

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III

Curso 1

Segundo cuatrimestre de 2020

Grupo T10		
Nombre	Padrón	Mail
Bergman, Guido	104030	gbergman@fi.uba.ar
Fábregas, Camilo Ezequiel	103740	cfabregas@fi.uba.ar
Delgado, Nahuel	104078	ndelgado@fi.uba.ar
de Montmollin, Magdalena	99331	mdemontmollin@fi.uba.ar

Índice

1. Introducción	2
2. Supuestos	2
3. Diagramas de clase	2
4. Diagramas de secuencia	4
5. Diagramas de paquetes	7
6. Diagramas de estado	7
7. Detalles de implementación	7
7.1. Condición del lápiz y su relación con el dibujo	7
7.2. Ejecución del Algoritmo	7
7.3. Contenedor de Bloques	7
7.4. Bloque Personalizado	7
7.5. Detalles de la Interfaz	7
8. Excepciones	8

1. Introducción

El presente informe reúne la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación, de manera grupal, que permite aprender los conceptos básicos de la programación de manera didáctica y visual, en Java utilizando la programación orientada a objetos y las técnicas de TDD e integración continua. La interfaz visual fue realizada en JavaFX.

2. Supuestos

Se supuso que el personaje no tiene restricciones espaciales para moverse, es decir, puede moverse en cualquier dirección tantas veces quiera.

Un lápiz puede estar arriba o abajo, no tiene estados intermedios.

Se puede subir o bajar el lápiz 2 veces. La segunda no produce cambios.

Suponemos que al invertir un bloque repetición, los bloques interiores se recorren en el orden inverso y se invierte cada uno de ellos.

Al invertir un algoritmo personalizado, se lo recorre secuencialmente invirtiendo cada uno de los bloques que contiene.

Un bloque personalizado puede llamarse como desee el usuario, toda combinación de teclas alfanumérica es válida. Además, que el usuario guarde más de un algoritmo con el mismo nombre no se considera un problema.

3. Diagramas de clase

El programa se separó en dos diagramas de clases para facilitar la lectura de ellos. En las figuras 1 y 2 se pueden observar los diagramas mencionados.

