



FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Trabajo Práctico 2 — Java

AlgoBlocks - JavaFX

Algoritmos y Programación III - [7507/9502]

Grupo 13 - Curso 2

Segundo cuatrimestre de 2020

Alumno

Mail

Apaldetti, Tomás

tapaldetti@fi.uba.ar

Kelman Axel

akelman@fi.uba.ar

Perticaro Agustin

aperticaro@fi.uba.ar

Gazzola Franco

fgazzola@fi.uba.ar



Índice

1. Introducción	1
2. Supuestos	2
3. Diagramas de clase	3
4. Detalles de implementación	8
5. Diagramas de secuencia	9
6. Diagrama de Paquetes	11



1. Introducción

El presente informe reúne la documentación del progreso del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación que consiste en un juego que permite aprender los conceptos básicos de programación armando algoritmos utilizando bloques visuales.



2. Supuestos

- Al eliminar un bloque del algoritmo que se esta armando, este elimina todos los bloques subsiguientes.
- Un “algoritmo” puede tener infinitos niveles de bloques compuestos (llámese bloque repetir, bloque invertir, entre otros).
- Un “algoritmo” guardado no puede ser modificado una vez se ha guardado.
- Un bloque de repetición repite el algoritmo contenido como si se tratase de un “for loop”.
- Al moverse de una posición con el lápiz apoyado, solo se pintara la posición que se abandono.
- Todos los movimientos que realiza un personaje son consecutivos; no “salta” entre posiciones no adyacentes entre si.

