FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Algoblocks

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

Alumno	Padron	Email
FIRMAPAZ, Agustin	105172	afirmapaz@fi.uba.ar
GUTIERREZ, Serena	105193	segutierrez@fi.uba.ar
DE SANTIS, Federico	101830	fdesantis@fi.uba.ar

Índice

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 un juego que cuenta con un personaje que debe moverse por la pantalla mientras dibuja con un lápiz, logrando realizar distintos diseños. El trabajo practico se desarrollará utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Supuestos

Para la implementacion del modelo, se han considerado algunas suposiciones sobre casos de estudio no especificados en la consigna del trabajo:

- La pizarra sobre la cual es personaje dibuja es de dimensiones finitas (esto no requiere de muchas explicaciones).
- Existe una sola pizarra, y si bien por el momento solo existe un personaje, no se descarta que en un futuro pudiese buscarse que existan mas de uno.
- Si un personaje se mueve de una celda a otra con el lapiz bajo, entonces dibuja tanto la celda donde se encontraba parado como la celda a donde se mueve. Esto simboliza los dos puntos que conectan la linea que el personaje dibuja.
- Una celda dibujada no puede ser borrada (a excepcion de que se de la orden de limpiar toda la pizarra).
- Al insertar un bloque, si no se determina la posicion de insercion, entonces el bloque se insertara al final de la secuencia.
- Una vez que una secuencia de bloques se inserta dentro de un bloque contenedor (bloque de repeticion, inversion de comportamiento, o personalizado), esta no puede ser modificada.
- Al guardar una secuencia personalizada, se asume que el usuario no ingresara un nombre vacio, o dos nombres iguales.
- Si el personaje llega al borde de la pizarra, cualquier movimiento que provoque que se salga de la pizarra se anula.

3. Modelo de dominio

La aplicacion Algoblocks consiste en un tablero interactivo, mediante el cual el usuario puede ordenar algoritmos de forma secuencial, para luego ejecutarlos sobre el dibujante y la pizarra. Dado esto, el tablero se encuentra compuesto por una pizarra, el personaje, el sector de bloques y el sector del algoritmo.

El sector de bloques es la parte del modelo que se encarga de proporcionar los bloques pedidos por el usuario, para su posterior agregacion al sector de algoritmos. Ademas, permite la posibilidad de almacenar un algoritmo personalizado.

El sector de algoritmos implementa la parte donde el usuario movera los bloques de algoritmo a voluntad con el fin de formar la secuencia de ejecucion. Este se encuentra implementado mediante una lista enlazada de secuencias de bloques, en donde el usuario puede insertar una nueva secuencia dentro de la secuencia que quisiese, y separar una secuencia deseada para recuperar una porcion de ella.

El personaje es el ente que se encarga de obedecer las ordenes dadas por la secuencia de bloques de algoritmos. Puede recibir ordenes de bajar o subir su lapiz, y de moverse en cualquier direccion

a lo largo de la pizarra. Dependiendo de la posicion del lapiz, el personaje ira dibujando la pizarra a medida que se desplaza por ella.

La pizarra de dibujo es el lienzo donde, como se menciono previamente, el personaje podra dibujar. Se encuentra implementado mediante una matriz de celdas de tamaño definido, las cuales tienen como atributo la posicion que la identifica y el estado en el que se encuentra (dibujada o en blanco).

4. Diagramas de clase

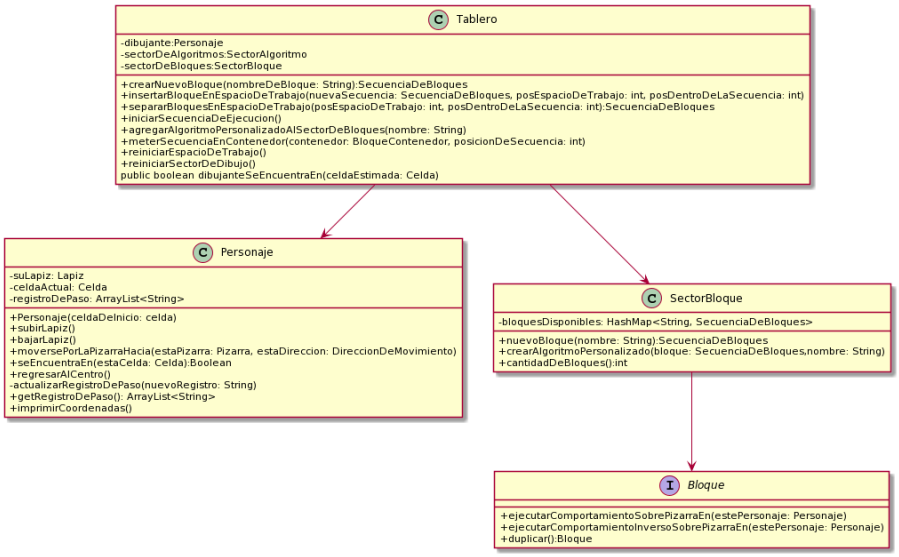

Figura 1: Tablero.