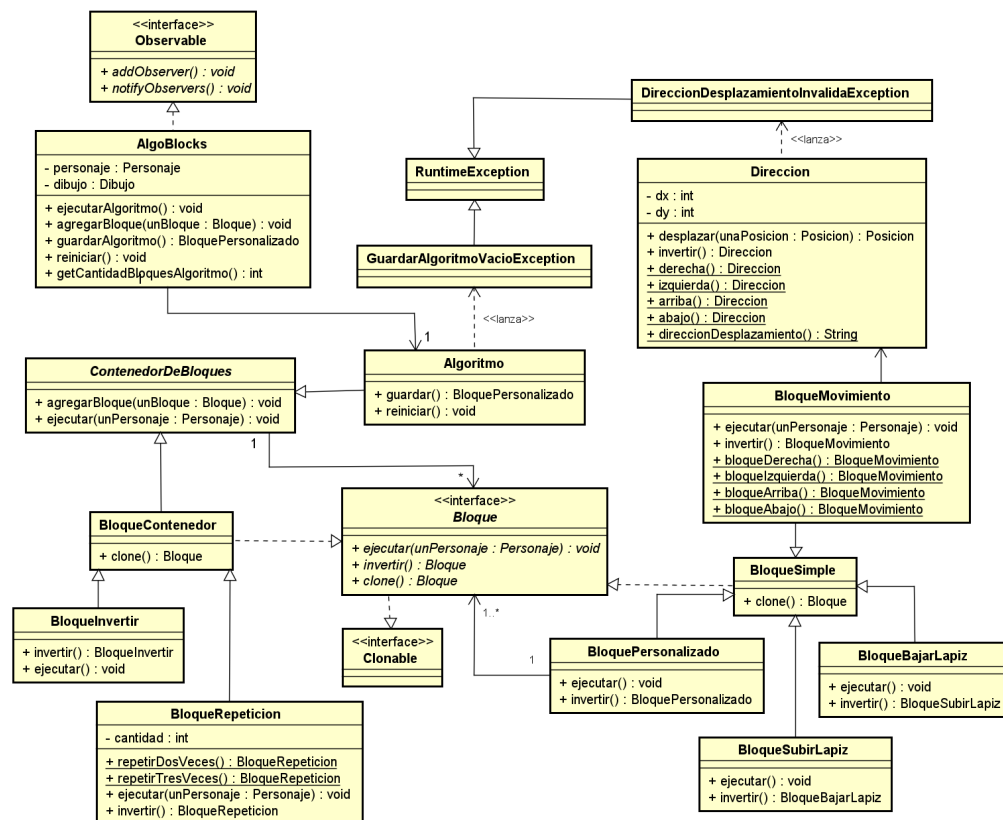


Figura 1: Diagrama de Clases Algoritmo

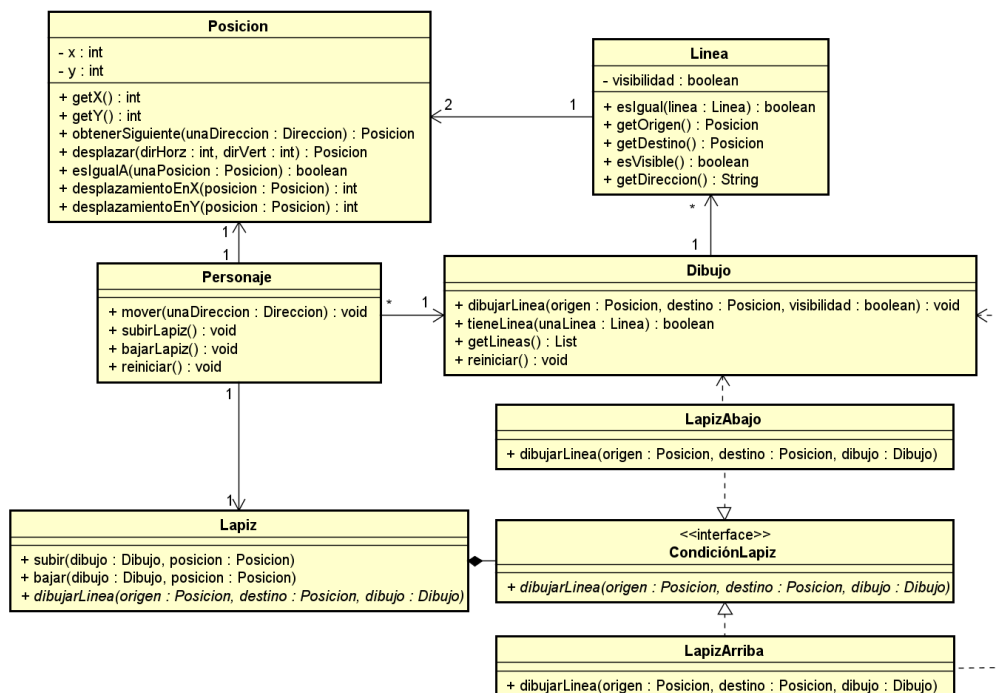


Figura 2: Diagrama de Clases Personaje

4. Diagramas de secuencia

Los diagramas de secuencia están dirigidos a la ejecución de bloques y al manejo de bloques compuestos.

El primer diagrama representa ejecutar un algoritmo con solo un bloque de movimiento hacia la derecha, debido a que el personaje comienza con lápiz abajo, también se dibuja la línea que ocasiona dicho movimiento. El diagrama se divide en dos, la primera parte muestra el desplazamiento del personaje y la segunda muestra como se dibuja la línea.

