FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 2
Segundo cuatrimestre de 2020

Grupo N7		
Alumno	**Padrón**	**Mail**
Alejo Villores	105285	avillores@fi.uba.ar
Alex Teper	105301	ateper@fi.uba.ar
Andreas Kuballa	103612	akuballa@fi.uba.ar
Kevin Meaca	102437	kmeaca@fi.uba.ar
Valentina Varela Rodríguez	105374	vvarela@fi.uba.ar

Corrector: Tomás Bustamante

Índice

1. Introducción

El presente informe reúne la documentación de la solución de la segunda entrega del segundo trabajo práctico de la materia Algoritmos y Programación III. Dicha solución consiste en diseñar y desarrollar una aplicación en Java que permita:

- Creación de un personaje que pueda levantar o bajar su lápiz, y que por defecto lo tenga levantado.
- Mover al personaje en todas las direcciones utilizando los bloques correspondientes.
- Creación del sector dibujo (en el modelo, sin interfaz gráfica)
- Mover al personaje con el pincel arriba y abajo, verificando que el sector dibujo quede dibujado según corresponda
- Creación de un algoritmo usando los bloques de repetición
- Creación de un algoritmo usando el bloque de invertir

2. Supuestos

- El personaje no puede moverse en diagonal
- No se puede ejecutar un algoritmo sin bloques
- No se puede ejecutar un bloque de repetición o inversión que no tenga bloques dentro