3. Diagramas de clase

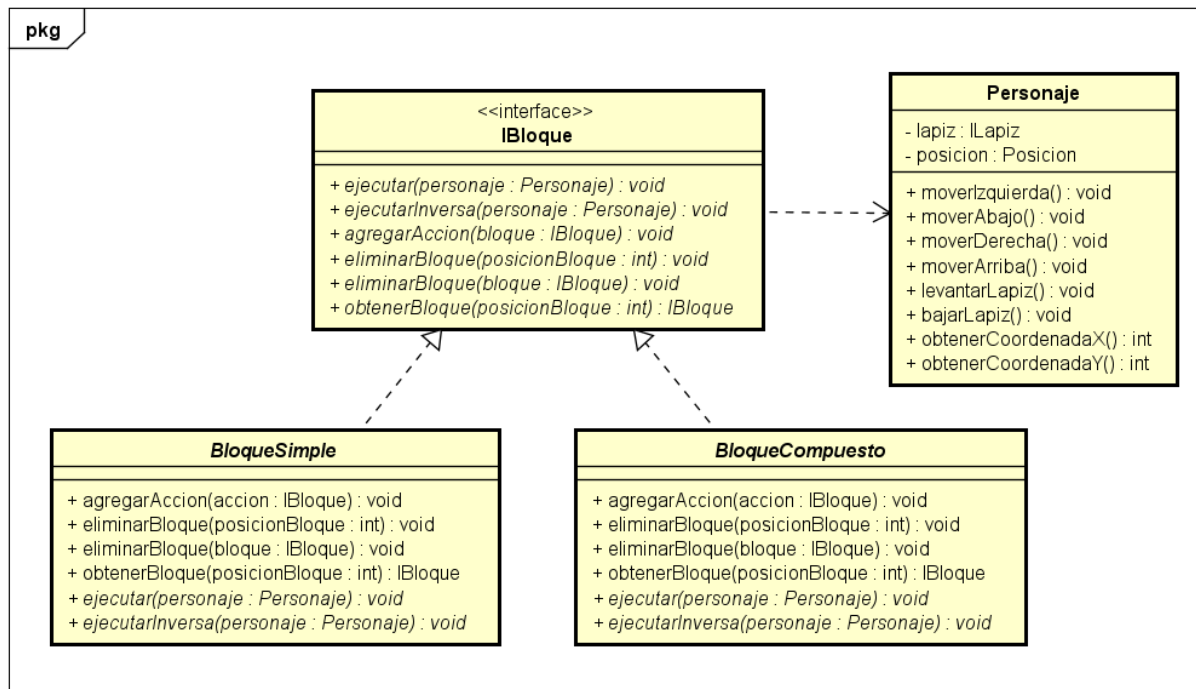


Figura 1: Diagrama Estructura de Bloques

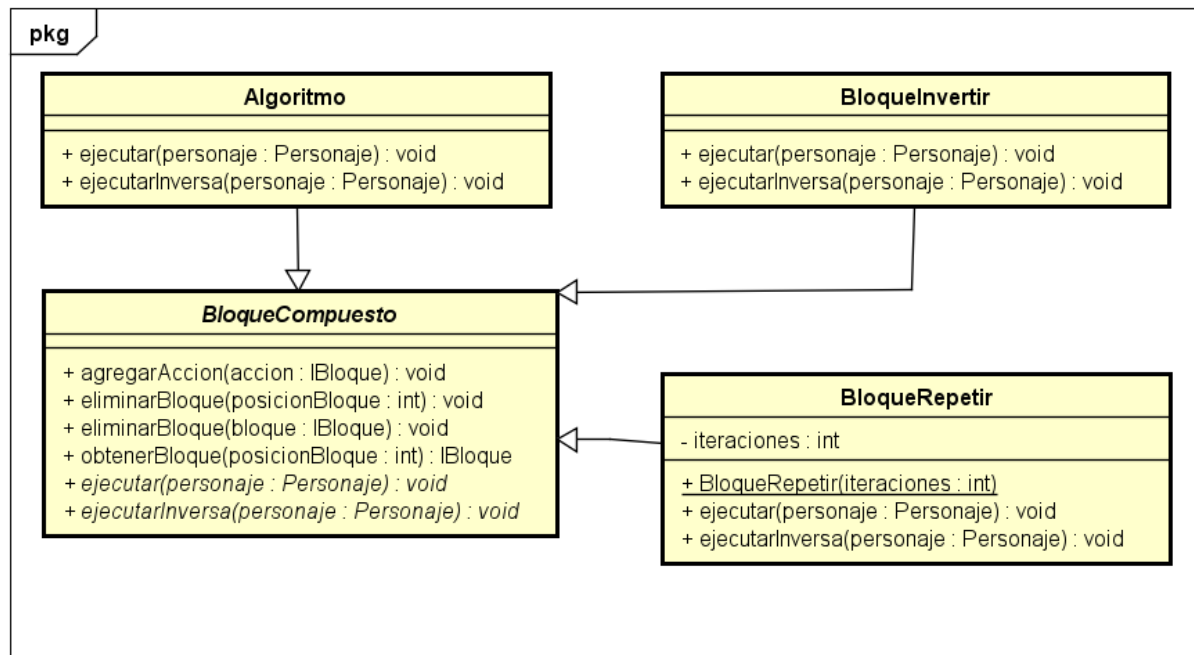


Figura 2: Bloques Compuestos disponibles en el modelo.

