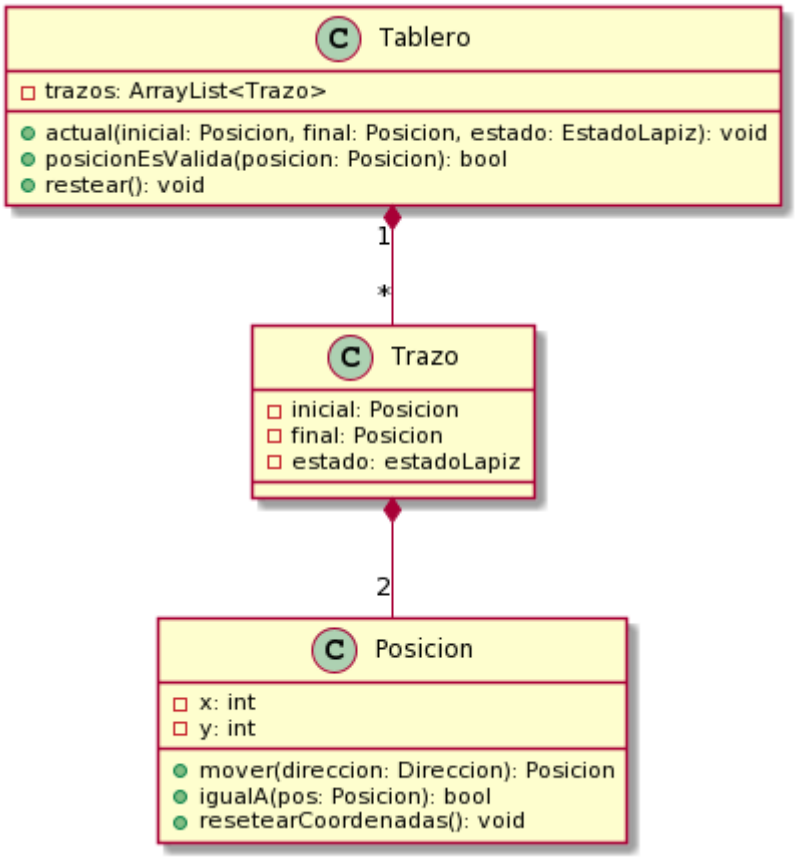

Figura 4: Diagrama de relación entre Tablero, Trazo y Posición.

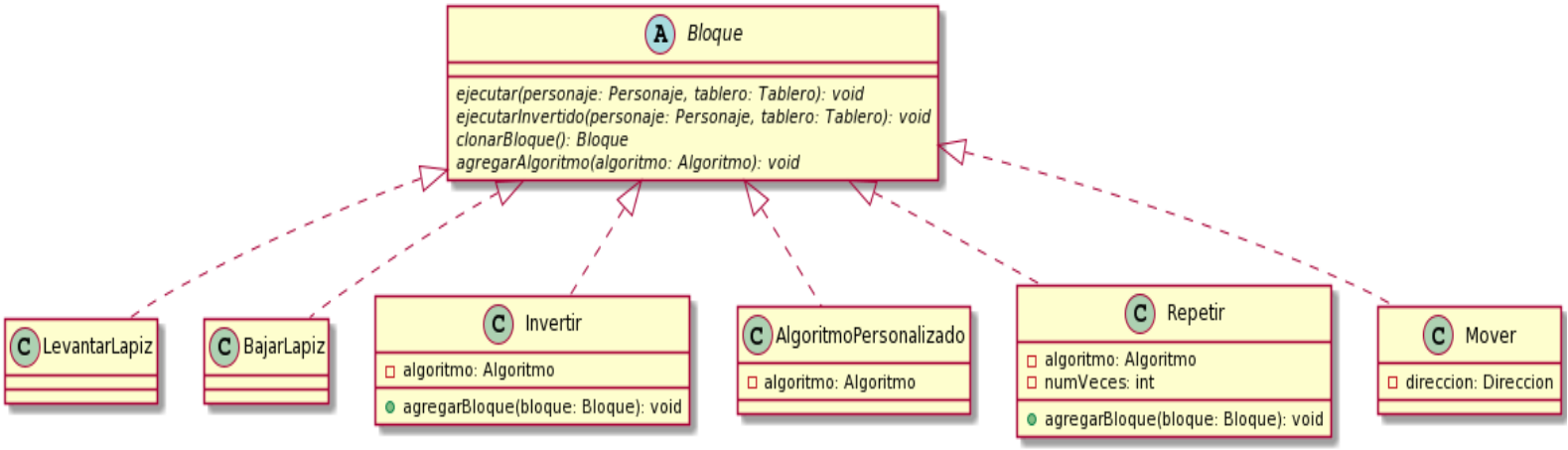

Figura 5: Diagrama de herencia de Bloque.

