FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 2 – Grupo 10
Segundo Cuatrimestre 2020

Alumno 1:	Joaquín Lerer
Número de padrón:	105493
Email:	joacolerer@gmail.com

Alumno 2:	Nicolás Podesta
Número de padrón:	104077
Email:	npodesta@fi.uba.ar

Alumno 3:	Dante Reinaudo
Número de padrón:	102848
Email:	dreinaudo@fi.uba.ar

Alumno 4:	Sebastián Vera Benitez
Número de padrón:	104115
Email:	svera@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 una aplicación de un juego que permite aprender los conceptos básicos de programación, mediante el armado de algoritmos y la utilización de bloques visuales. Para lograr el objetivo se aplicaran de manera grupal todos los conceptos vistos en el curso, utilizando un leguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Supuestos

No existen límites para el movimiento del personaje.

3. Diagrama de clase

A continuación se muestra el diagrama de clases de la solución propuesta para abordar el presente trabajo práctico. En él se muestran todos las clases del sistema, sus atributos, sus métodos y la relación que mantienen entre si dichas clases. Cabe aclarar que en el diagrama no aparecen absolutamente todos los métodos y atributos, sino los más representativos para esclarecer la solución planteada.

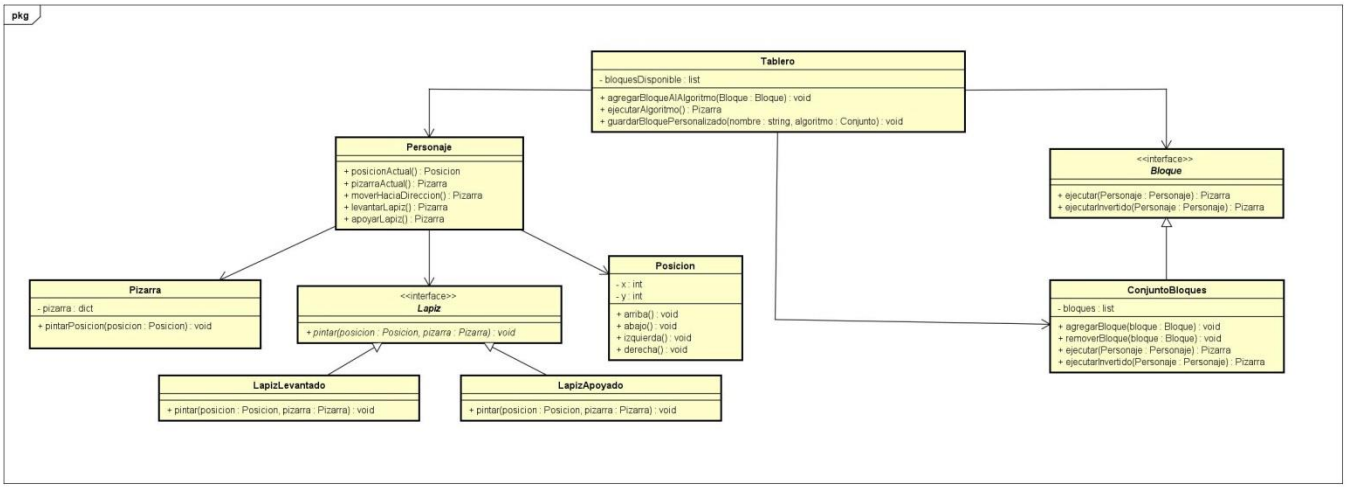

Figura 1: Diagrama de clases.