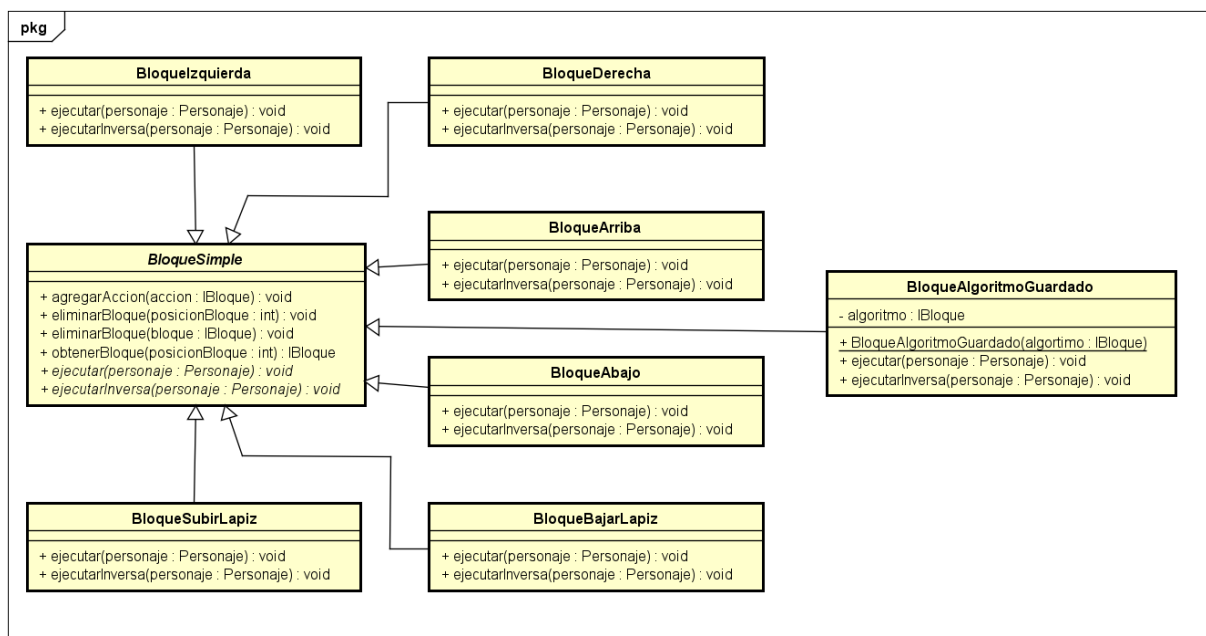


Figura 3: Bloques Simples disponibles en el modelo.

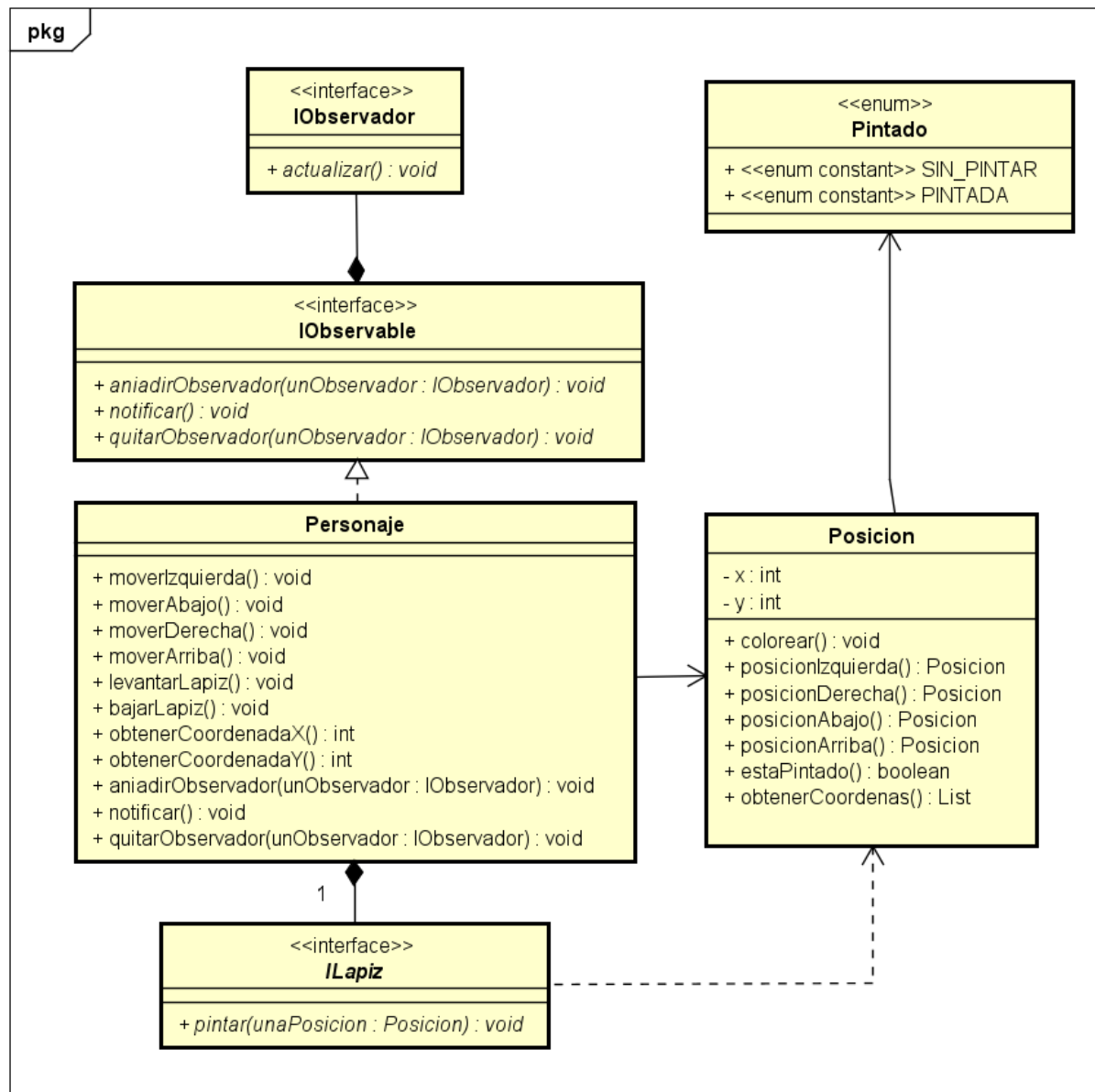


Figura 4: Relación Personaje con Posición. Un personaje es un observable.