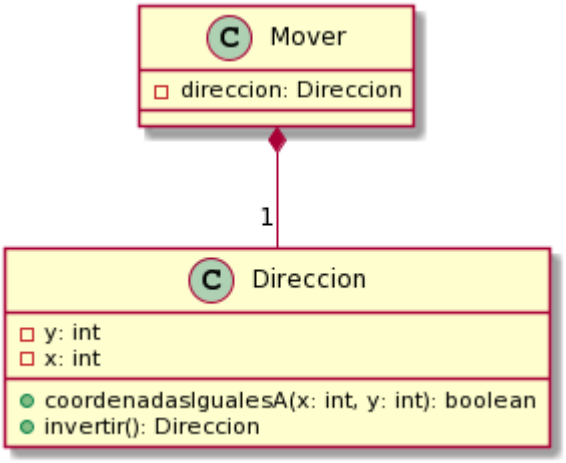

Figura 6: Diagrama de relación entre bloque Mover y Dirección

Diagramas de secuencia

En el siguiente diagrama puede observarse la acción de bajar el lápiz del personaje y moverse hacia arriba.

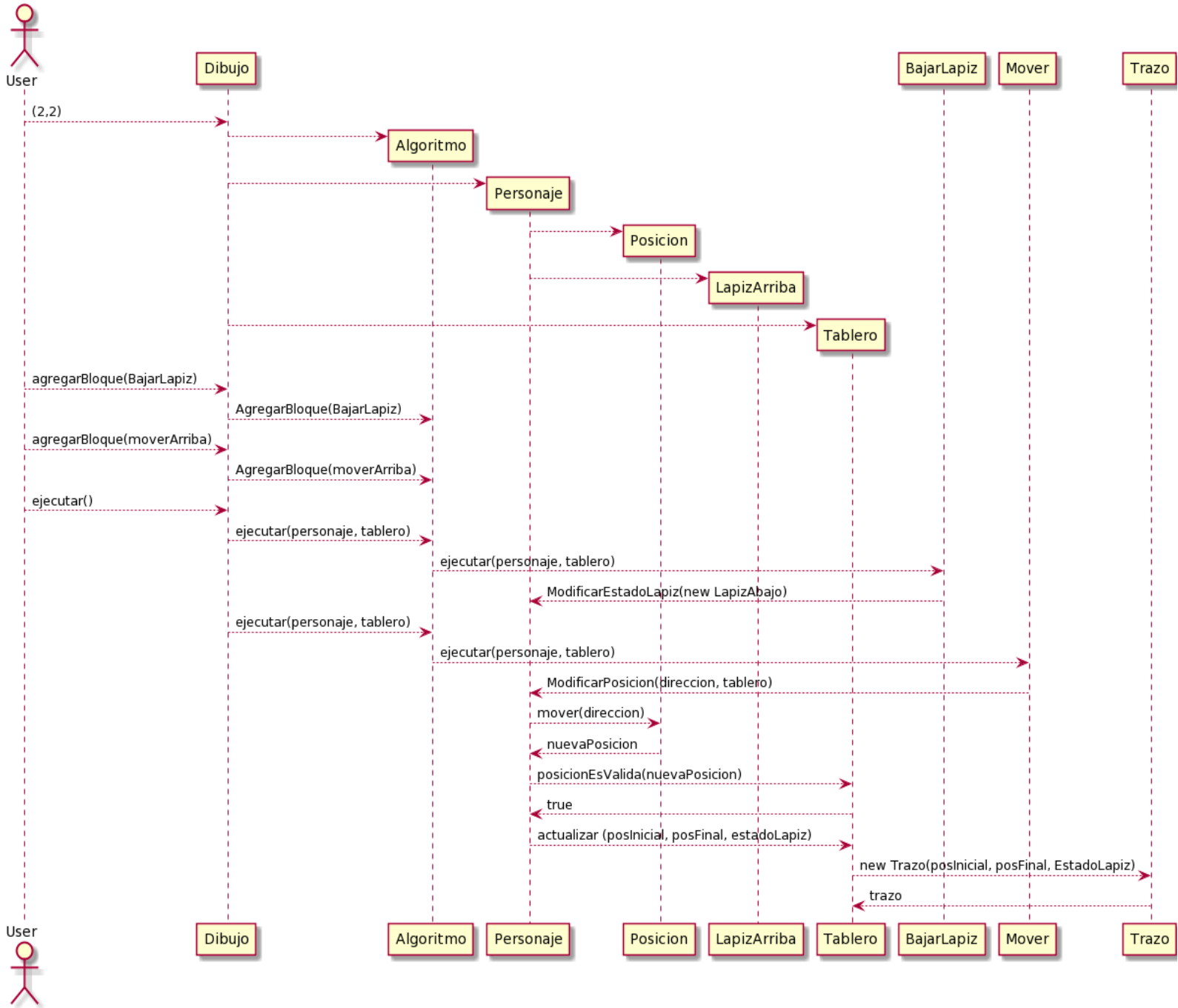

Figura 7: Diagrama de secuencia.

El siguiente diagrama muestra el caso límite de un personaje que trata de moverse fuera del límite establecido por el tamaño del tablero.