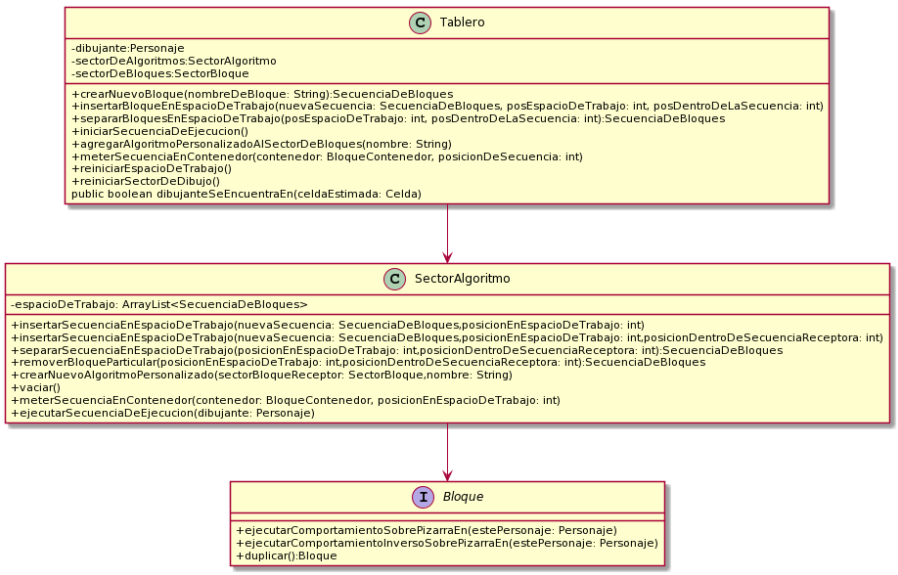

Figura 2: Tablero.

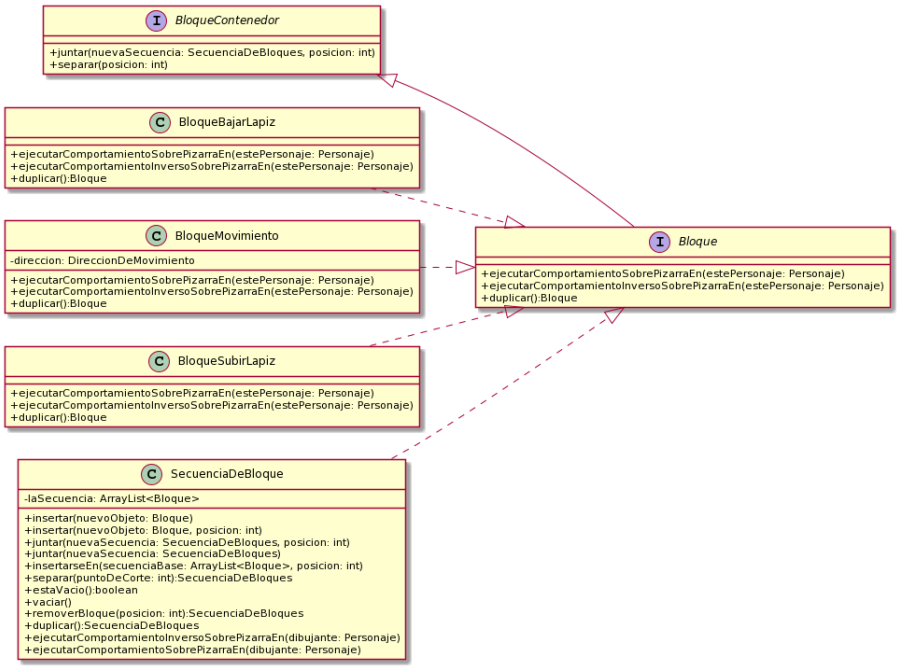

Figura 3: Bloques simples.

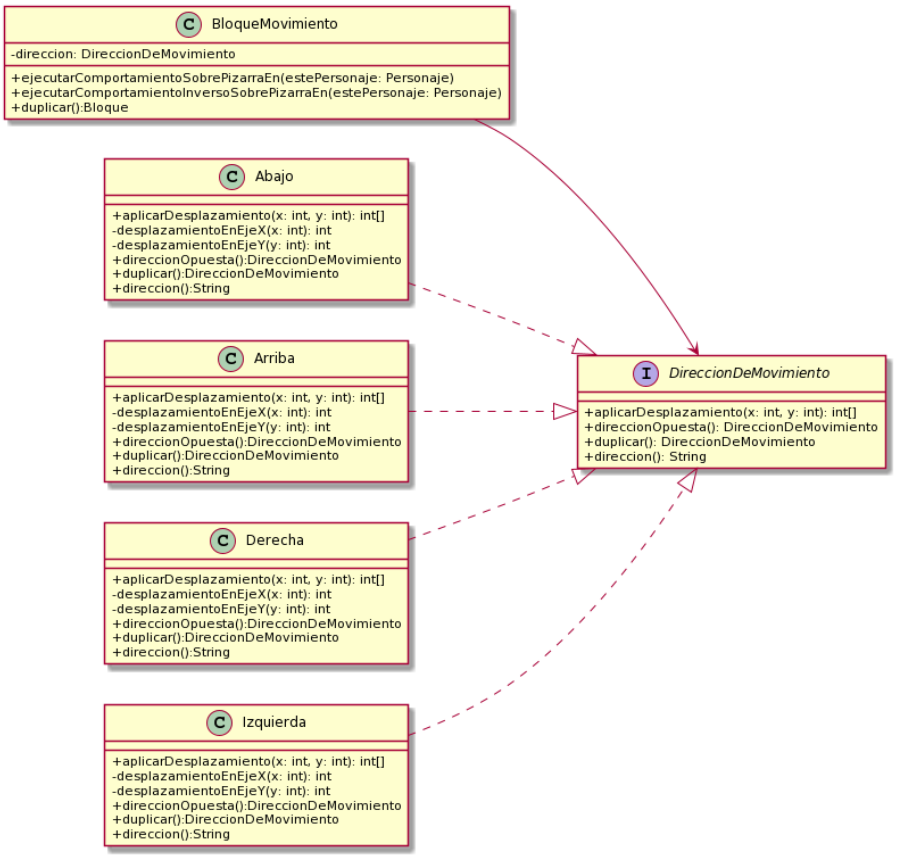

Figura 4: Direcciones de movimiento.

