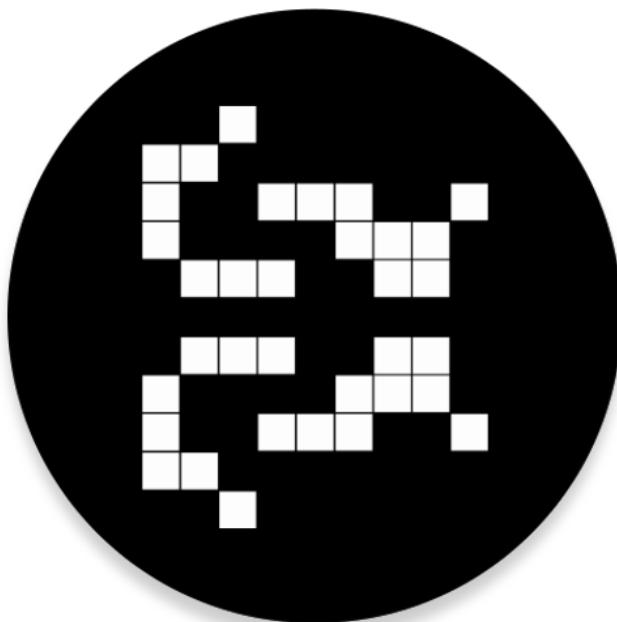


Design Patterns

Conway's Life Game 기능 확장 및 설계 개선



중앙대학교
창의ICT공과대학
소프트웨어학부

20185659 김혜성
20181267 박서현
20186669 박지수

Contents

Table of contents

1. 요약
 - 1.1. 기능 확장
 - 1.2. 설계 개선
2. 분석
 - 2.1. Use-case Diagram
 - 2.2. System Sequence Diagram
3. 아키텍처
 - 3.1. Design Class Diagram
 - 3.2. Sequence Diagram
4. 개발
 - 4.1. Class Diagram
 - 4.2. Gradle
 - 4.3. Spring
 - 4.4. MVC Architecture Pattern
 - 4.4.1. Model
 - 4.4.1.1. [Model] Model class
 - 4.4.1.2. [Model] PatternModel class
 - 4.4.2. View
 - 4.4.2.1. [View] View class
 - 4.4.2.2. [View] LifePanel class
 - 4.4.2.3. [View] Cell interface
 - 4.4.2.4. [View] Neighborhood class
 - 4.4.2.5. [View] Resident class
 - 4.4.2.6. [View] Menu class
 - 4.4.2.7. [View] MenuBar class
 - 4.4.2.8. [View] PaletteView class
 - 4.4.2.9. [View] PatternImagePanel class
 - 4.4.2.10. [View] StatusLabel class
 - 4.4.3. Controller

- 4.4.3.1. [\[Controller\] Action interface](#)
- 4.4.3.2. [\[Controller\] ClearAction class](#)
- 4.4.3.3. [\[Controller\] LoadAction class](#)
- 4.4.3.4. [\[Controller\] StoreAction class](#)
- 4.4.3.5. [\[Controller\] ExitAction class](#)
- 4.4.3.6. [\[Controller\] HaltAction class](#)
- 4.4.3.7. [\[Controller\] TickAction class](#)
- 4.4.3.8. [\[Controller\] AgonizingAction class](#)
- 4.4.3.9. [\[Controller\] SlowAction class](#)
- 4.4.3.10. [\[Controller\] MediumAction class](#)
- 4.4.3.11. [\[Controller\] FastAction class](#)
- 4.4.3.12. [\[Controller\] DrawBehavior interface](#)
- 4.4.3.13. [\[Controller\] DefaultDrawBehavior class](#)
- 4.4.3.14. [\[Controller\] PatternDrawBehavior class](#)
- 4.4.3.15. [\[Controller\] LifeController class](#)
- 4.4.3.16. [\[Controller\] MenuController class](#)
- 4.4.3.17. [\[Controller\] PaletteController class](#)

4.5. [Service](#)

- 4.5.1. [\[Service\] MapData class](#)
- 4.5.2. [\[Service\] Service class](#)

4.6. [Worker](#)

- 4.6.1. [\[Worker\] Worker class](#)

4.7. [Test Cases](#)