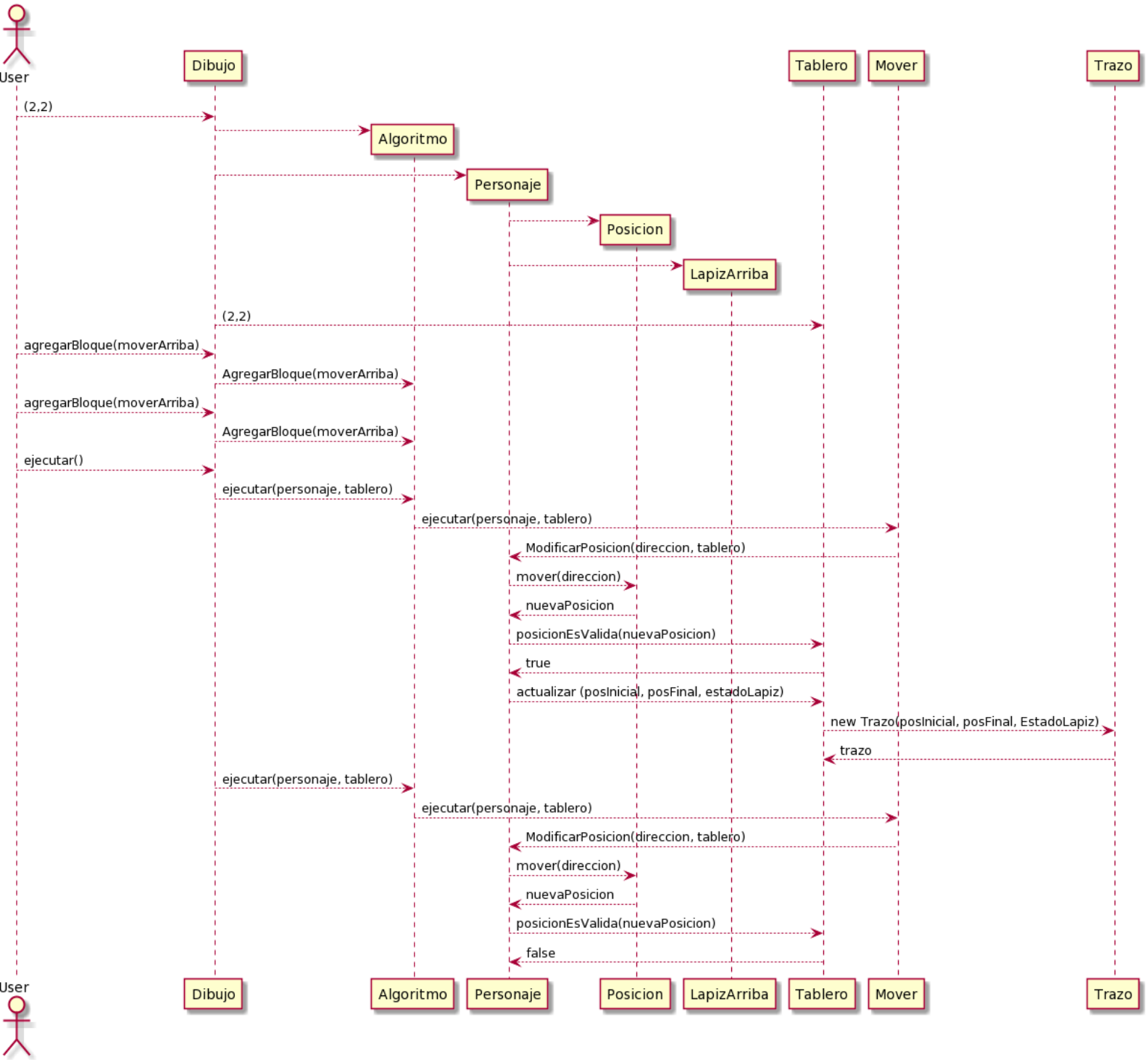

Figura 8: Diagrama de secuencia.

El siguiente diagrama muestra cómo se ejecuta un bloque mover de forma invertida.