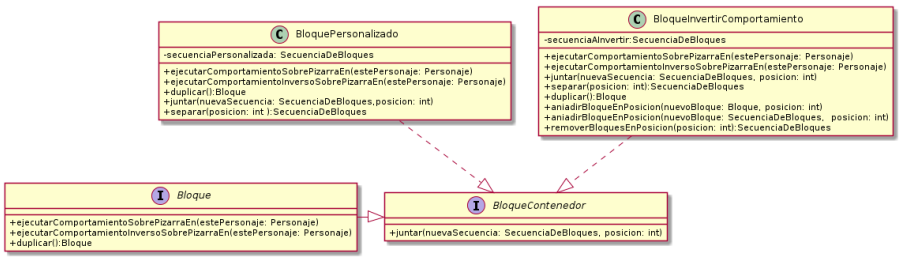

Figura 5: Bloques contenedores.

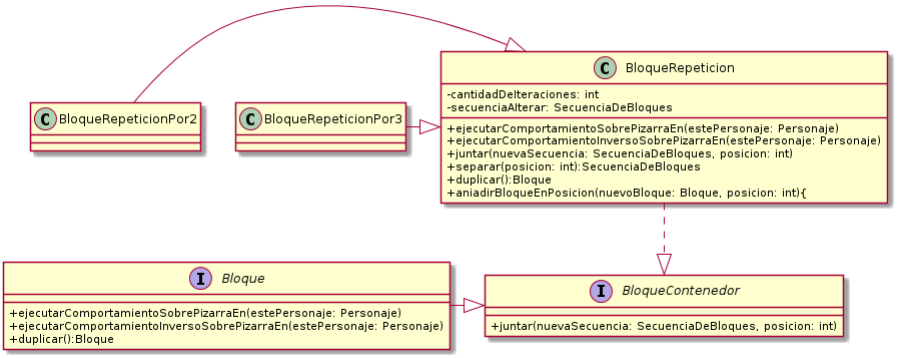

Figura 6: Bloque de repeticion.

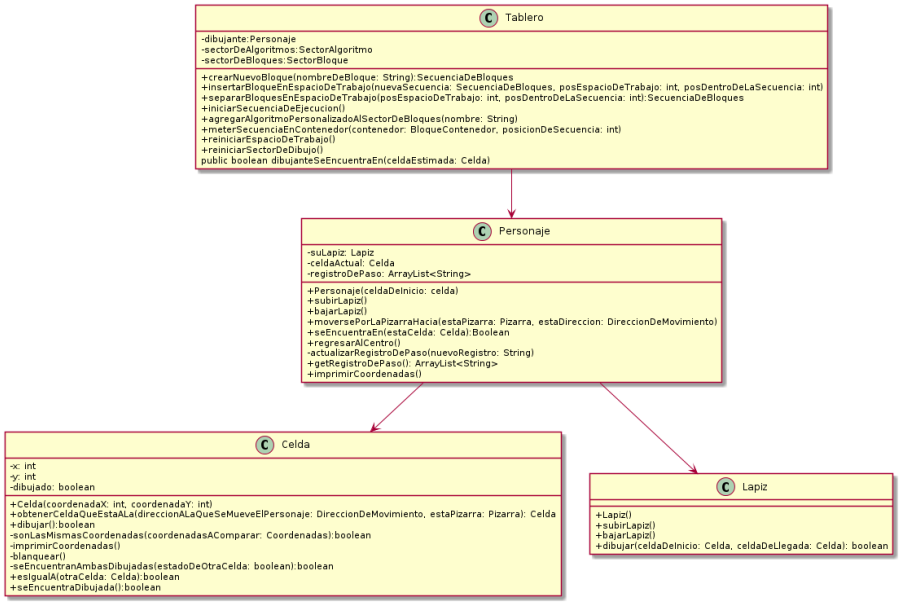

Figura 7: Personaje.

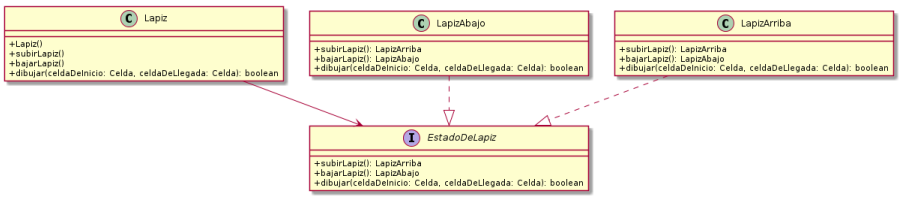

Figura 8: Estados del lapiz.