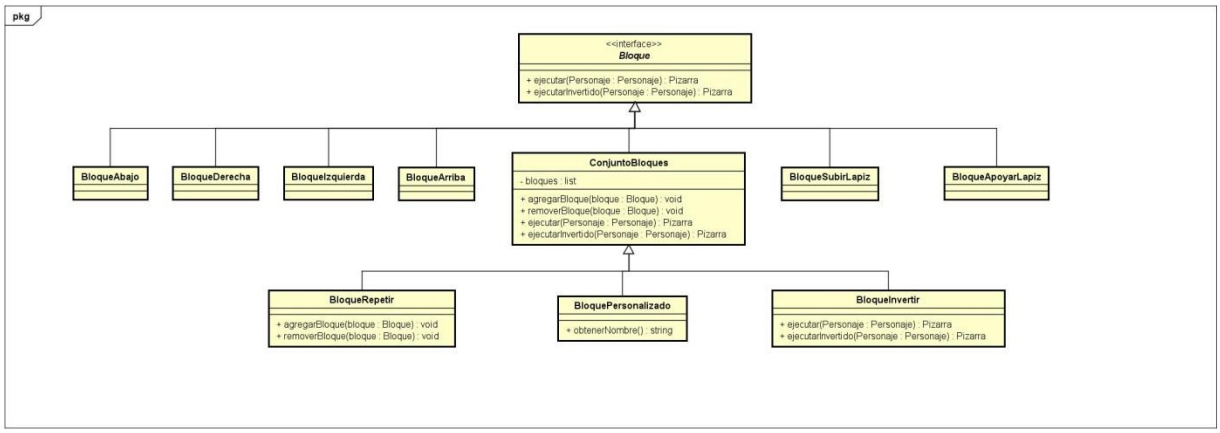

Figura 2: Diagrama de clases.

4. Diagramas de secuencia

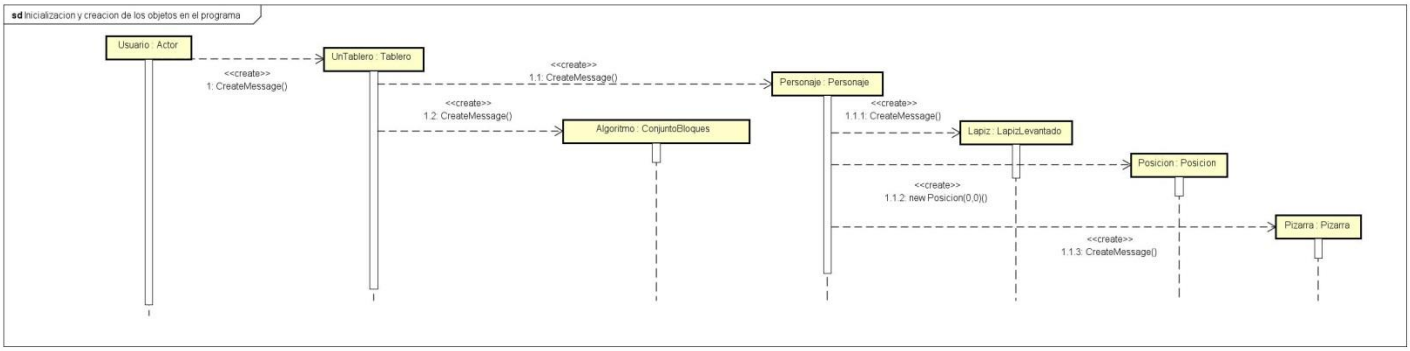

Figura 3: Diagrama de secuencia: Inicialización y creación de los objetos del programa