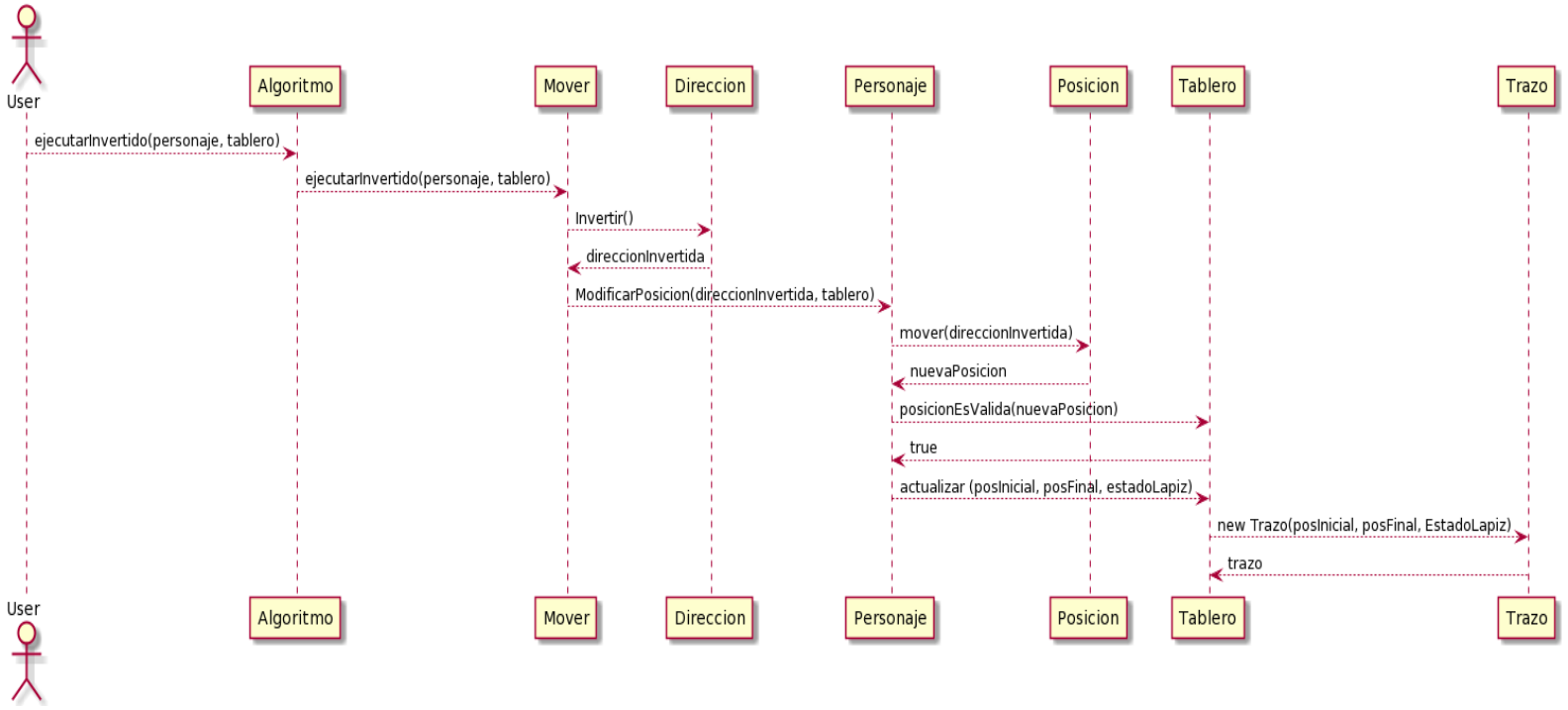

Figura 9: Diagrama de secuencia de ejecución invertida de bloque mover.

El siguiente diagrama muestra la ejecución de un Algoritmo que contiene un bloque AlgoritmoPersonalizado