- 4.7.1. [\[Test\] ModelTest](#)
- 4.7.2. [\[Test\] PatternModelTest](#)
- 4.7.3. [\[Test\] MenuTest](#)
- 4.7.4. [\[Test\]MenuBarTest](#)
- 4.7.5. [\[Test\]LifePanelTest](#)
- 4.7.6. [\[Test\]PatternImagePanelTest](#)
- 4.7.7. [\[Test\]StatusLabelTest](#)
- 4.7.8. [\[Test\]LifeControllerTest](#)
- 4.7.9. [\[Test\]MenuControllerTest](#)
- 4.7.10. [\[Test\]PaletteControllerTest](#)
- 4.7.11. [\[Test\]ClearActionTest](#)
- 4.7.12. [\[Test\]LoadActionTest](#)

- 4.7.13. [\[Test\] StoreActionTest](#)
- 4.7.14. [\[Test\] HaltActionTest](#)
- 4.7.15. [\[Test\] TickActionTest](#)
- 4.7.16. [\[Test\] AgonizingActionTest](#)
- 4.7.17. [\[Test\] SlowActionTest](#)
- 4.7.18. [\[Test\] FastActionTest](#)
- 4.7.19. [\[Test\] MediumActionTest](#)
- 4.7.20. [\[Test\] DefaultDrawBehaviorTest](#)
- 4.7.21. [\[Test\] PatternDrawBehaviorTest](#)
- 4.7.22. [\[Test\] Service Test](#)
- 4.7.23. [\[Test\] WorkerTest](#)

5. 결과

5.1. [기존 Life Game 프로젝트 지원 기능](#)

5.2. [확장 기능](#)

6. GitHub

6.1. [Repository URL](#)

6.2. [Commit history](#)

6.3. [Issues \(역 할분배\)](#)

6.3.1. [김혜성 작업 issue](#)

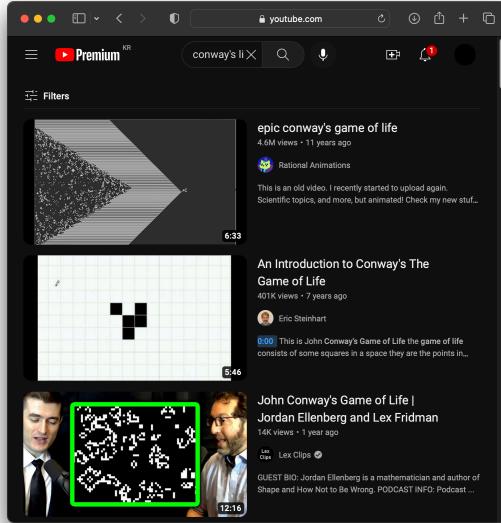
6.3.2. [박서현 작업 issue](#)

6.3.3. [박지수 작업 issue](#)

6.4. [Actions](#)

1. 요약

1.1. 기능 확장

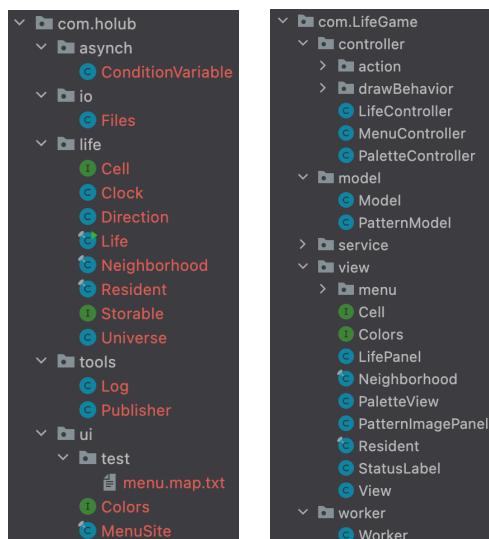


▲ [picture 01] “Conway’s Life Game”, YouTube

[picture 01]과 같이, 기존 Conway's Life Game 프로그램의 흥미 요소를 분석해보았을 때 게임적인 부분이 아닌, cell의 생명 패턴으로 여러 작품들을 만들어내는 예술적인 요소가 가장 크다고 생각했다. 따라서, 새로운 시스템이 아닌 해당 프로그램을 사용하여 느낀 불편함을 해결 할 수 있는 편의 기능을 추가하는 방향으로 기능 확장을 설계하였다.