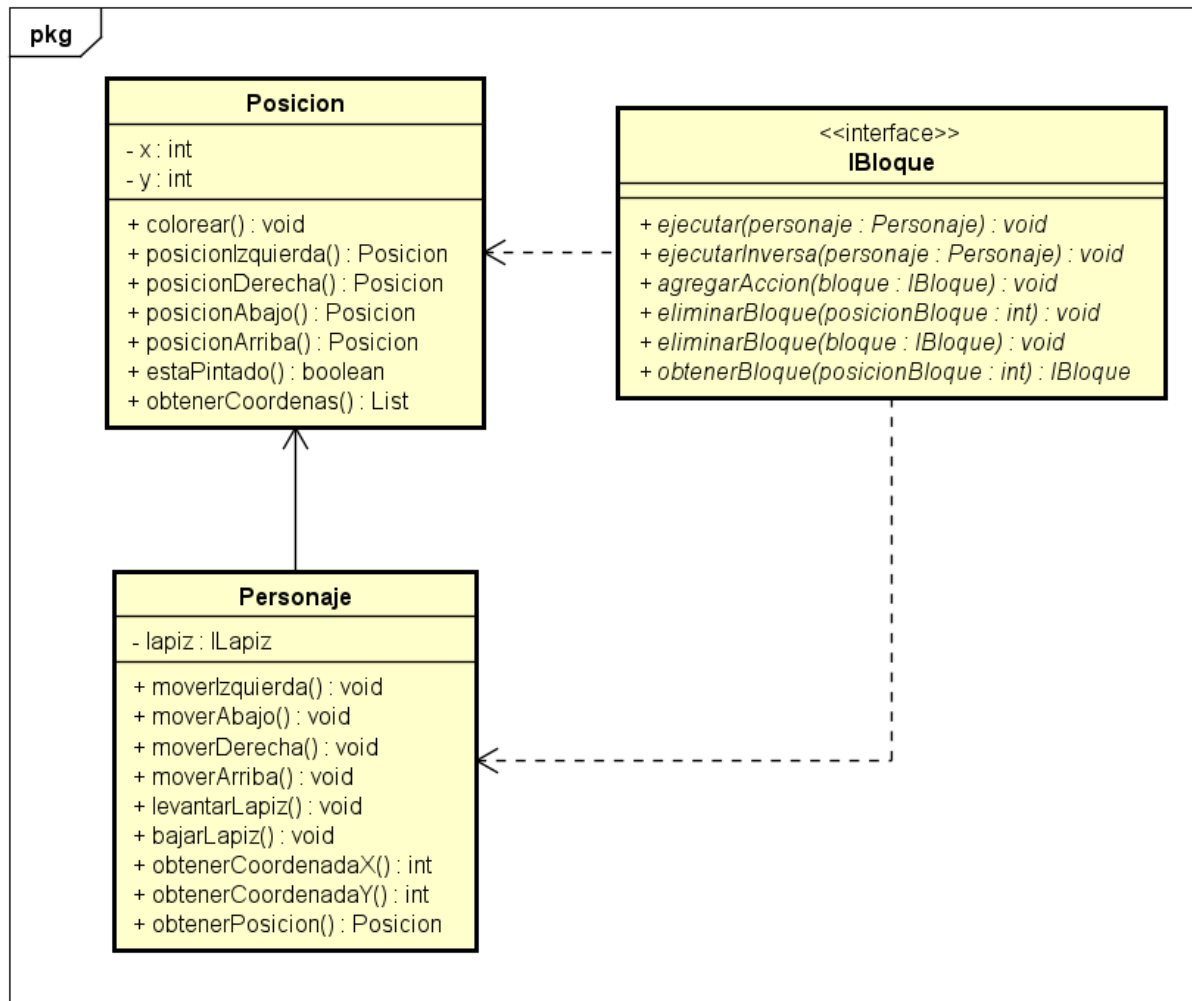


Figura 5: Relación entre personaje, bloque y posición.