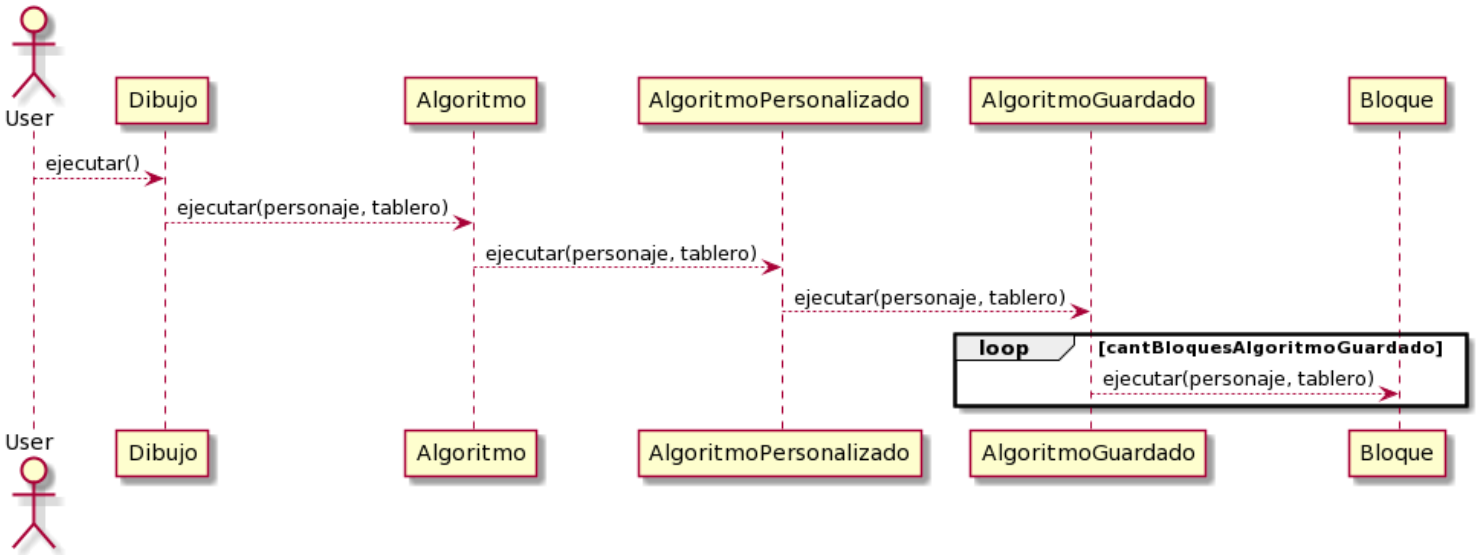

Figura 10: Diagrama de secuencia de ejecución de algoritmo con AlgoritmoPersonalizado.

El siguiente diagrama muestra la ejecución de un algoritmo que contiene un bloque Invertir.