우선, 작은 크기의 cell을 하나씩 누르며 패턴을 그리는 것이 번거롭고 규모가 큰 패턴을 만들기 어렵다고 느껴졌다. 또한 빠르게 동시다발적으로 cell의 상태가 변화하여 중간 진행 상황을 따라가기 힘들었다. 이러한 불편함을 해결하기 위해 1) Palette 기능과 2) 현재 generation수, live cells 개수 확인 기능을 구현하여 프로그램을 확장시켰다.

1.2. 설계 개선

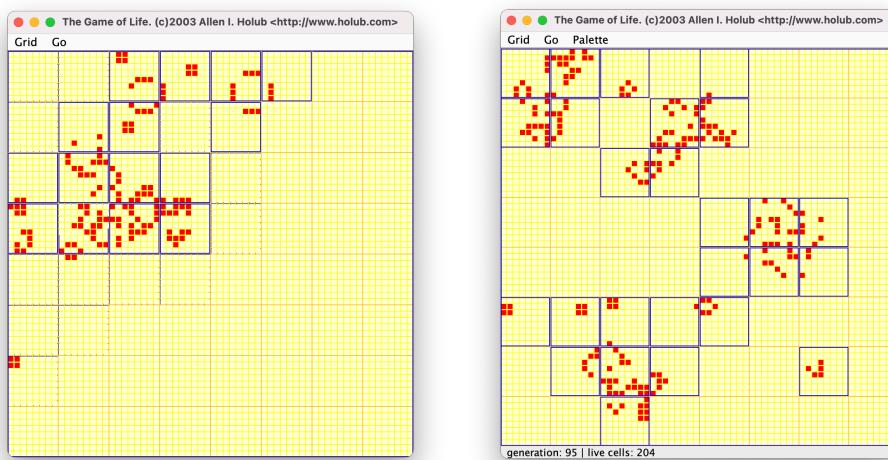


▲ [picture 02] 기존 파일 구조(좌), 개선한 파일 구조(우)

기존의 프로그램은 model, view, controller의 구분 없이 view에서 모든 로직을 수행하는 구조를 가지고 있어 여러 책임을 동시에 가져 SRP를 위반하고 있었다. 가장 큰 문제점은 OCP를 위반하여 기능을 확장하고자 하는데 어려움이 있었다. 또한 의존성 주입을 하지 않고 의존관계가 많아 DIP를 위반하고 테스트를 작성하는데 어려움이 많았다.

개선된 설계에서는 Spring Framework를 채택해 UI 컴포넌트들과 컨트롤러를 Spring Bean으로 등록해 Singleton으로 구성하고, Spring Container를 이용해 의존성을 주입해 DIP를 해소했다. 또 MVC Observer 패턴으로 작성하여 Model-View-Controller의 책임을 분리해 SRP를 지키고, View를 Observer로, Model을 Subject로 Observer Pattern을 채택하여 OCP를 지켰다.

또 Palette 기능을 추가하기 위해 Strategy Pattern을 채택하였고, Menu의 OCP를 잘 지키기 위해 Command Pattern을 적용하여 세부적으로도 유연한 설계를 완성하였다.

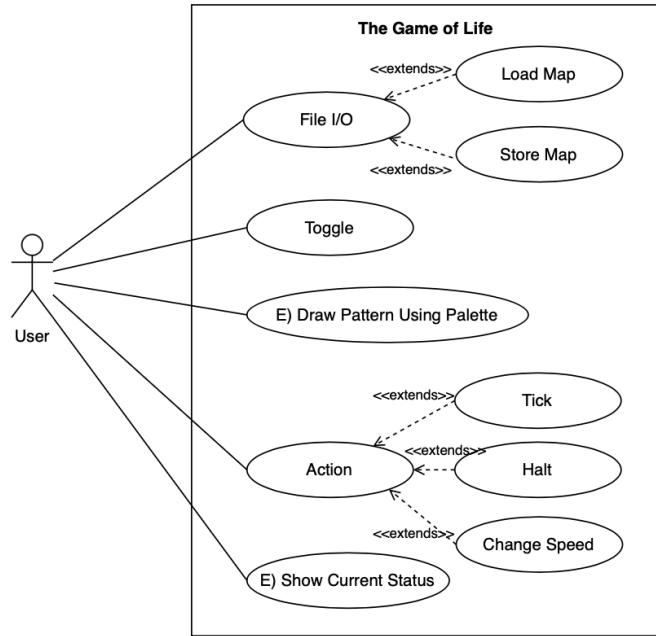


▲ [picture 03] 기존 프로그램(좌), 개선된 프로그램(우)