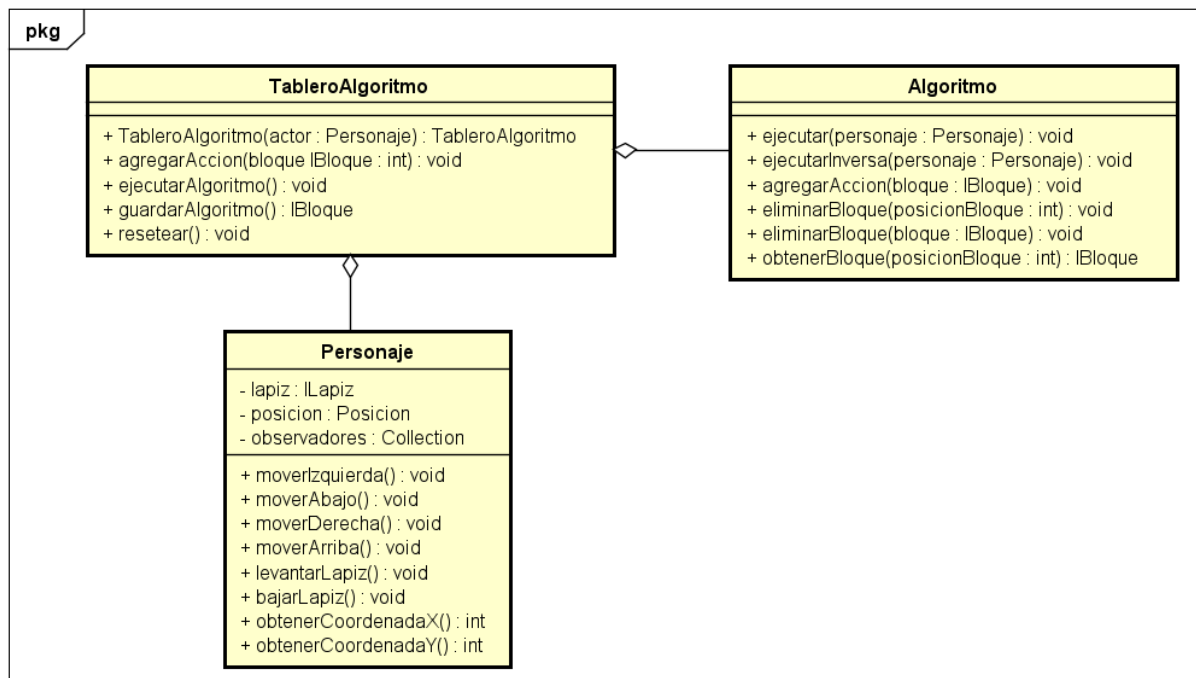


Figura 6: Tablero Algoritmo como interfaz al usuario.



4. Detalles de implementación

Para la implementación del modelo se propuso una clase “principal” **Personaje** que modela el objeto a mover mediante el algoritmo armado por bloques.

La posición de un personaje esta modela mediante la clase **Posición**. Esta clase se encarga de saber unas coordenadas y conseguir la siguiente posición según se pida. Cuando un personaje cambia de posición este puede pintar, o no, la posición en la que se encontraba; esto dependerá del estado del **Lápiz** al momento de moverse, pudiendo estar levantado o apoyado, pintando, o no, la posición abandonada según corresponda.

Las ordenes de movimiento, así como la de cambiar el estado del lápiz, están dadas por los **Bloques**. Estos conocen al personaje, y le dicen las acciones que debe realizar mediante el uso de la interfaz publica del personaje. Si bien todos los grupos cumplen con la misma interfaz, estos están en divididos en dos grupos: los **Bloques Simples**, y los **Bloques Compuestos**.

Los bloques simples son aquellos que por su comportamiento, no contienen otros bloques internos, es decir son ordenes simples.

Los bloques compuestos son aquellos que pueden, y para tener comportamiento deben, tener otros bloques, para darle ordenes con mas complejidad al personaje.

Si bien ambas responden a la misma interfaz de Bloque, los bloques simples no responden a los comandos que intentar modificar el bloque, estos métodos lanzan excepciones. Esto es necesario para que el patrón de diseño usado **Composite**, sea posible utilizar sin perder la posibilidad de modificar los bloques compuestos mas internos de un algoritmo.