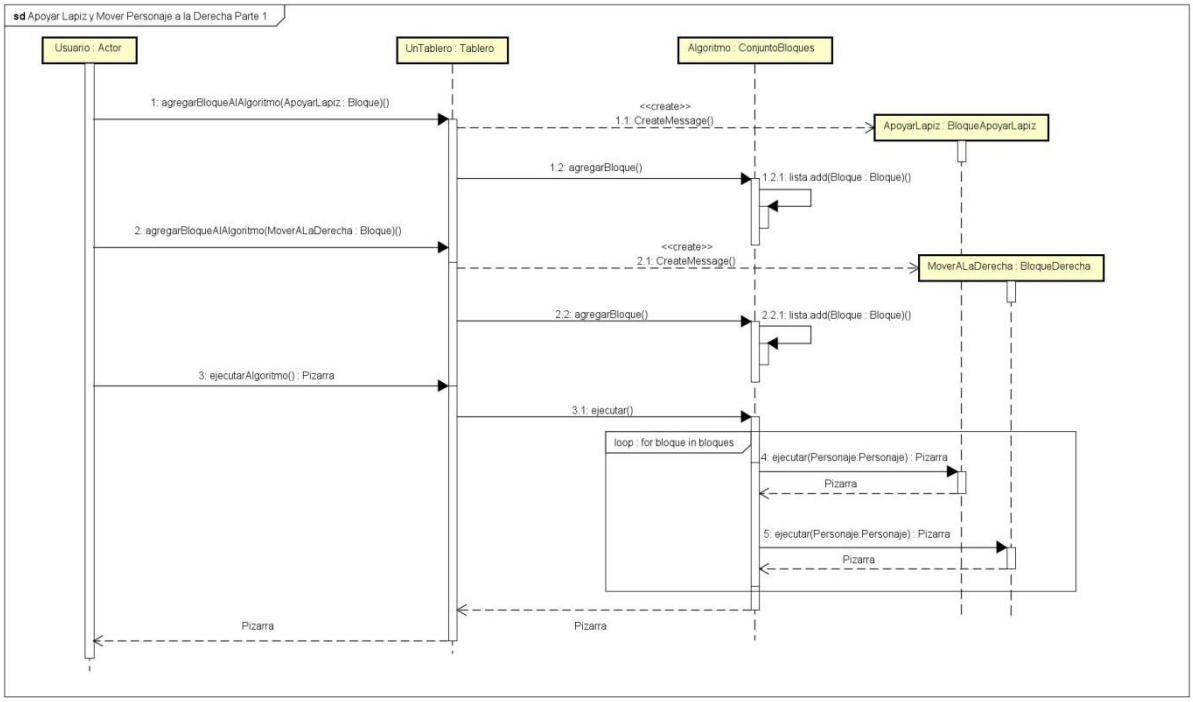

Figura 4: Diagrama de secuencia Apoyar Lapiz y mover Personaje a la derecha (Primera Parte).

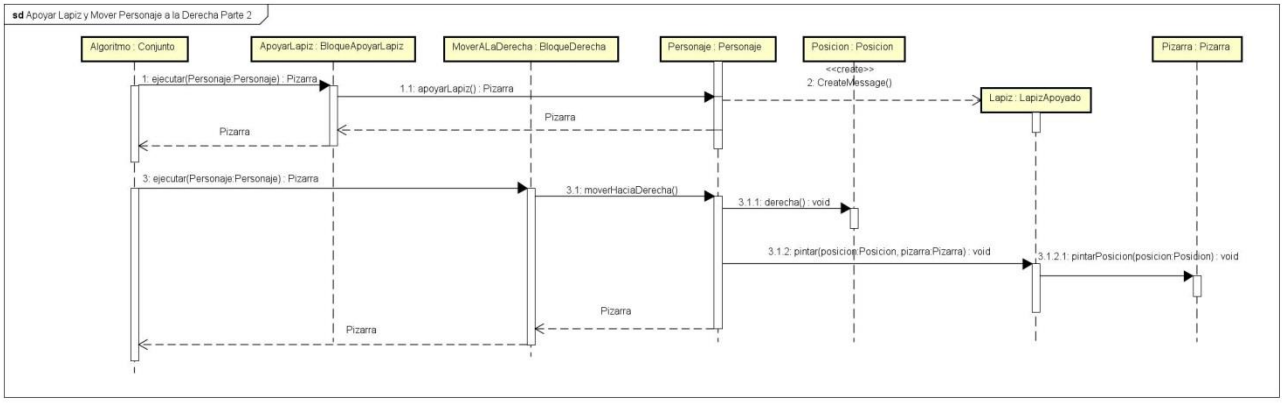

Figura 5: Diagrama de secuencia Apoyar Lapiz y mover Personaje a la derecha (Segunda Parte).