MVC 패턴을 이용한 설계를 적용한 결과, [picture 03]과 같이, 원래 프로그램에서는 Neighborhood의 파란색 테두리가 올바르게 그려지지 않았는데, 개선된 설계에서는 깔끔하게 적용된 것을 확인할 수 있었다. 또한 Unit Test를 작성할 때 Mockito를 활용해 쉽게 테스트를 작성할 수 있었고, 새로운 기능을 추가할 때도 OCP를 지켰기 때문에 개발 생산성 면에서도 장점을 보였다.

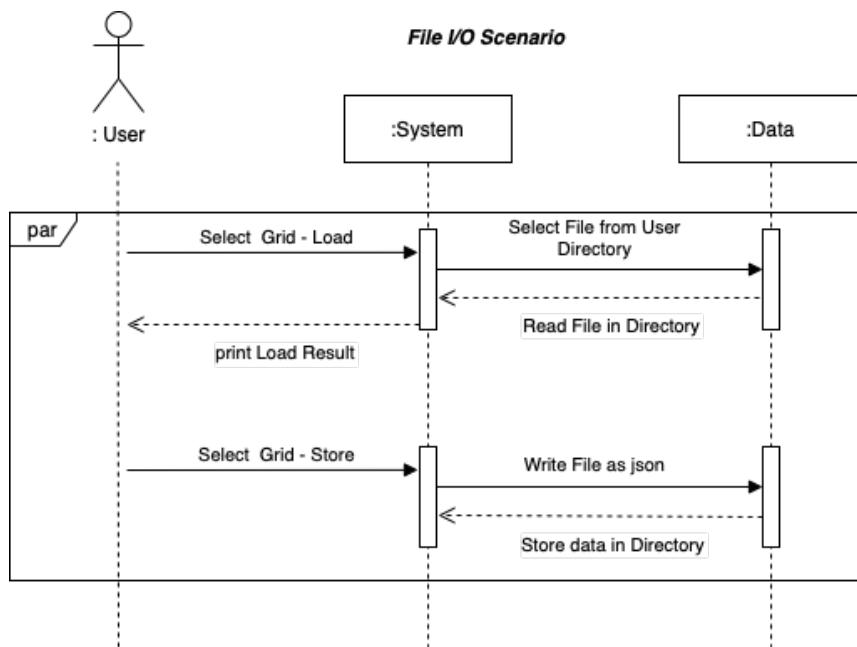
2. 분석

2.1. Use-case Diagram