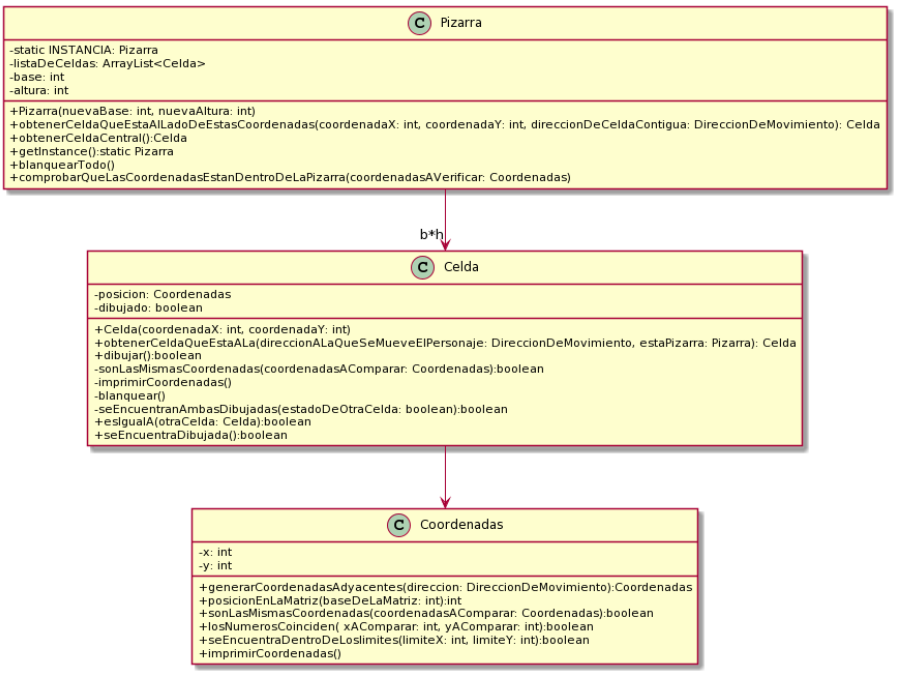

Figura 9: Pizarra.

5. Detalles de implementación

5.1. Movimiento del Personaje (Dibujante) por la Pizarra

El sector dibujo se compone de dos elementos basicos: el dibujante, y la pizarra donde se dibuja. La pizarra se encuentra implementada por una lista de celdas, la cuales simulan la estructura de una matriz de b x h tamaño. Debido a que el dentro del modelo solo puede existir una sola pizarra, se ha optado por crear una unica instancia global mediante el patron de diseño Singleton. Si bien podria ser problematico, dado que brindamos un acceso global a la pizarra para cualquier objeto, dentro de nuestro modelo la unica clase que se comunica con la pizarra es nada mas ni nada menos que el propio personaje.

El dibujante del sector dibujo se encuentra representado por el objeto Personaje. Este personaje posee un lapiz (objeto Lapiz) y una referencia a la celda de la pizarra donde se encuentra actualmente (*Ver figura 7*). El dibujante podra tomar como acciones: subir y bajar su lapiz, y moverse por la pizarra en cierta direccion de movimiento.

En particular, para desplazarse por la pizarra, el sector dibujo debera indicarle al personaje la direccion a la que se debe mover, y pasarle como argumento la referencia a la pizarra donde esta el dibujante. El dibujante, al saber la celda donde se encuentra parado, le indicara a la celda que le devuelva la celda que se encuentra en la direccion que el personaje debe moverse. Una vez

obtenida la celda contigua, se derivara la orden de dibujar tanto la celda origen como la celda destino, la cual dependera del estado en que el lapiz se encuentre. Y, por ultimo, el personaje reemplazara su celda actual por la celda destino, concretando asi el movimiento.

Por otra parte, para obtener la celda contigua, la celda origen (donde se encuentra parado el personaje en este momento) debera pasarle sus coordenadas a la pizarra y marcarle a que lado de esas coordenadas se encuentran la celda deseada. La pizarra realizara una conversion de las coordenadas para obtener las coordenadas de la celda deseada, y la buscara en la matriz para finalmente devolverla.

5.2. Dibujo sobre la pizarra

Cuando un personaje se mueve a lo largo de la pizarra dibujara (o no) una linea en la direccion a la que se mueve. Esto dependera del estado en que se encuentre el lapiz a la hora del movimiento: si el lapiz se encuentra levantado, el personaje no dibujara; si en cambio el lapiz se encuentra bajo, entonces dibujara .

Como evidenciamos en la explicacion anterior, para manejar esta rotacion de estados entre lapiz arriba y lapiz abajo, se ha optado por implementar el patron de diseño State. Esto permite al personaje conservar un unico lapiz el cual unicamente ira variando su estado, el cual se encargara de decidir si se dibujara o no al ejecutarse un movimiento (*Ver figura 8*).

5.3. Creacion de un bloque personalizado

Al momento de agregar un algoritmo personalizado al sector de bloques disponibles, el Sector Algoritmo envía un mensaje al Sector Bloque pasandole por parámetro la secuencia de ejecución y el nombre del nuevo algoritmo (Ver figura 14). El Sector Bloque se encarga de guardar una copia de la misma entre los bloques disponibles, con el nombre que indica el usuario. Ésta copia la hace la misma secuencia de ejecución, mediante el método duplicar(), crea una nueva lista, donde guarda una copia de los Bloques que contiene en su interior en el orden en que se encuentran (Ver figura 15) .

A su vez, las copias de los bloques las hacen los mismos Bloques, y las Direcciones de Movimiento dentro de cada uno tambien se duplican a ellas mismas (ambas clases poseen el método duplicar()), para que luego el bloque cree una nueva instancia de Bloque con la dirección dentro y lo devuelva a la secuencia de ejecucion. Esta secuencia se encarga de crear una nueva Secuencia de Bloques con la nueva lista dentro y la devuelve al Sector Bloque.

De esta forma, el Sector Bloque obtiene una copia exacta de la secuencia de ejecución, para guardarla dentro de un Bloque Personalizado entre los bloques disponibles para que el usuario utilice.