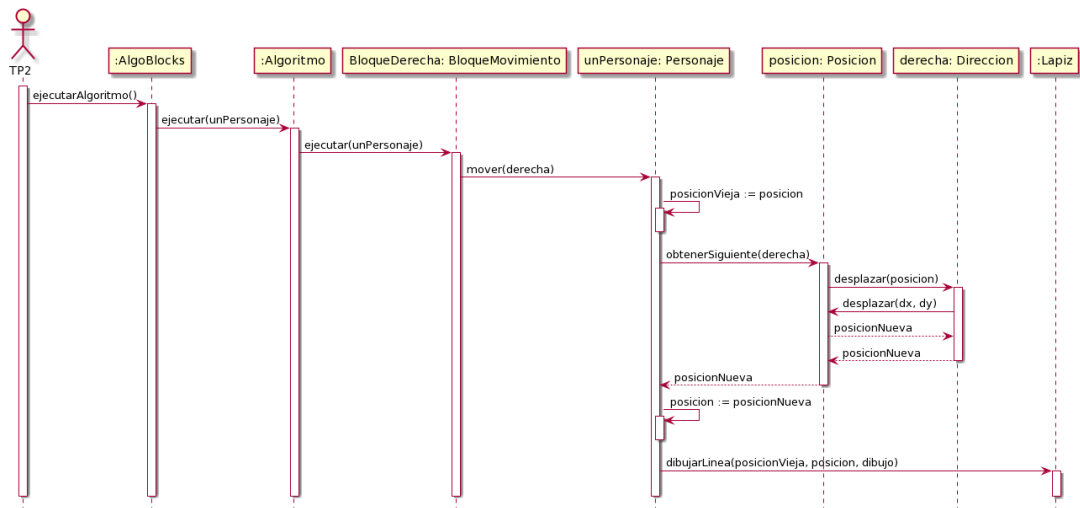


Figura 3: Diagrama de Secuencia ejecutar un bloque movimiento a la derecha con lápiz abajo (parte 1).

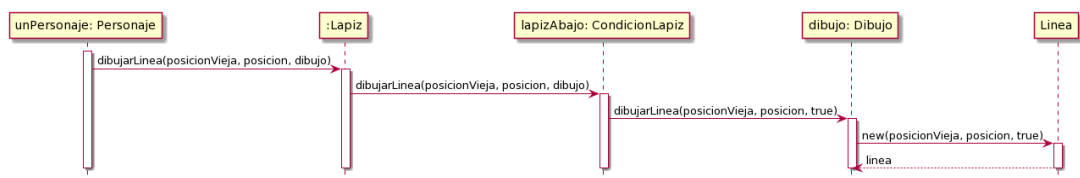


Figura 4: Diagrama de Secuencia ejecutar un bloque movimiento a la derecha con lápiz abajo (parte 2).

El siguiente diagrama muestra como se ejecuta un bloque repetición de tres veces con bloques dentro.

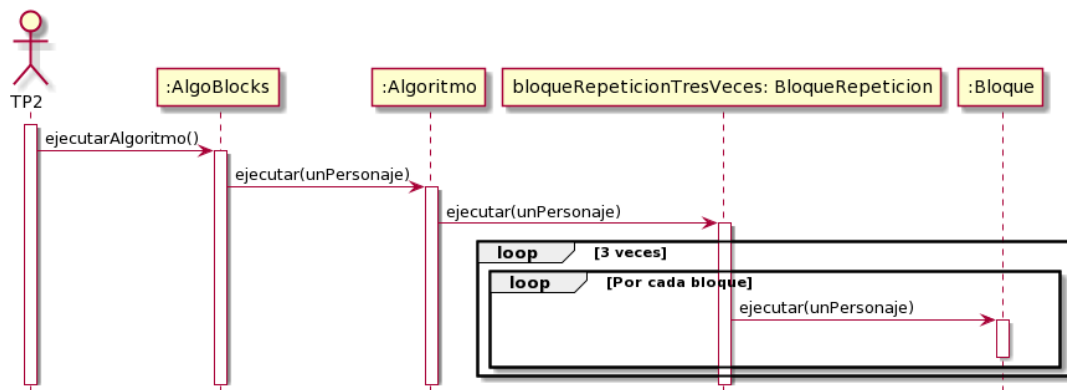


Figura 5: Diagrama de Secuencia ejecutar un bloque repetición de tres veces.

El diagrama de secuencia de guardar un algoritmo personalizado muestra como se crea una copia de los bloques agregados hasta el momento y los almacena en un nuevo bloque.