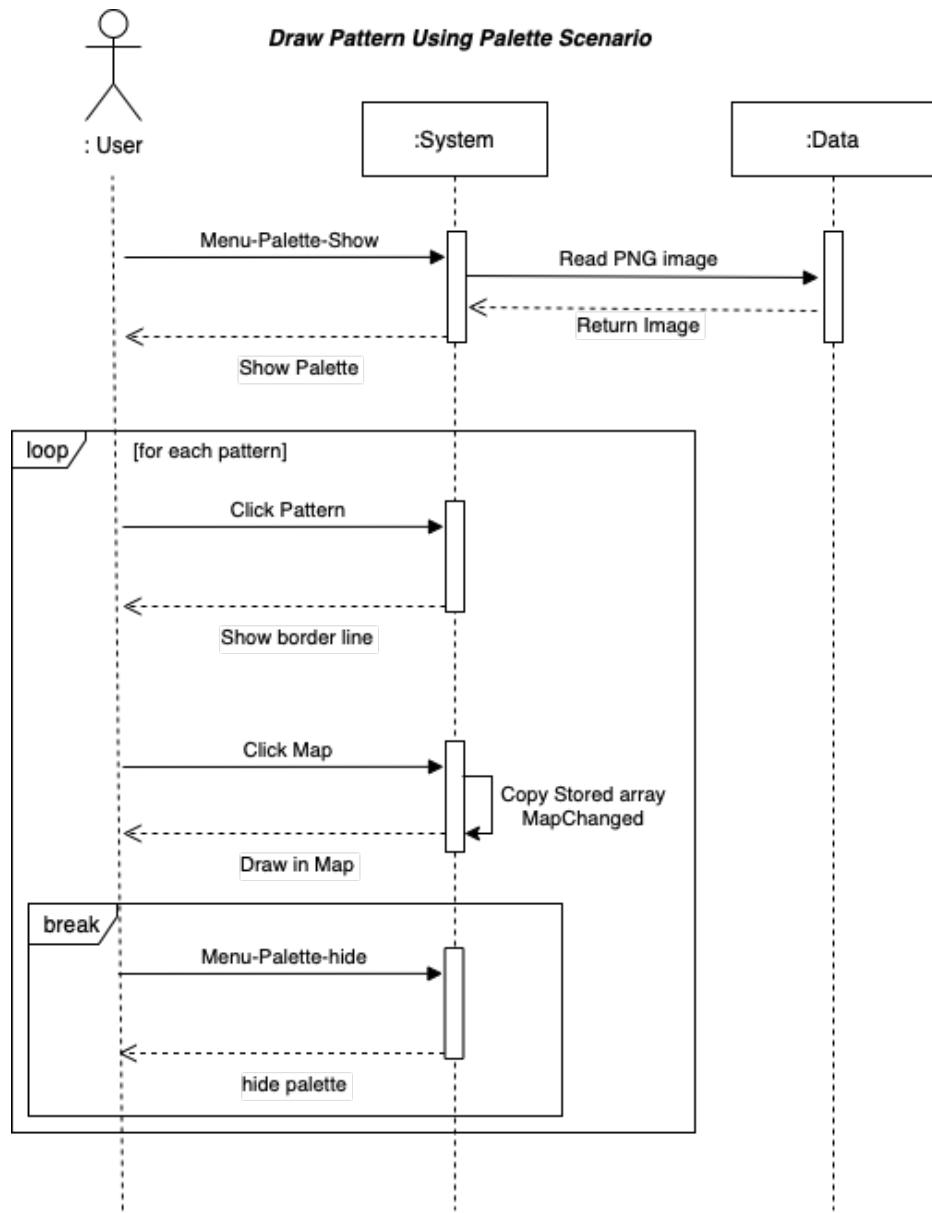


▲ [picture 04] Use Case Diagram

2.2. System Sequence Diagram



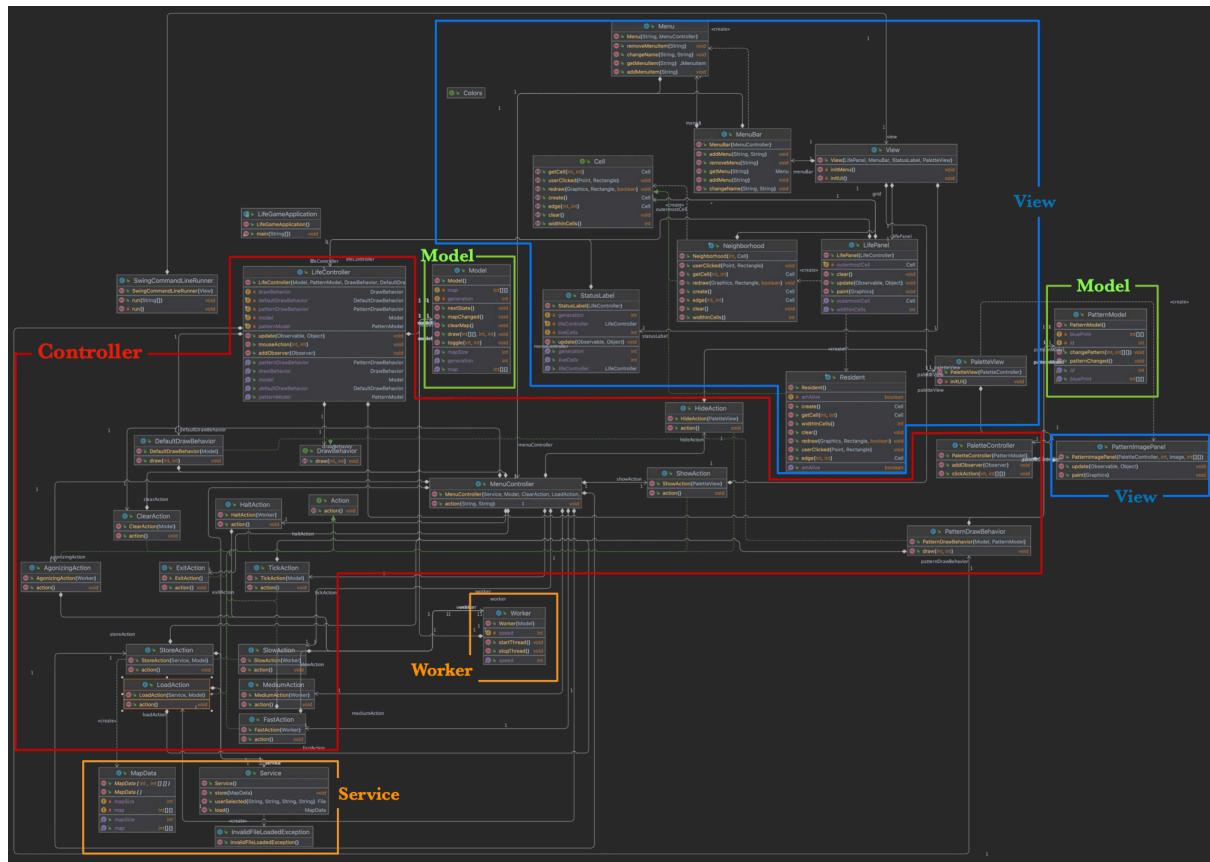
▲ [picture 05] System Sequence Diagram (File I/O Scenario)



▲ [picture 06] System Sequence Diagram (Draw Pattern Using Palette Scenario)