Para notificar un cambio de posición a cualquier objeto interesado, un Personaje implementa la interfaz de **Observer**, notificando a cualquier objeto que este suscrito cuando este cambia de posición. Esto se utilizo luego para poder dibujar el personaje en su perteneciente tablero en la vista. Un **TableroAlgoritmo** es el responsable del manejo de un Algoritmo principal(armar, ejecutar, guardar) y entregárselo al personaje correspondiente a ejecutar dicho algoritmo.

5. Diagramas de secuencia

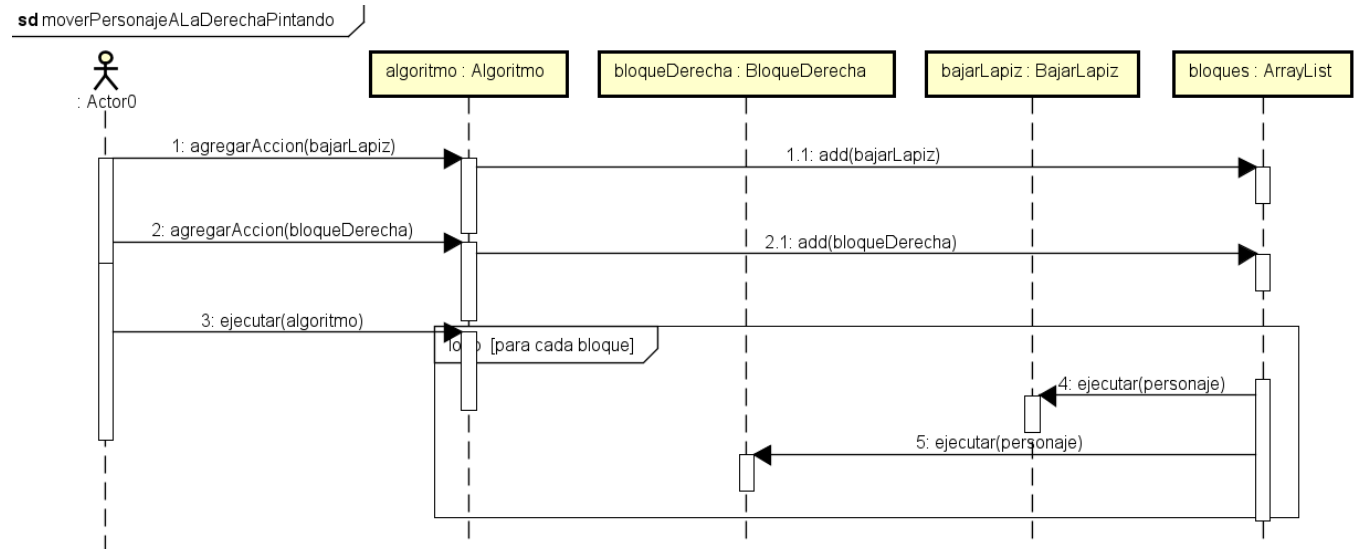


Figura 7: Mover el personaje a la derecha pintando.