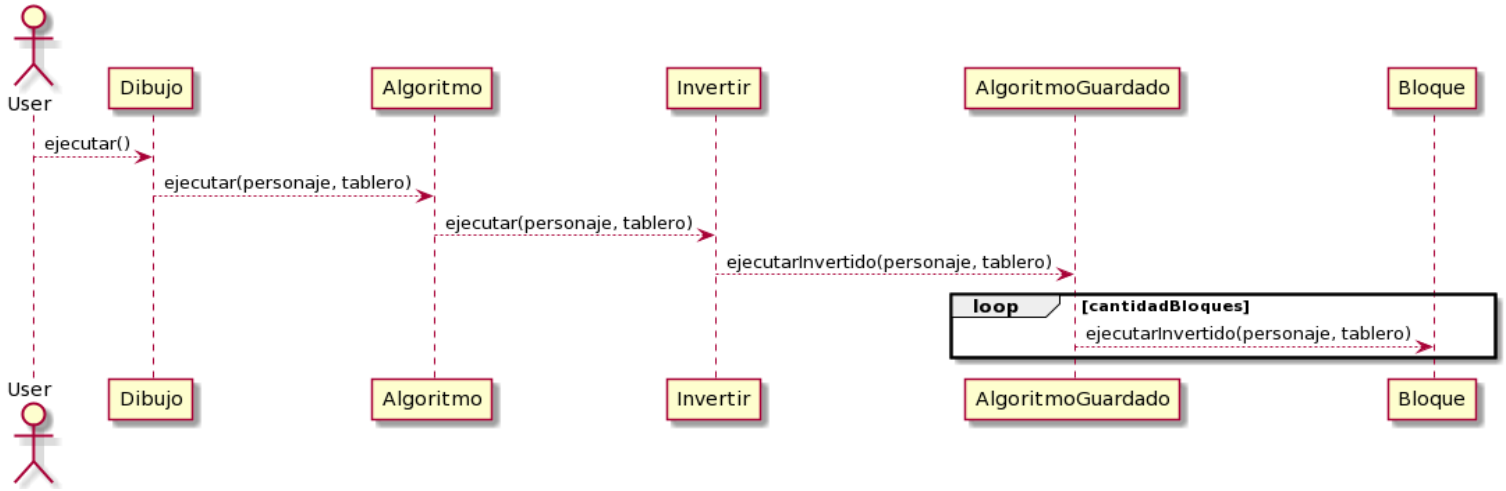

Figura 11: Diagrama de secuencia de ejecución de algoritmo con bloque Invertir.

El siguiente diagrama muestra la ejecución de un algoritmo que contiene un bloque Repetir. El atributo “numVeces” es la cantidad de veces que se repite el algoritmo que guarda el bloque repetir, el cual puede ser 2 o 3 veces.