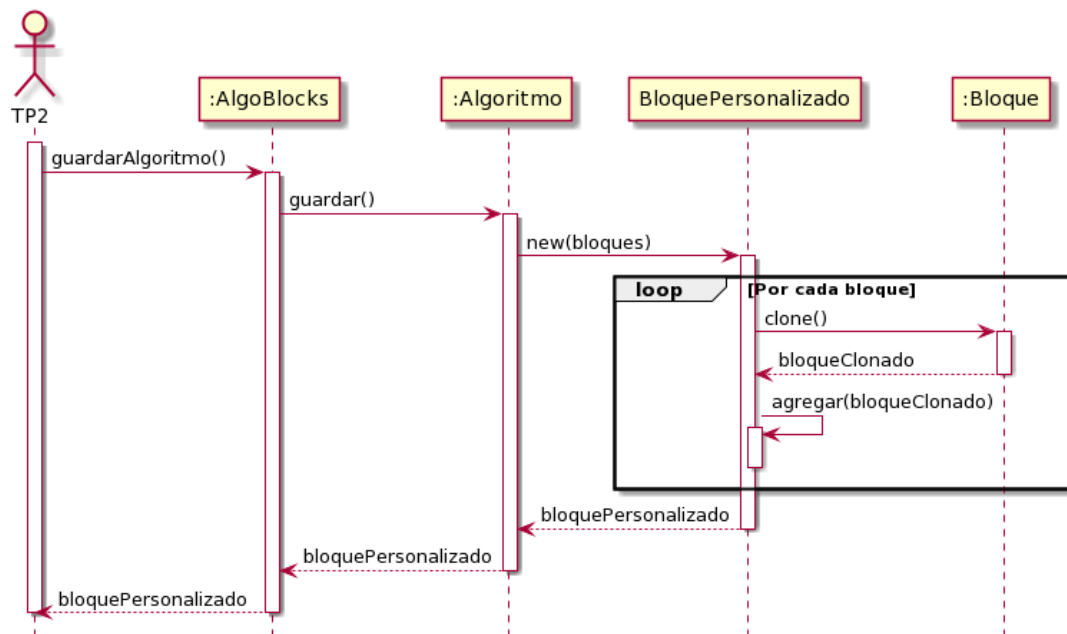


Figura 6: Diagrama de Secuencia guardar algoritmo personalizado.

El diagrama de secuencia de ejecutar un bloque inversión muestra como se invierte cada bloque y luego se ejecuta su resultado.

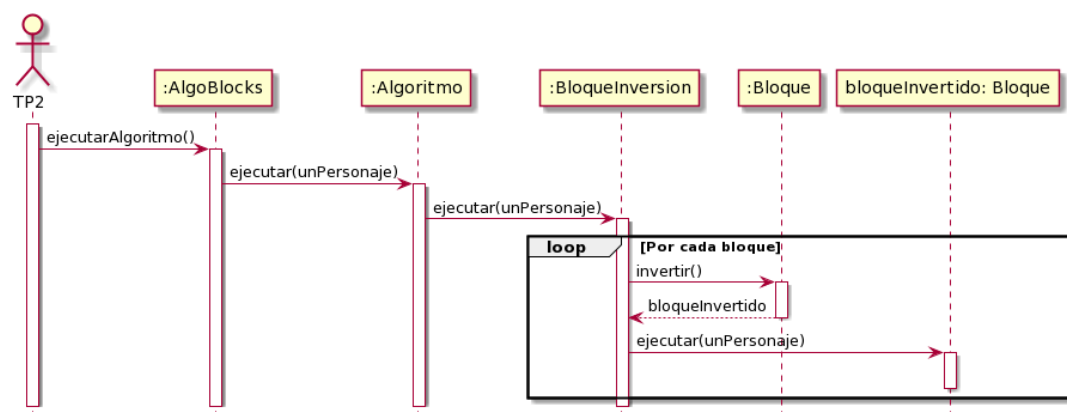


Figura 7: Diagrama de Secuencia ejecutar bloque inversión.

Los siguientes diagramas hacen referencia a casos que pueden ocurrir como iteraciones del loop del diagrama de secuencia anterior. El primer diagrama muestra el caso de invertir un bloque repetición, que como resultado invertiría el orden y cada bloque dentro del mismo. El segundo hace referencia a invertir un bloque inversión.

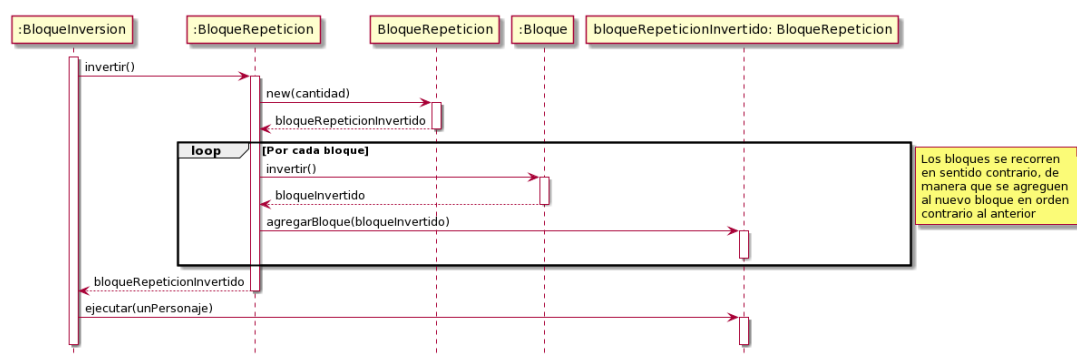


Figura 8: Diagrama de Secuencia invertir un bloque repetición.