sd moverPersonajeAlternandoLapiz

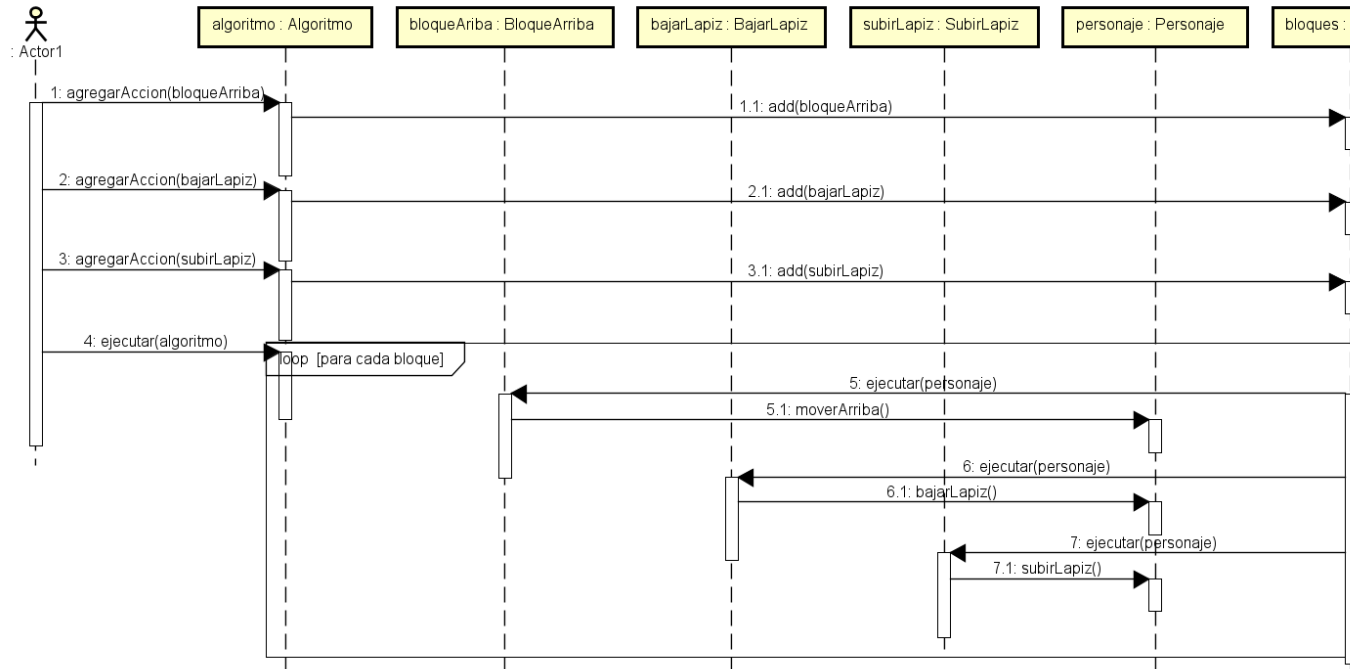


Figura 8: Mover el personaje subiendo y bajando el lápiz.

sd moverPersonajeUsandoBloqueDeRepeticion

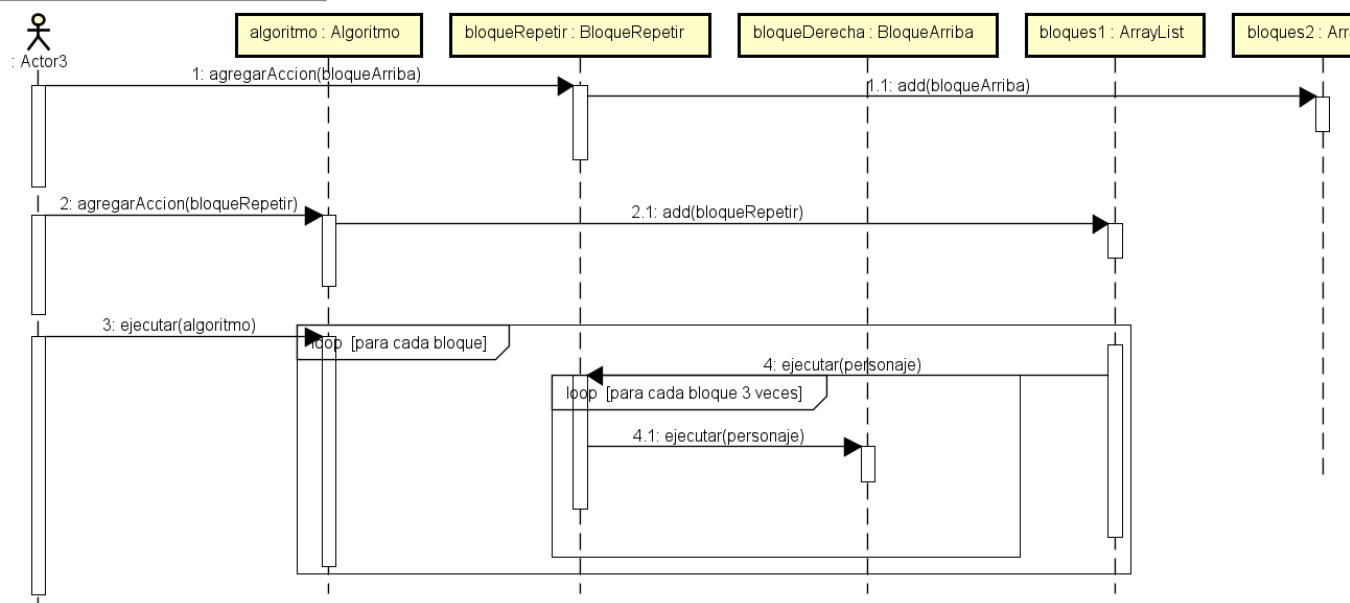


Figura 9: Mover el personaje subiendo y bajando el lápiz.