3. Diagramas de clase

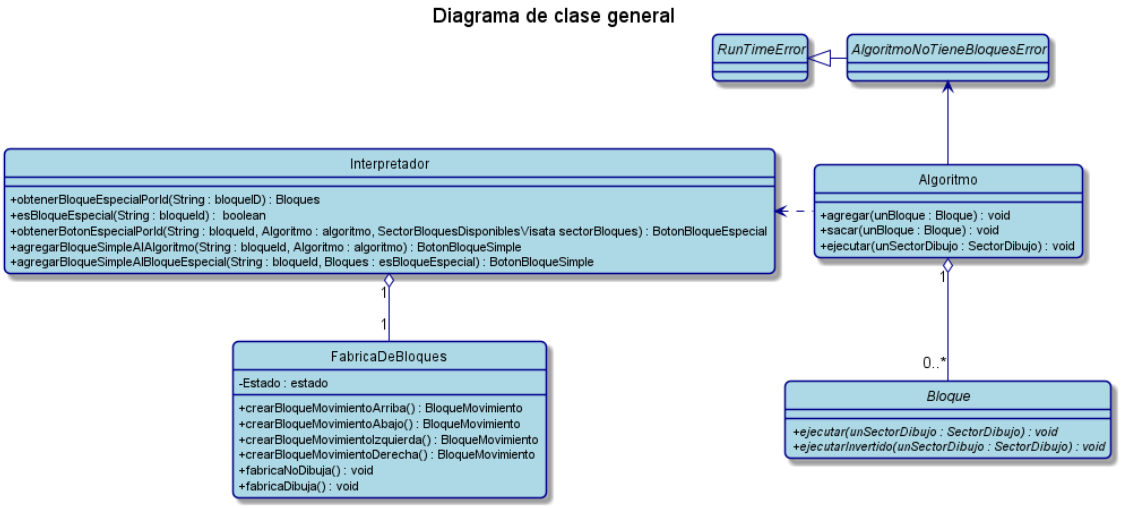

Figura 1: Diagrama de clase general

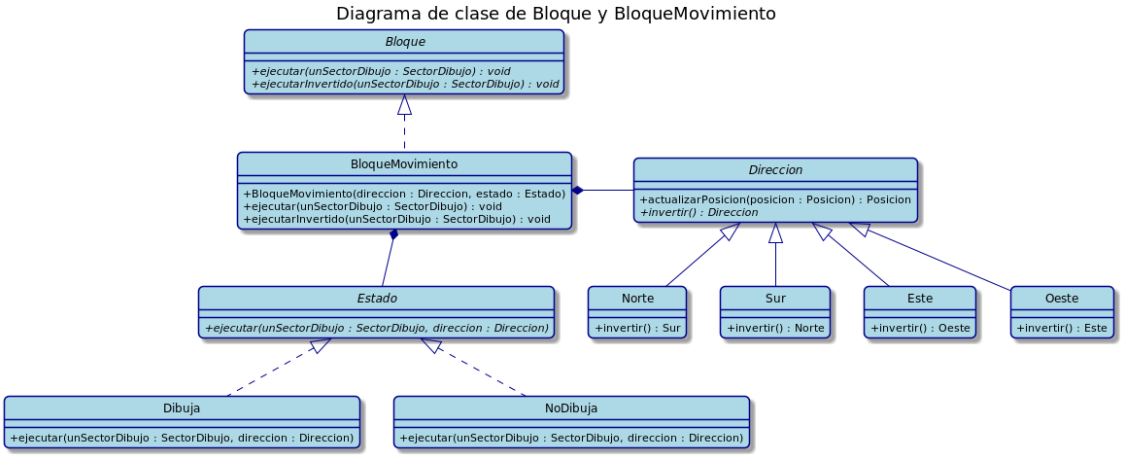

Figura 2: Diagrama de clase de los bloque de movimiento

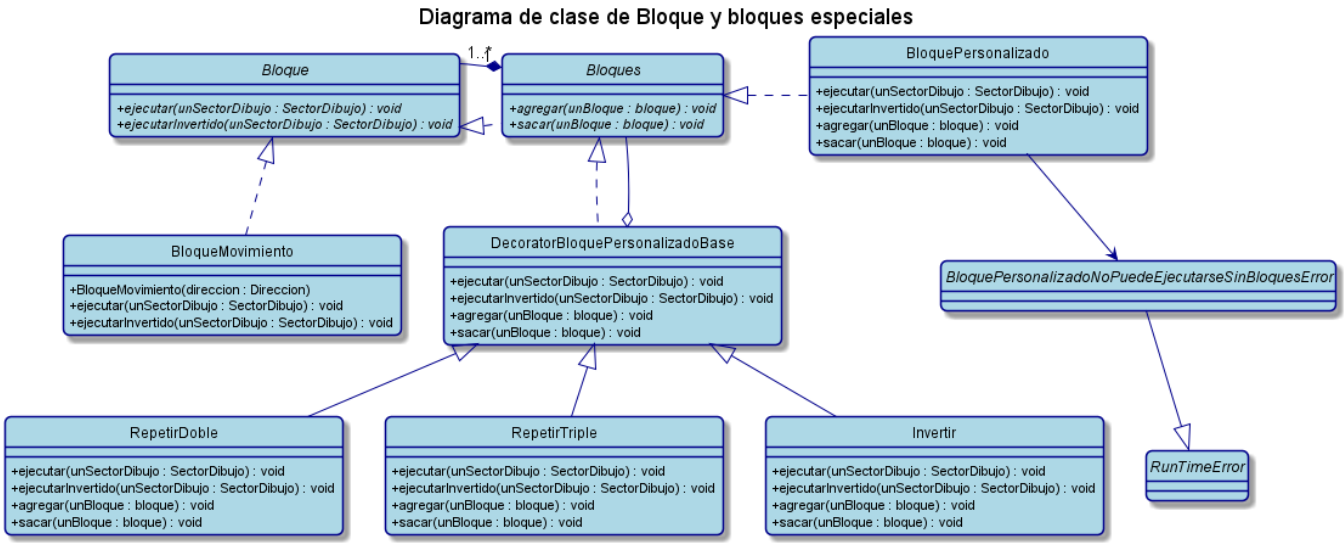

Figura 3: Diagrama de clase de los bloques especiales

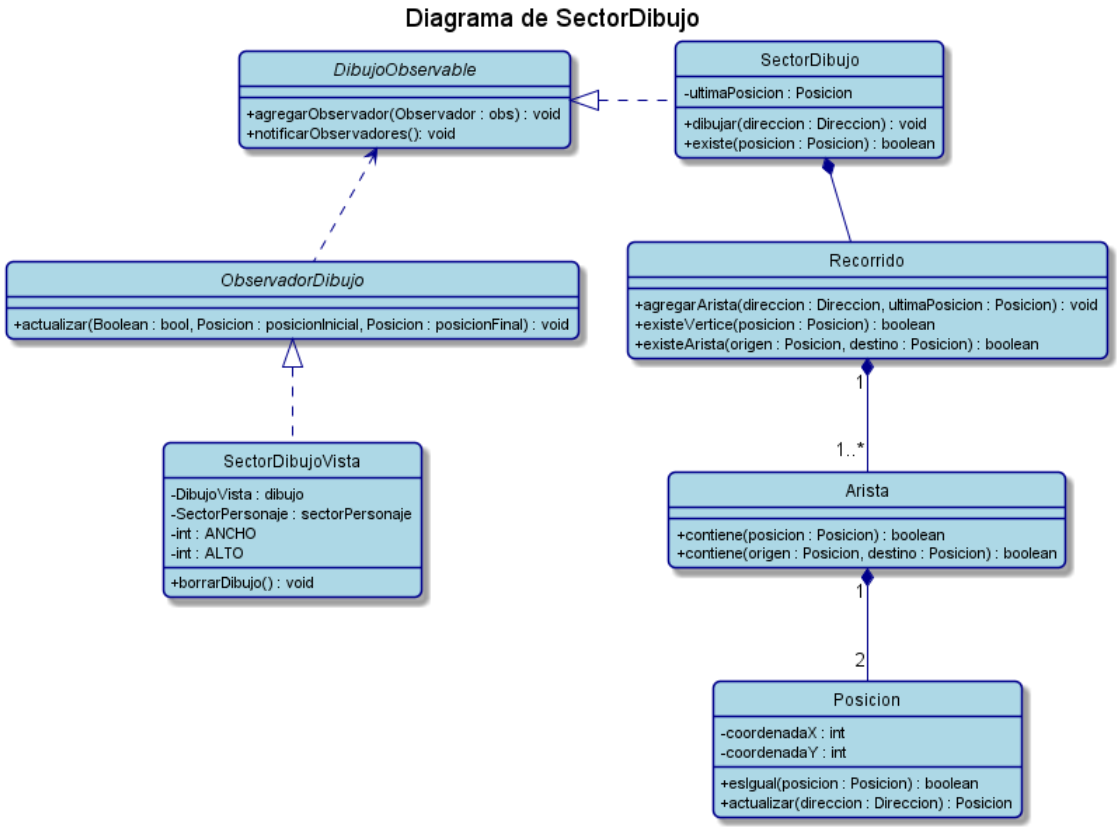