5.4. Ingreso de numeros por parte del usuario

Una de las caracteristicas de nuestra interfaz es que el usuario modela su algoritmo a traves de botones de ingreso y remocion de bloques. Para ello, el usuario tiene la posibilidad de ingresar un numero entero que indique en que posicion de la secuencia se accedera. Si el numero ingresado es positivo o cero, entonces la posicion comenzara a contarse desde arriba hacia abajo. Caso contrario, si el numero ingresado es negativo, entonces la posicion comenzara a contarse de abajo hacia arriba.

Para remediar este problema, se ha optado por recurrir a estructuras condicionales que manejen el numero ingresado al usuario y lo traduzcan a un numero que la secuencia de bloques pueda comprender. Esto podria parecer algo conflictivo a nivel de diseño, pero hemos considerado que la estructura condicional seria una buena opcion para resolver el problema debido a los siguientes motivos:

- El codigo condiciona el flujo del programa en una unica bifurcacion, determinada por una condicion logica (n pertenece al conjunto de los enteros positivos, o no). Esto es una regla de negocio que no depende de ningun objeto, sino de una simple condicion matematica.
- Las operaciones realizadas dentro de la estructura condicional no varian basandose en clases o interfaces que entiendan metodos, ni en valores de atributos de algun objeto.
- Si bien el flujo del programa si depende del resultado de un metodo enviado a un objeto (label.getText()), al ser un metodo de una libreria externa, el resultado obtenido no es un objeto al cual podamos implementarle comportamientos que creamos efectivo para la resolucion del problema dado que, por un lado, el resultado obtenido es un numero, y por otro lado, no podemos alterar el codigo de una libreria externa.

Bajo este analisis, concluimos que el uso de condicionales soluciona un problema que no podemos atacar desde ciertas perspectivas que el POO nos brinda.

6. Excepciones

FueraDeSecuenciaException Esta excepcion es utilizada para verificar que el acceso a las secuencias de bloques del espacio de trabajo sea a traves de una posicion valida en dicha secuencia.

FueraDePizarraException Esta excepción es utilizada para validar el rango en el que se ejecutaran los movimientos del personaje dentro de la pizarra.

NombreDeAlgoritmoPersonalizadoRepetidoException Cuando se intenta guardar un algoritmo con un nombre ya existente dentro de la lista de algoritmos, se lanza esta excepcion para notificar el rechazo.

7. Diagramas de secuencia

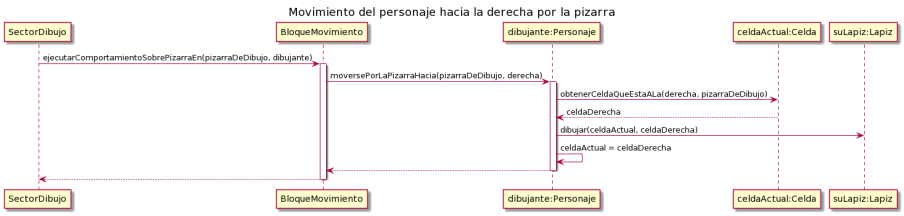

Figura 10: Personaje se mueve hacia la derecha.

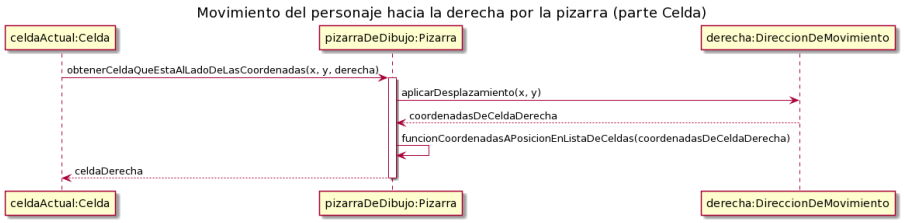

Figura 11: Personaje se mueve hacia la derecha (parte celda).

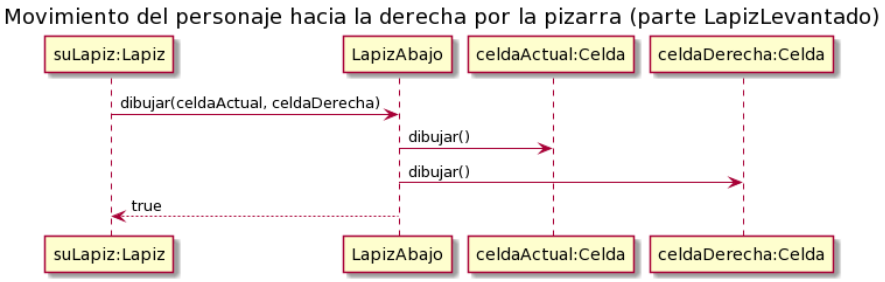

Figura 12: Personaje se mueve hacia la derecha (parte Lapiz Bajo).

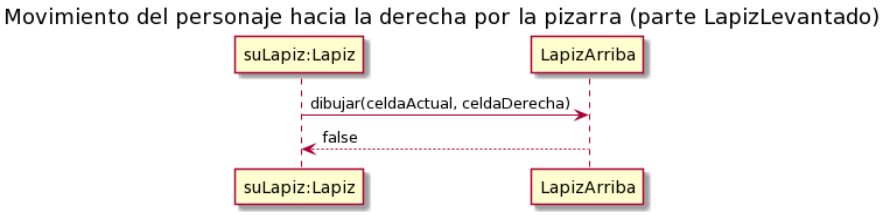

Figura 13: Personaje se mueve hacia la derecha (parte Lapiz Levantado).