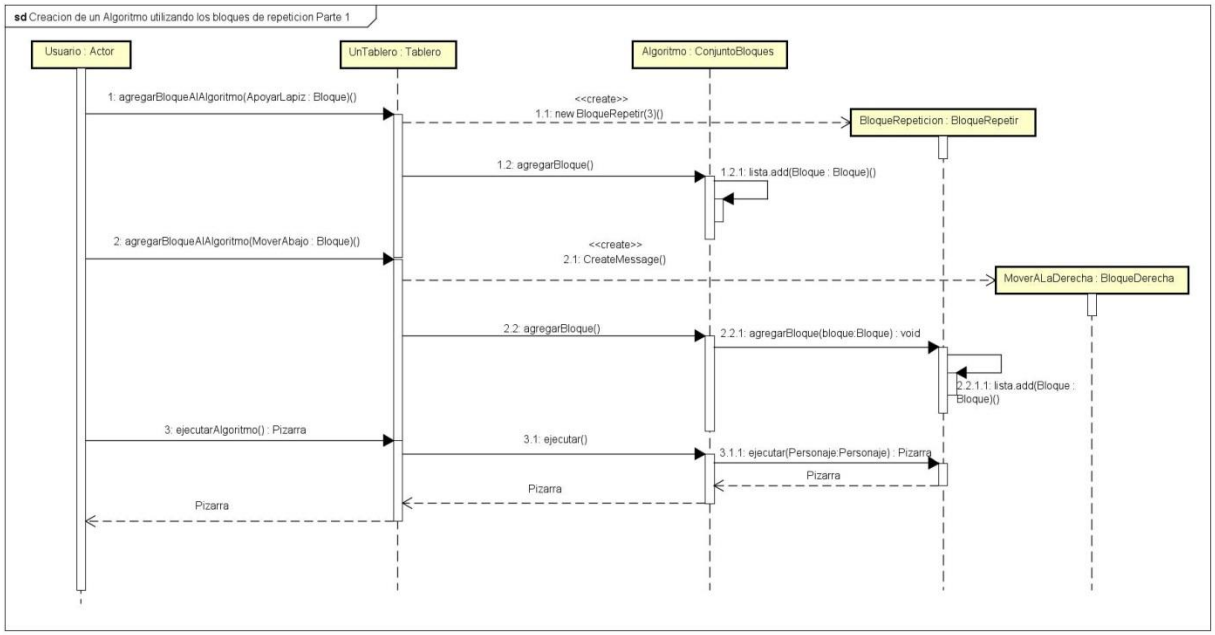

Figura 6: Diagrama de secuencia creación de un Algoritmo utilizando bloques de repetición (Primera Parte).