Figura 4: Diagrama de clase del sector dibujo y sus relaciones

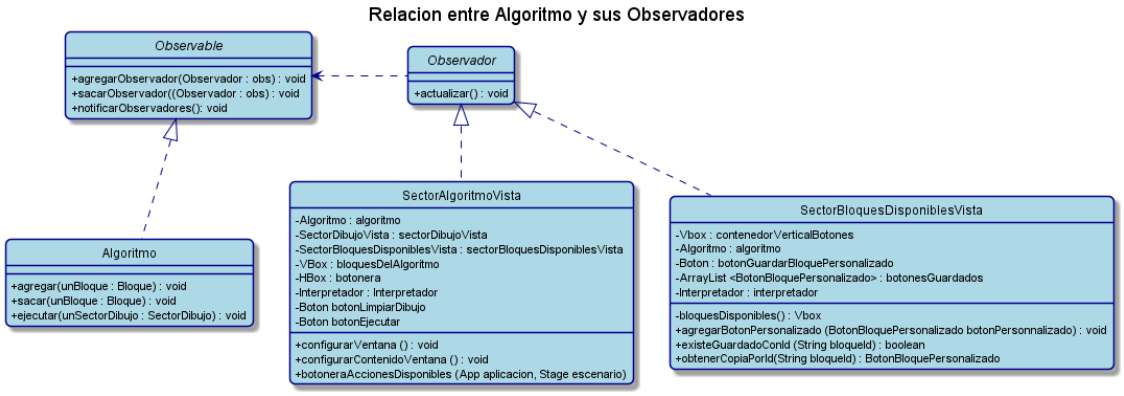

Figura 5: Diagrama de clase del algoritmo y sus relaciones

4. Diagrama de estado

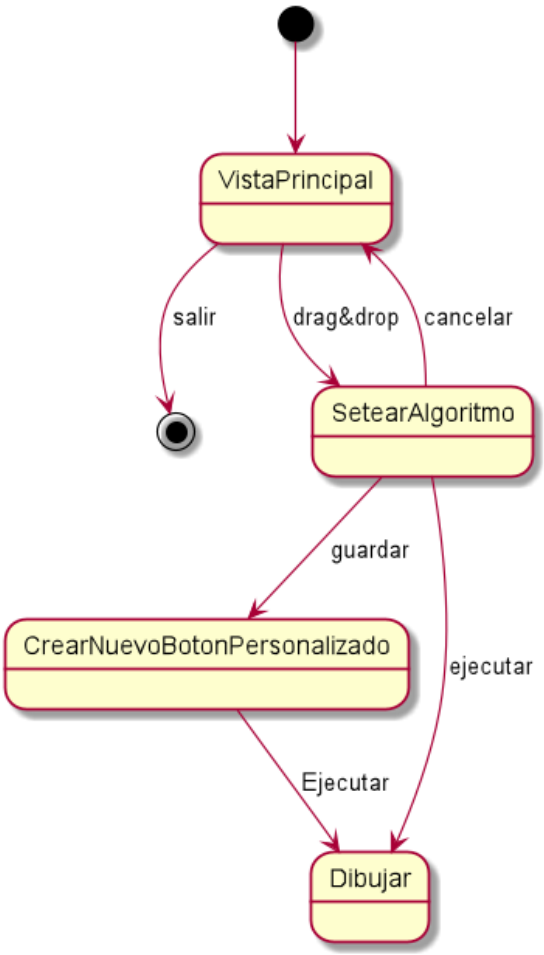

Figura 6: Diagrama de estados de la vista principal

5. Diagrama de paquetes

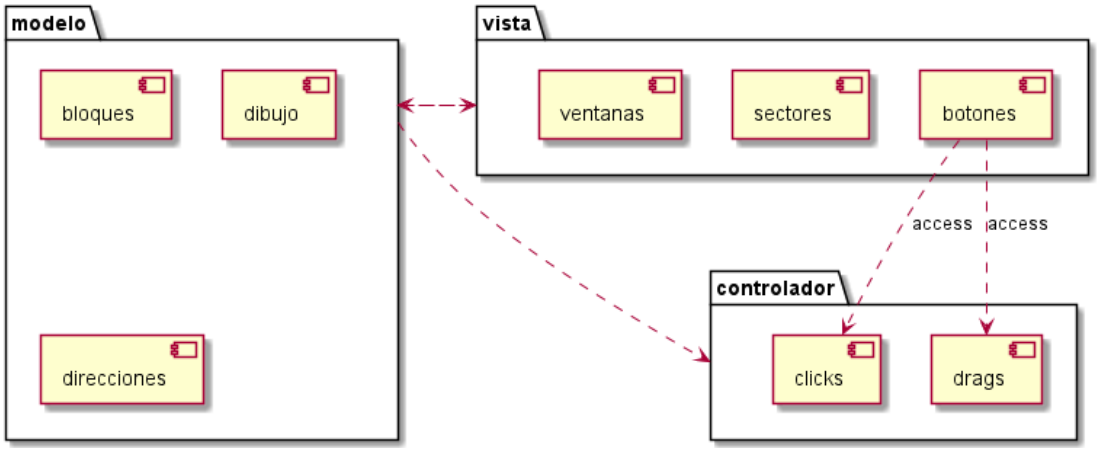

Figura 7: Diagrama de paquetes de AlgoBlocks

6. Detalles de implementación

6.1. Algoritmo

Es una de las clases con las cuales el cliente estará interactuando directamente. La responsabilidad de esta es ejecutar el conjunto de bloques que contiene. Para esto itera sobre los mismos y delega la ejecución a cada bloque.