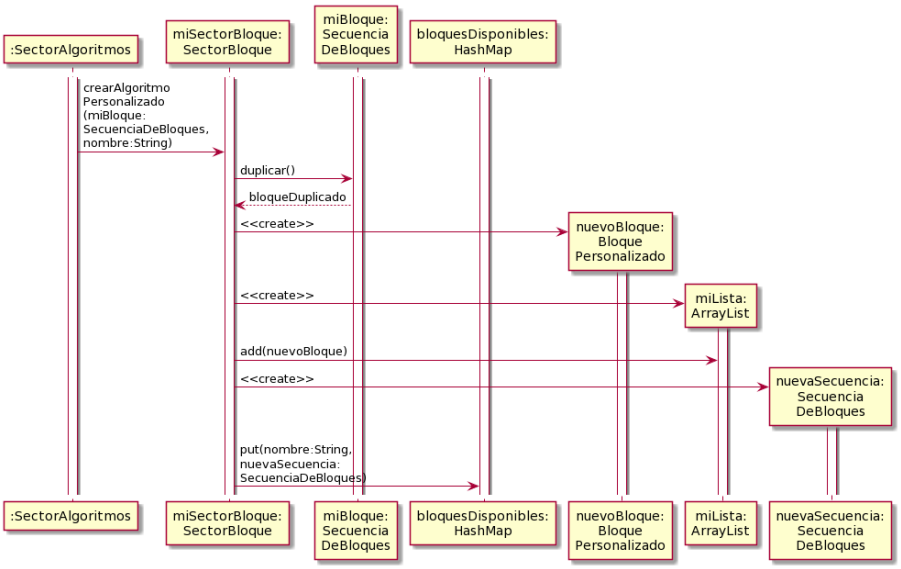

Figura 14: Creacion de algoritmo personalizado.

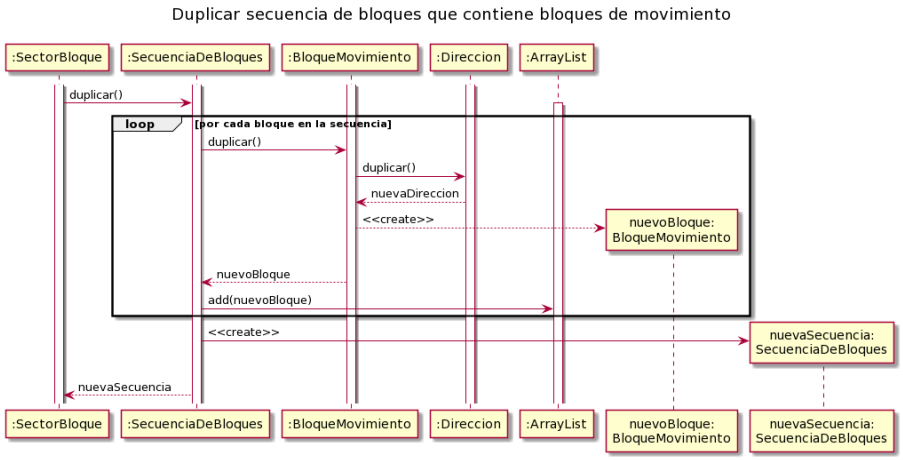

Figura 15: Duplicar bloque.

8. Diagramas de estado

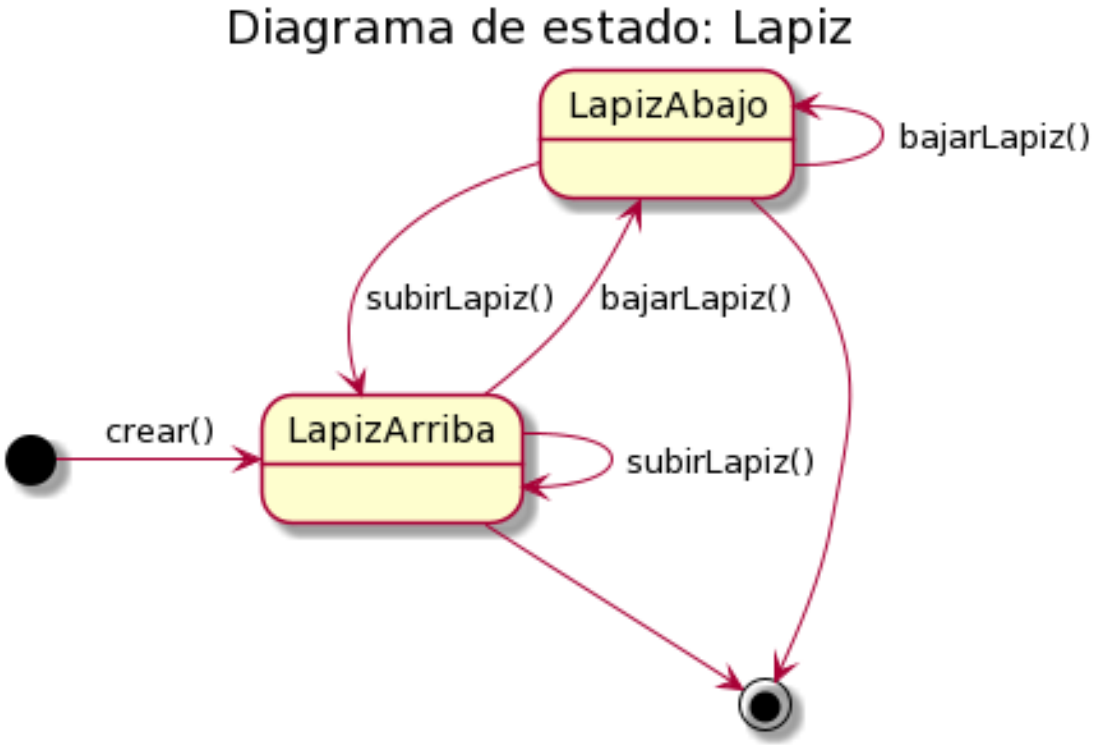

Figura 16: Estados del lapiz.