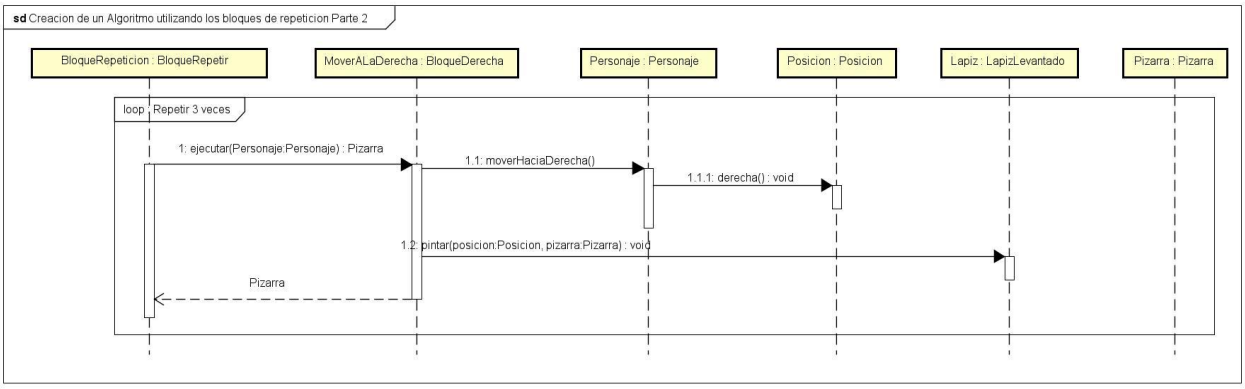

Figura 7: Diagrama de secuencia creación de un Algoritmo utilizando bloques de repetición (Segunda Parte).

5. Diagramas de paquete

6. Diagramas de estado

7. Detalles de implementación

7.1. Técnicas de diseño y planteo de la solución

Para resolver este trabajo utilizamos la metodología TDD (Test-Driven Depelopment) vista en clase, esta técnica de diseño consiste en ir realizando pruebas unitarias primero, haciendo lo mínimo y suficiente para poder cumplir con las pruebas proporcionadas por la cátedra. Una vez logrado esto se refactoriza el código escrito para obtener una mayor prolijidad, y así, cumplir con las llamadas buenas prácticas de programación. Este proceso se repite una y otra vez hasta poder abarcar todos los campos y llegar a una solución muy optimizada. Esta metodología presenta muchas ventajas, entre ellas: la garantía del correcto funcionamiento del código, gracias a que se cuenta con diversas pruebas de este; se minimiza la cantidad del código a escribir ya que al realizar pruebas, se evita la posibilidad de escribir código que no se va a utilizar; se mejora el diseño y la calidad de código.

Por otro lado, también hicimos uso de la práctica de Integración Continua, para así poder optimizar la detección de fallos y errores durante la compilación o ejecución de nuestro programa. Para lograr esto, utilizamos la herramienta de Integración Continua Travis-CI.

A su vez, para abordar diversas cuestiones de este trabajo práctico, aplicamos la metodología de Pair Programming, o también conocido como programación en pareja. Haciendo uso de la plataforma Disccord, pudimos debatir los puntos más conflictivos del trabajo, mientras que un compañero hacia *streaming* del código desde su computador y se encargaba de escribirlo, el resto de los integrantes del grupo supervisaba en tiempo real, debatiendo los problemas y encontrando soluciones.

7.2. Explicación de las Clases utilizadas y sus métodos

Como puede verse claramente en el diagrama de clases mostrado en la sección anterior, el modelo propuesto consta de dieciséis clases (más dos interfaces), las cuales se detallaran a continuación:

- **Tablero:**
- **Personaje**:
- **Pizarra**:
- **Posicion**:
- **Lapiz (Interfaz):**
- **LapizApoyado:**
- **LapizLevantado:**
- **Bloque(Interfaz):**

- **BloqueAbajo:**
- **BloqueArriba:**
- **BloqueDerecha:**
- **BloqueIzquierda:**
- **BloqueLapizaApoyado:**
- **BloqueLapizLevantado:**
- **ConjuntoBloques:**
- **BloqueInvertir:**
- **BloqueRepetir:**
- **AlgoritmoPersonalizado:**

8. Excepciones

Excepción AlgoritmoVacioException. Dado que en la consigna del trabajo se especifica claramente que, al momento de crear el algoritmo personalizado, el algoritmo a guardar debe contar con al menos un bloque, optamos por crear esta excepción. Se lanza cuando se intenta guardar un algoritmo personalizado que este vacío, es decir, no cuenta con ningún bloque.

Excepción BloqueInexistenteException. La clase ConjuntoBloques, y las subclases que de heredan de ella, nos permiten remover bloques del algoritmo con el cual estamos trabajando. Esta excepción nos permite solventar los casos en los cual el usuario desea remover un bloque del algoritmo, el cual no se encuentra en este.