6.2. Bloques especiales

Para los bloques de repetición, el bloque de invertir y el personalizado decidimos utilizar el patrón decorator. Para eso agregamos una interfaz Bloques que implementa Bloque y cuyo agregado es un método agregar(Bloque bloque). Tanto DecoratorBloquePersonalizadoBase como BloquePersonalizado implementan esta interfaz y BloquePersonalizado es quien guarda una colección de Bloque. Luego agregamos otros tres decoradores que heredan del base: RepeticiónDoble, RepeticiónTriple e Invertir.

6.3. Patrón Interpretetor mediante la clase Interpretador

En el momento que el usuario arrastra los diferentes botones disponibles y los traslada hacia la sección del algoritmo, se agregan los bloques correspondientes al algoritmo. Por el funcionamiento de arrastrar y trasladar (la acción drag and drop) solo es posible copiar objetos de tipo String y Image entre las diferentes secciones. Se usa dicho string (bloqueId) que contiene la información sobre el botón trasladado para que el interpretador pueda leerla y crear el botón y el bloque deseado.

6.4. Patrón Observer mediante las clases ObservadorDibujo y Observador

En el momento de ejecutar el algoritmo es necesario que tanto el algoritmo comunique su avance en la ejecución de los bloques a SectorAlgoritmoVista como también SectorDibujo debe comunicar su estado a SectorDibujoVista. Para ambas interacciones se utiliza el patrón observer para reducir el acoplamiento, fomentar la reusabilidad y separar el ámbito del modelo del ámbito de la vista.

6.5. Patrón Composite entre las clases BloqueMovimiento y Bloques

Dado que el algoritmo esta compuesto por objetos de clase BloqueMovimiento y de clase Bloques (un compuesto de otros objetos BloqueMovimiento o Bloques), que se podría comparar a un árbol con hojas (BloqueMovimiento) y subárboles (Bloques), se creo la interfaz Bloque la cual sera implementada por ambas clases para delegar la ejecución de cada bloque a los objetos diferentes. De esa manera se evita recorrer recursivamente todos los elementos del algoritmo y se evita una implementación compleja de esta estructura de árbol.

7. Excepciones

AlgoritmoNoPuedeSerEjecutadoSiNoTieneBloquesError Esta excepción la lanza una instancia de Algoritmo cuando se intenta ejecutar un algoritmo que no posee ningún bloque

BloquePersonalizadoNoPuedeSerEjecutadoSiNoTieneBloquesError Esta excepción la lanza una instancia de BloquePersonalizado cuando se intenta ejecutar sin ningún bloque