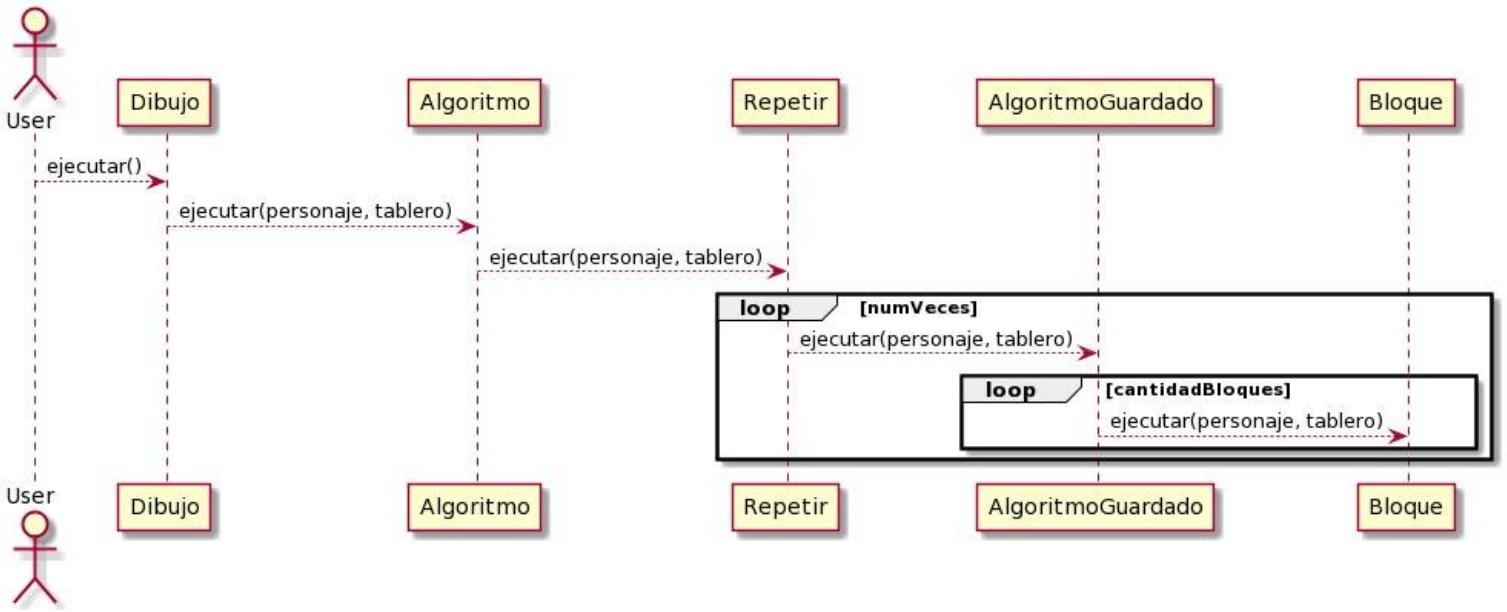

Figura 12: Diagrama de secuencia de ejecución de algoritmo con bloque Repetir.

Diagrama de paquetes

El siguiente diagrama de paquetes muestra el acoplamiento de nuestro programa. Se puede ver claramente la división entre el modelo del juego y la interfaz gráfica. Luego, esta última se encuentra dividida entre las imágenes que utilizamos y los eventos.

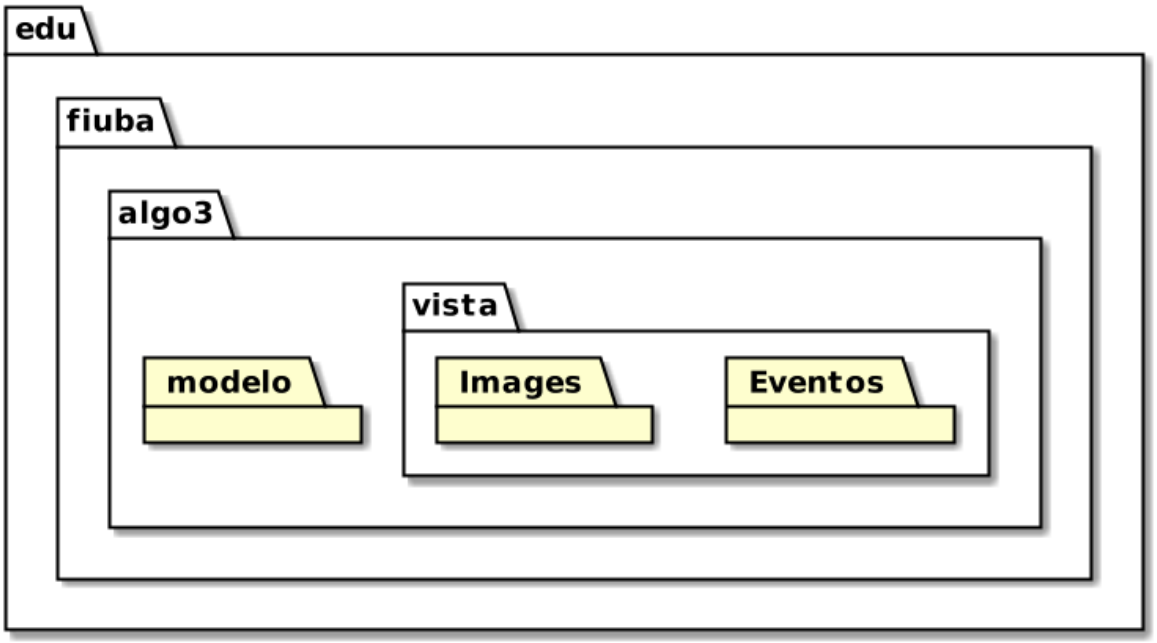

Figura 13: Diagrama de paquetes

Diagramas de estado

En el siguiente diagrama se muestran los posibles estados que puede tener el lápiz del personaje y cómo pueden variar entre ambos.