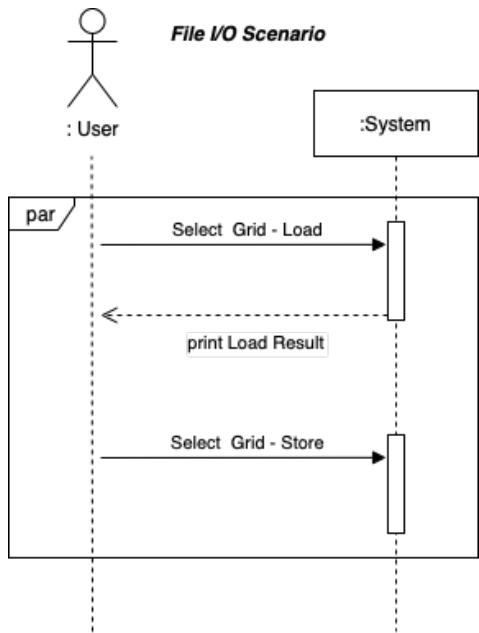
3. 아키텍처

3.1. Design Class Diagram

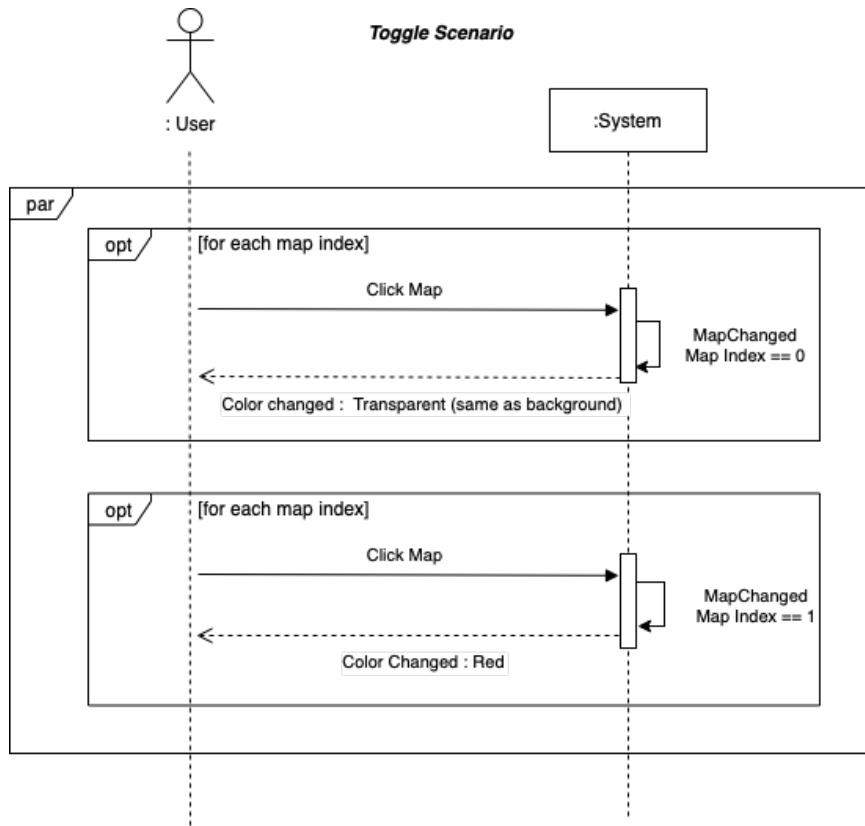