8. Diagramas de secuencia

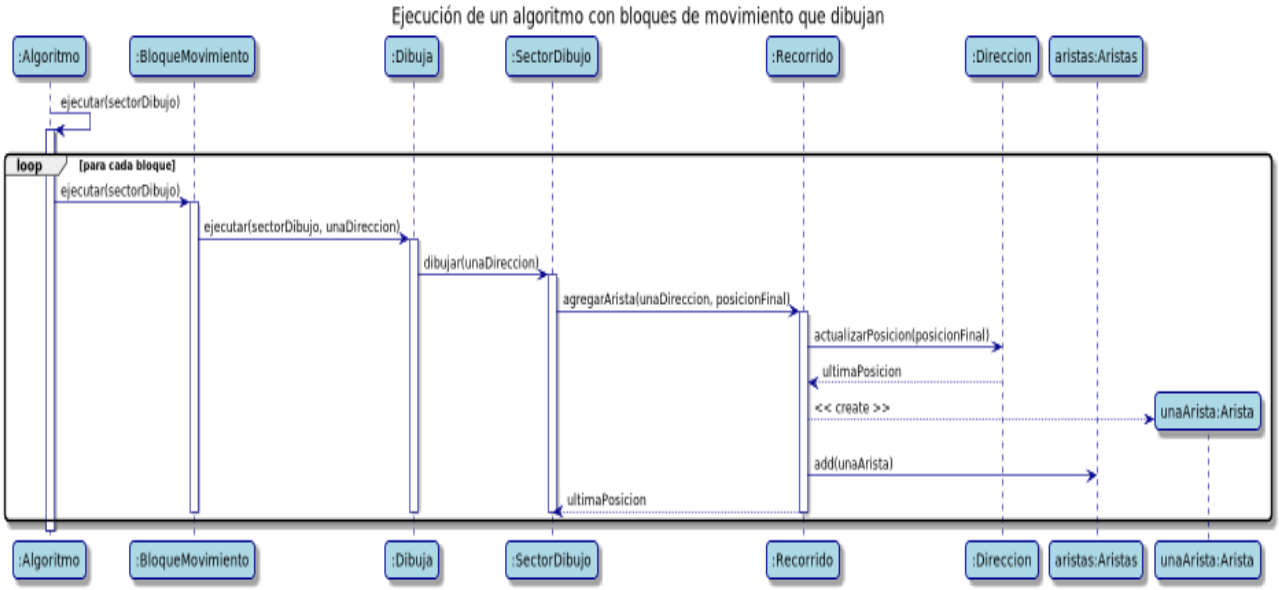

Figura 8: Ejecución de un algoritmo cuyos bloques de movimiento desconocemos pero que realiza un dibujo

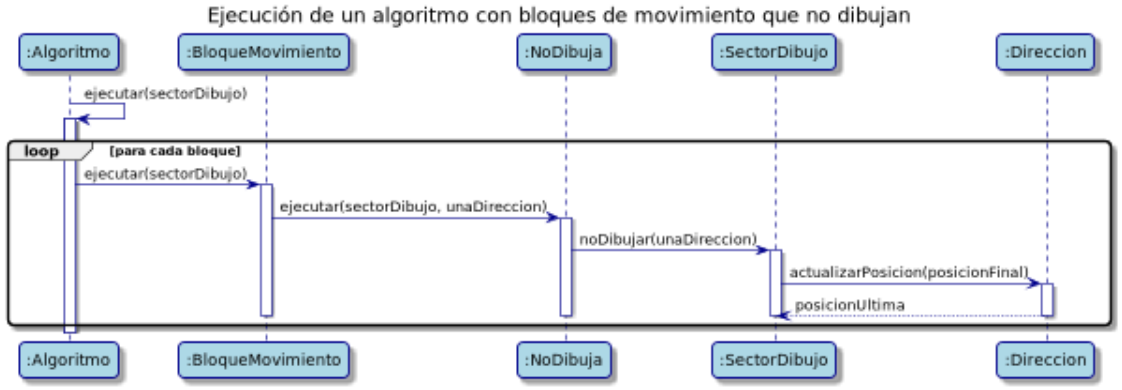

Figura 9: Ejecución de un algoritmo cuyos bloques de movimiento desconocemos pero que no realizan un dibujo