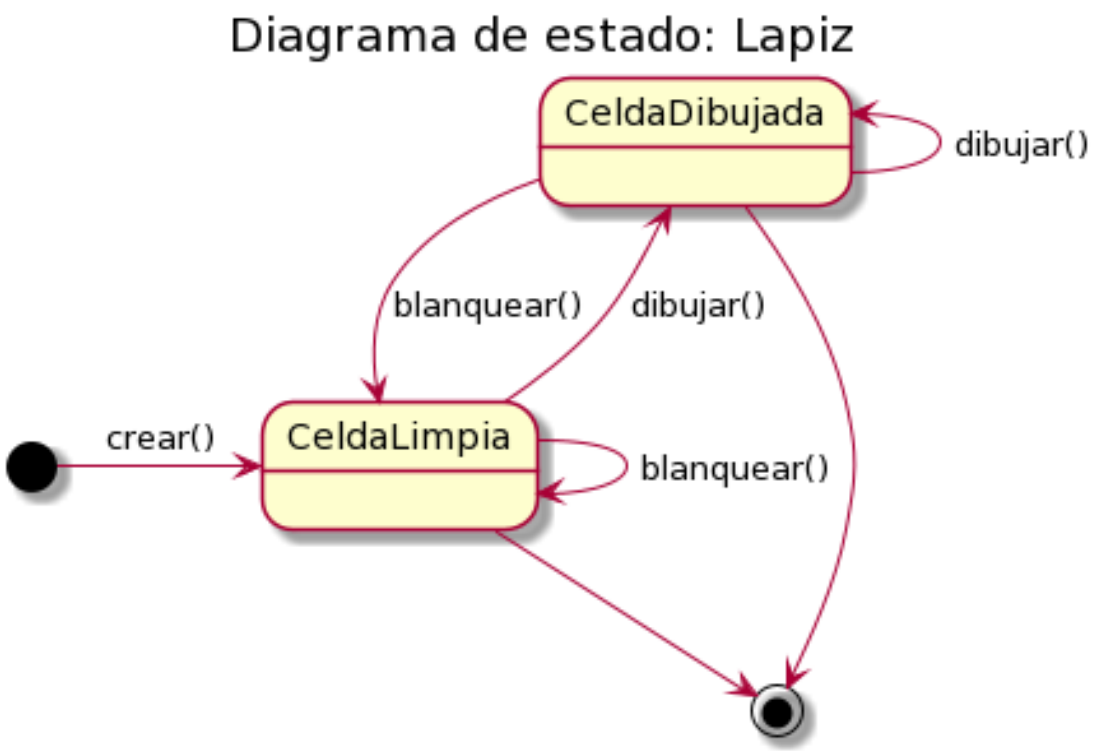

Figura 17: Estados de la celda.

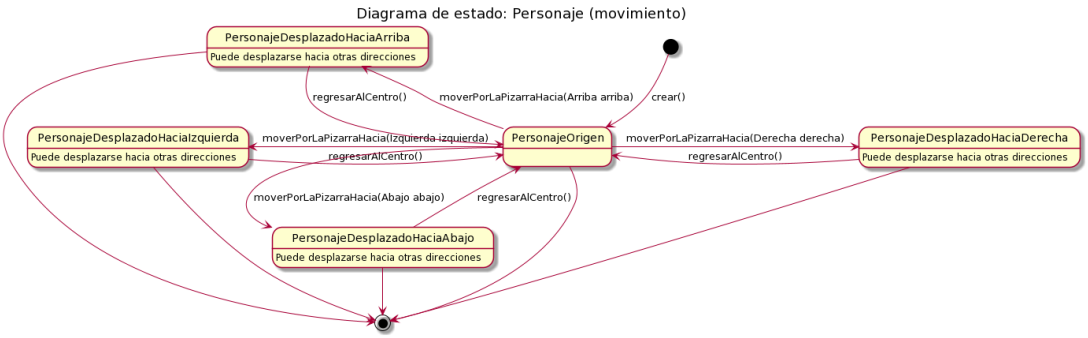

Figura 18: Estados del personaje al moverse.