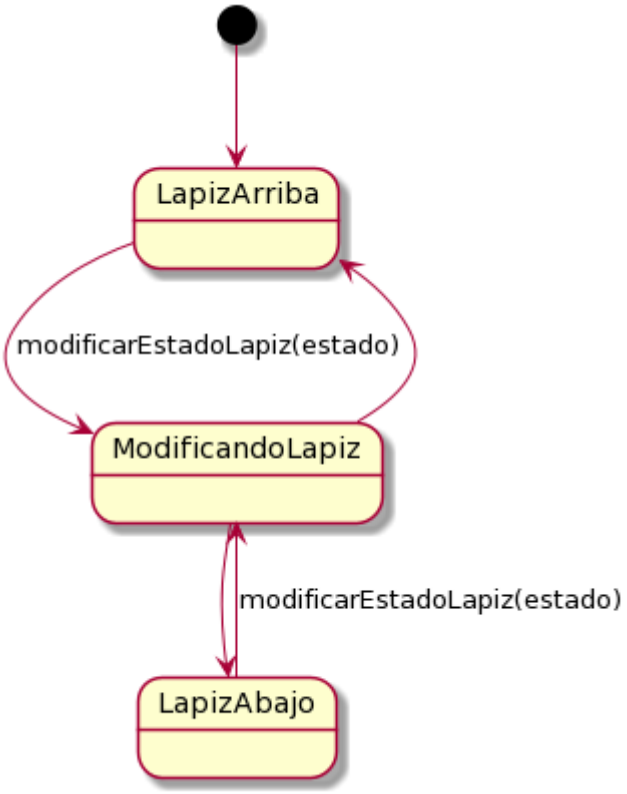

Figura 14 : Diagrama de Estado

Detalles de implementación

Se creó una clase Trazo para guardar los movimientos realizados por el personaje. Esta clase indica desde donde parte, hacia donde se mueve y el estado del lápiz durante ese movimiento para poder saber si el lápiz repercute o no en el dibujo.

Usamos la clase Dirección para poder indicar al personaje que tipo de movimiento (mover arriba, abajo, etc) debía realizar en pares coordenados (X,Y)

Para saber si una posición es válida y el personaje puede moverse hacia ella, utilizamos un método que toma la dimensión del tablero y verifica que la posición esté dentro de esta.

A diferencia de los bloques de movimiento y de cambio de estado del lápiz, los bloques de Repetir, Invertir y Algoritmo Personalizado guardan en sus atributos un objeto de tipo Algoritmo.

En los bloques de movimiento y de cambio de estado del lápiz, se utiliza el objeto DragAndDrop para agregarlos al algoritmo actual.

Por otro lado, los bloques Invertir, Repetir y Algoritmo Personalizado utilizan la dinámica de botones. Al pulsarlos, se crea un nuevo bloque que contiene una copia del algoritmo actual. A este nuevo bloque se le puede asignar un nombre para diferenciarlo y se convierte en un objeto de DragAndDrop.

Para la copia del algoritmo, creamos un método clonarAlgoritmo() en la clase Algoritmo. Este método realiza un ciclo for y llama cada bloque que tendrán un método clonarBloque() que devuelve un nuevo objeto con las mismas propiedades del bloque actual.

Cualquier bloque que se agregue al sector donde se arma el algoritmo, se agrega como un botón. Si se clickea se elimina del sector y del algoritmo.

Excepciones

Exception AlgoritmoPersonalizadoSinBloquesError. Esta excepción fue creada para el caso que se quiera crear un algoritmo personalizado y no hay bloques inicializados.

Exception AlgoritmoSinBloquesError. Esta excepción fue creada para el caso en que se quiera agregar un algoritmo a un bloque invertir o repetir y este no contenga bloques.

Exception BloqueSinNombreError. Esta excepción fue creada para el caso en que un usuario no establezca el nombre de un Algoritmo Personalizado, Repetir o Invertir.