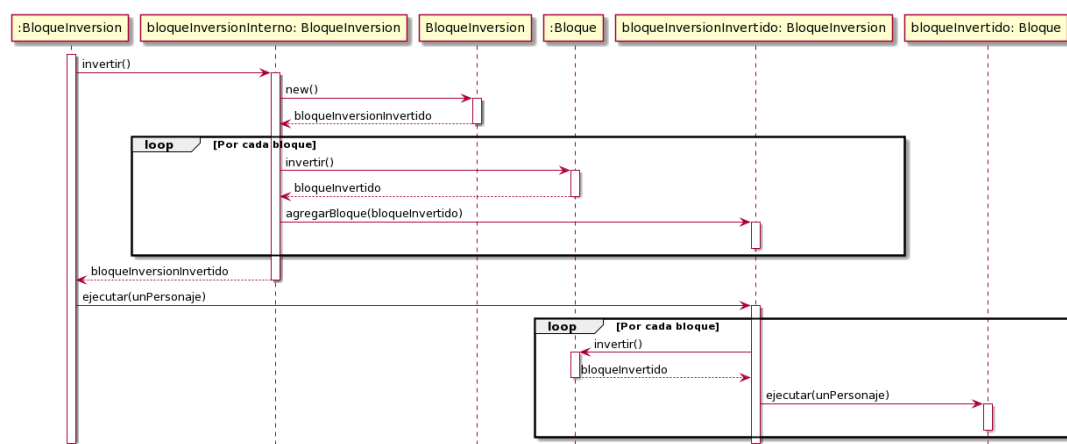


Figura 9: Diagrama de Secuencia invertir un bloque inversión.

5. Diagramas de paquetes

6. Diagramas de estado

7. Detalles de implementación

A la hora de implementar el modelo se plantearon alternativas sobre como debería comportarse el programa ante ciertas situaciones. En esta sección se mencionan las más relevantes.

7.1. Condición del lápiz y su relación con el dibujo

No se considera un problema que un **Lapiz** reciba el mensaje `bajar()` si su condición es **LapizAbajo**. Similarmente, no es un problema que reciba `subir()` si su condición es **LapizArriba**. Además, de llamar al método `dibujarLinea(origen: Posicion, destino: Posicion, dibujo: Dibujo)` se crea una instancia de la clase **Linea** en el objeto `dibujo` cuya visibilidad depende de la condición del lápiz. Si esta arriba la línea creada es invisible, caso contrario, es visible.

Al cambiar la condición de un **Lapiz** se dibuja, en el dibujo, una línea con una misma posición como origen y destino, cuya visibilidad depende de la nueva condición del lápiz. Esta es consistente a lo comentado en el párrafo anterior.

7.2. Ejecución del Algoritmo

En **AlgoBlocks** tras ejecutar el **Algoritmo**, este mantiene su secuencia de Bloques, no los elimina.

7.3. Contenedor de Bloques

Se implementó una clase abstracta llamada *ContenedorDeBloques* de la cual heredan las clases **Algoritmo**, **BloqueInvertir** y **BloqueRepeticion** para evitar la repetición de código al encontrar comportamientos similares en ellas.

7.4. Bloque Personalizado

Para guardar el algoritmo actual en una instancia de **BloquePersonalizado** se decidió implementar, para los bloques, la función `clone()`, la cual devuelve una copia del bloque. Es decir, el un **BloquePersonalizado** almacena copias de los bloques que hay, al momento de llamar al método `guardarAlgoritmo` en el algoritmo. Esto fue necesario para evitar que de agregarse bloques a un **BloqueContenedor** previamente almacenado en un **BloquePersonalizado**, éste se vea modificado.

7.5. Detalles de la Interfaz

También se tomaron decisiones a la hora de implementar la aplicación con la que interactúa el usuario. Las más importantes se comentan a continuación.

La animación del algoritmo ingresado por el usuario siempre realizará los movimientos indicados en él. Incluso cuando esto genere que el personaje salga de los límites visibles de su región.

Al usuario no se le permite ejecutar dos veces el mismo algoritmo. De todos modos tras ejecutarlo, sigue teniendo la opción de guardarlo como un nuevo bloque.

8. Excepciones

- **DireccionDesplazamientoInvalidaException** Surge al solicitar la dirección de un desplazamiento, que tiene valores no nulos en ambos ejes.
- **GuardarAlgoritmoVacioException** Surge al querer guardar un algoritmo personalizado cuando no hay bloques.