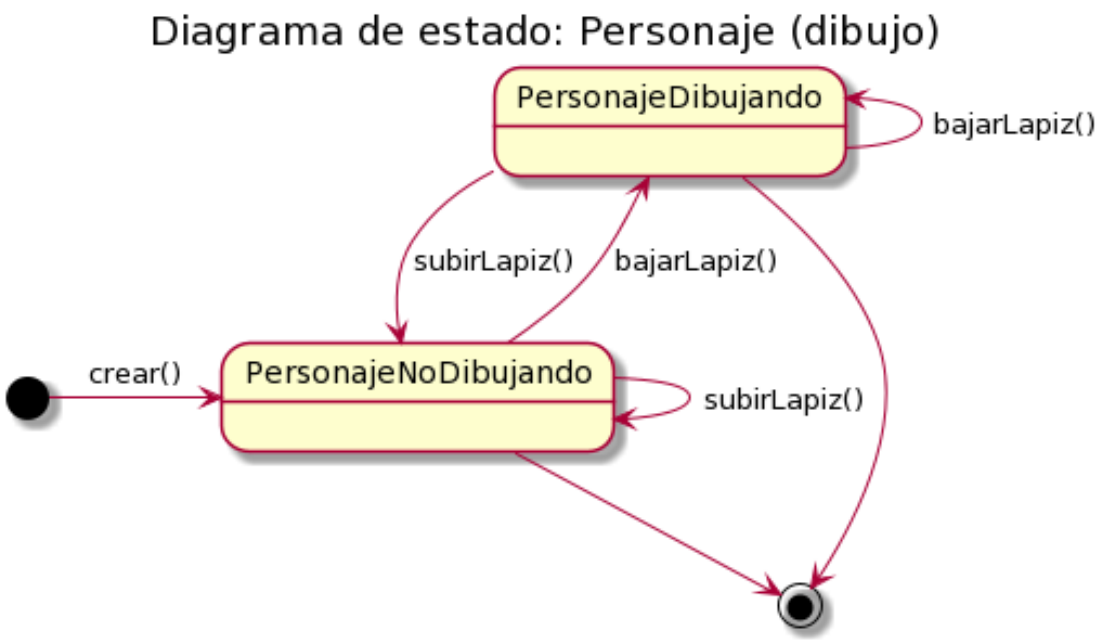

Figura 19: Estado de personaje al dibujar.

9. Diagramas de paquetes

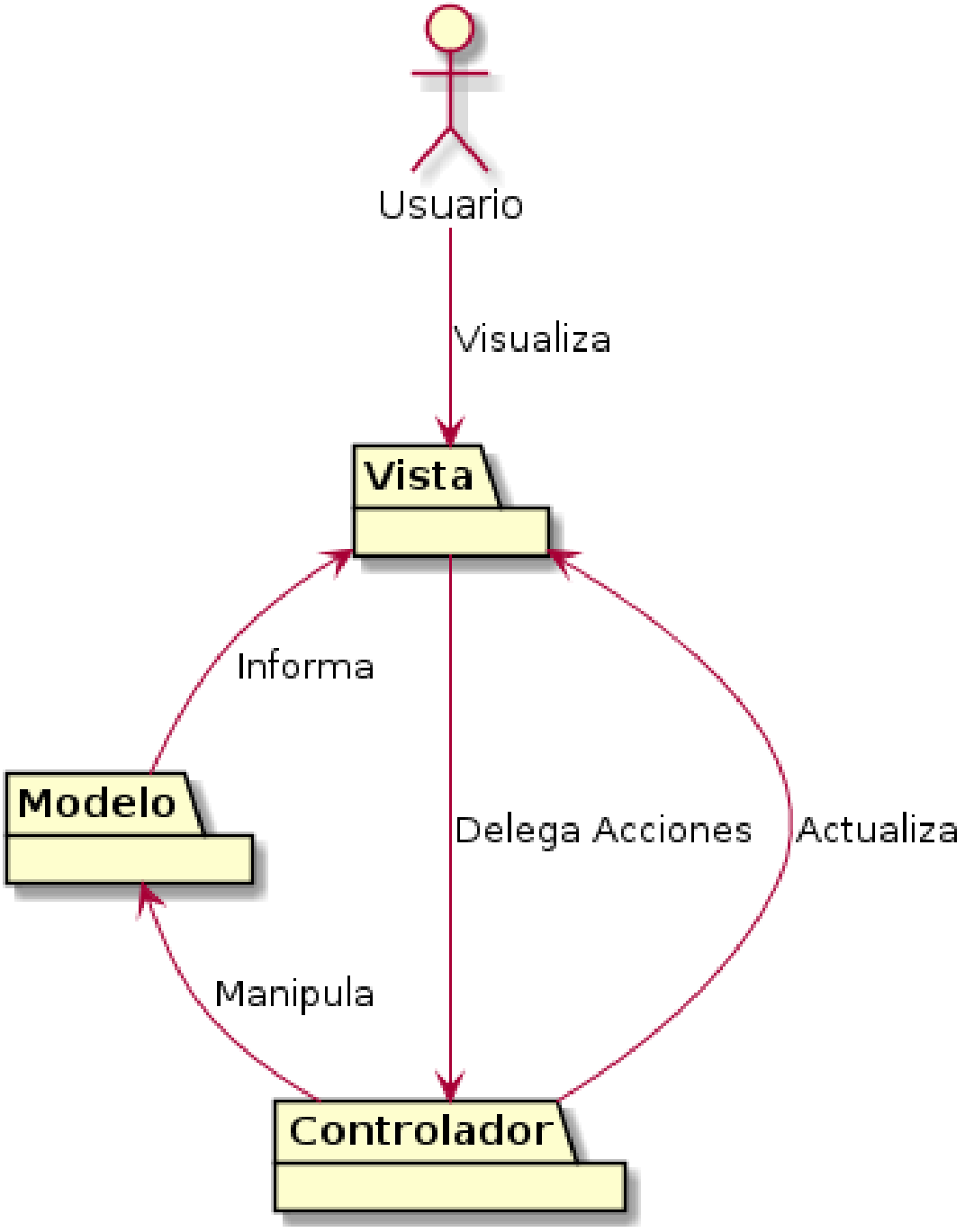

Figura 20: Diagrama de paquetes: MVC.