6. Diagrama de Paquetes

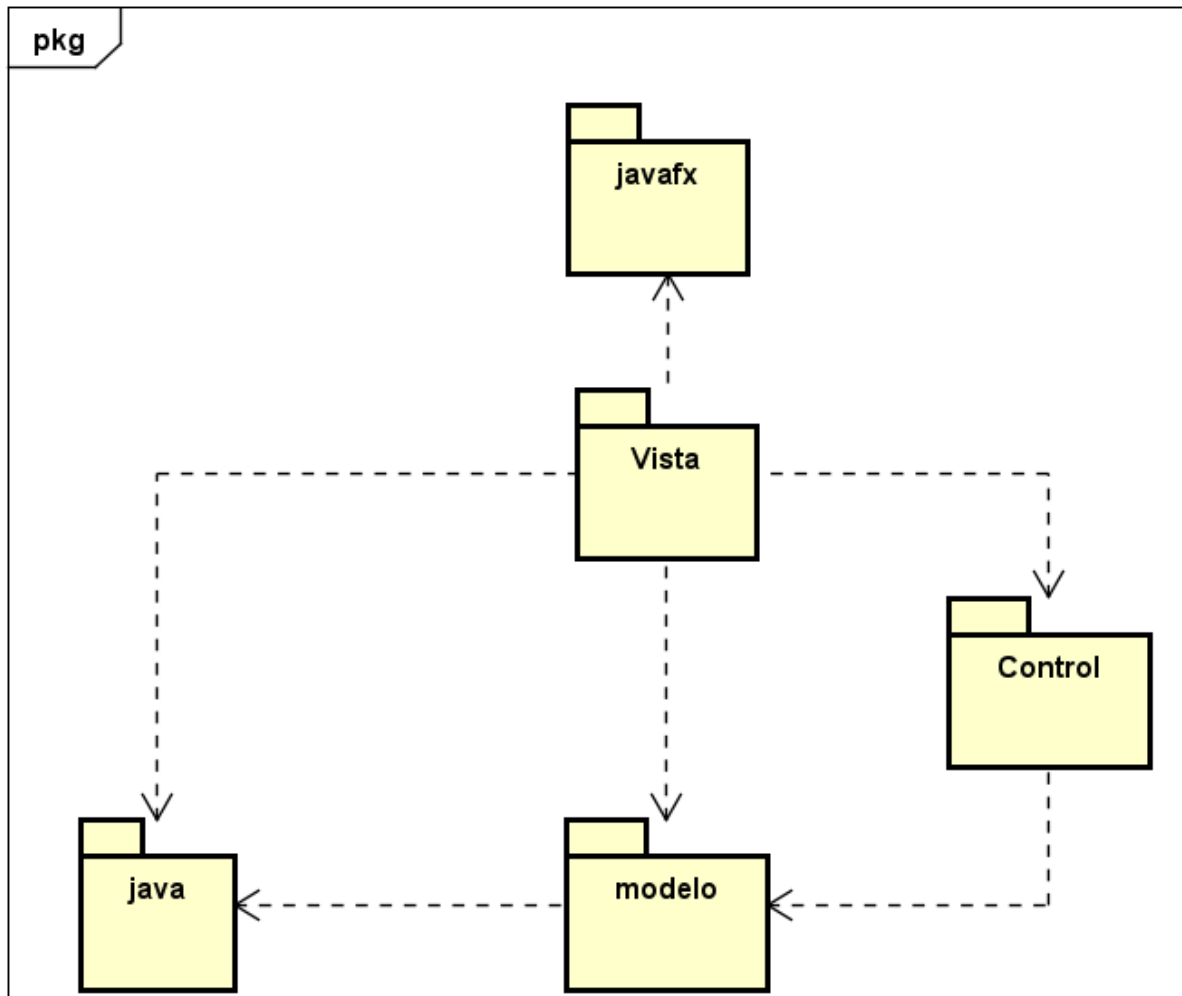


Figura 10: Diagrama de dependencia de los paquetes del programa.

En el diagrama de paquetes se puede apreciar que la parte del modelo solo depende de los paquetes de java. En otras palabras, el modelo no depende de la vista ni de sus controladores; cualquier cambio que se realiza a la interfaz del usuario, como a la lógica que afecta a esta, no afecta al modelo interno.

El diagrama es una simplificación de un diagrama generado con la herramienta **AstahUML** sobre todas las clases del programa, para verificar que el modelo efectivamente no tenga ningún acoplamiento con la vista.