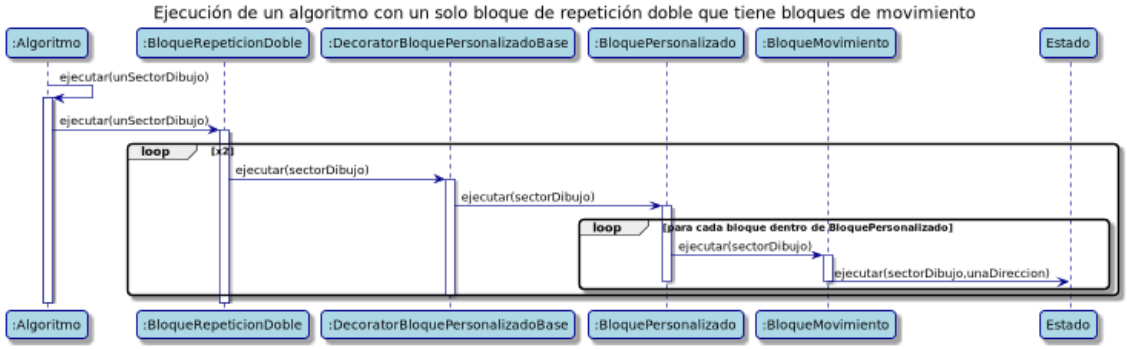

Figura 10: Ejecución de un algoritmo que tiene un bloque de repetición doble, que adentro tiene bloques de movimiento

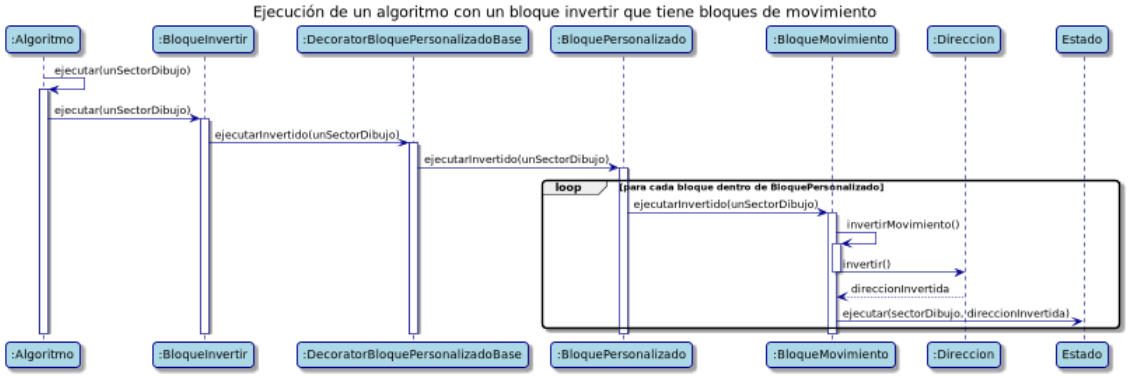

Figura 11: Ejecución de un algoritmo que tiene un bloque de invertir, que adentro tiene bloques de movimiento

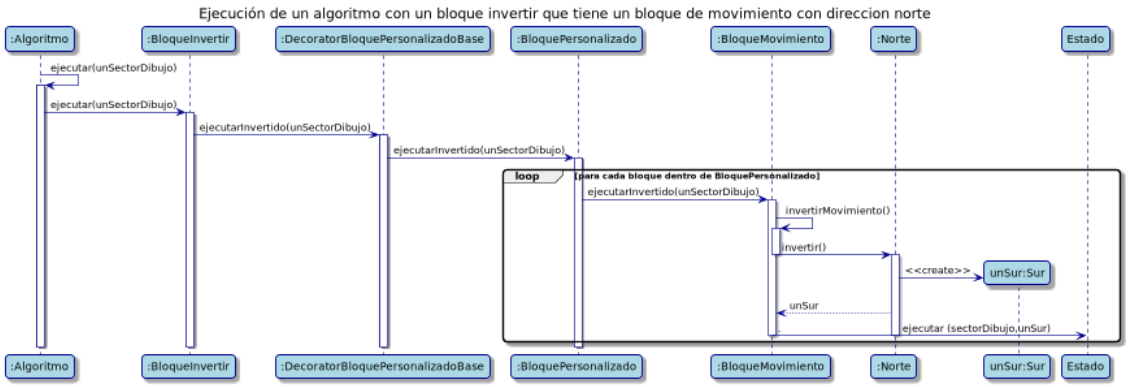

Figura 12: Ejecución invertida de un algoritmo que tiene un bloque de movimiento con dirección Norte