▲ [picture 07] Class Diagram Design

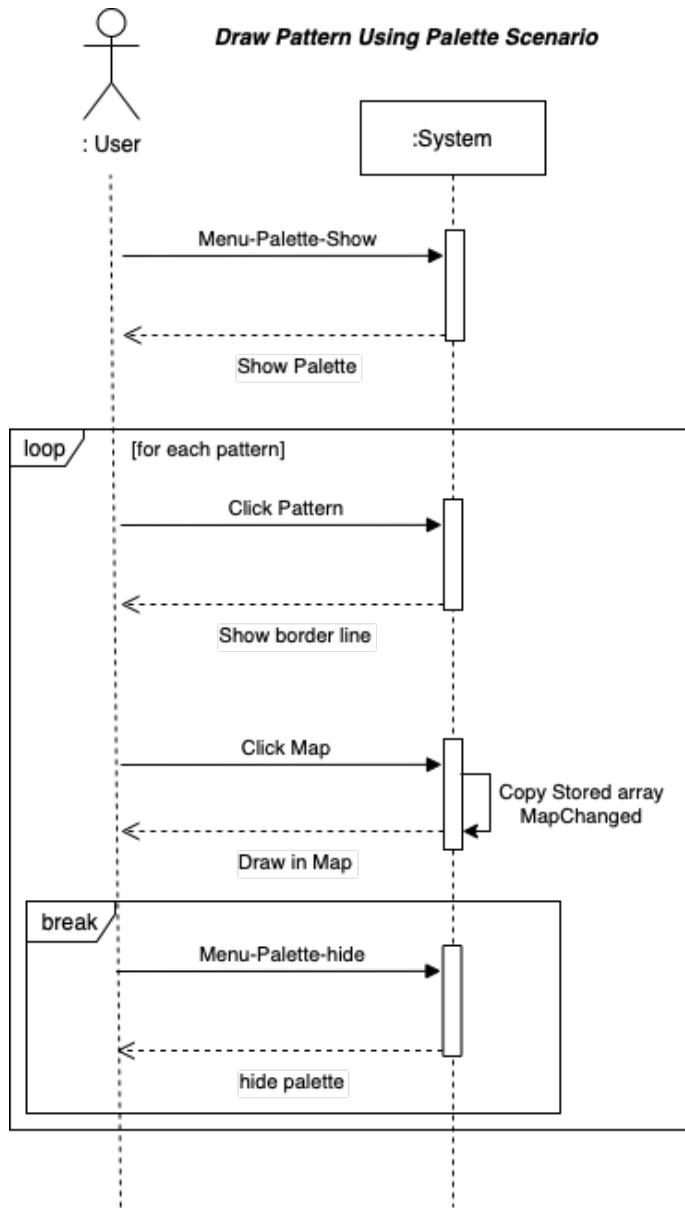
3.2. Sequence Diagram



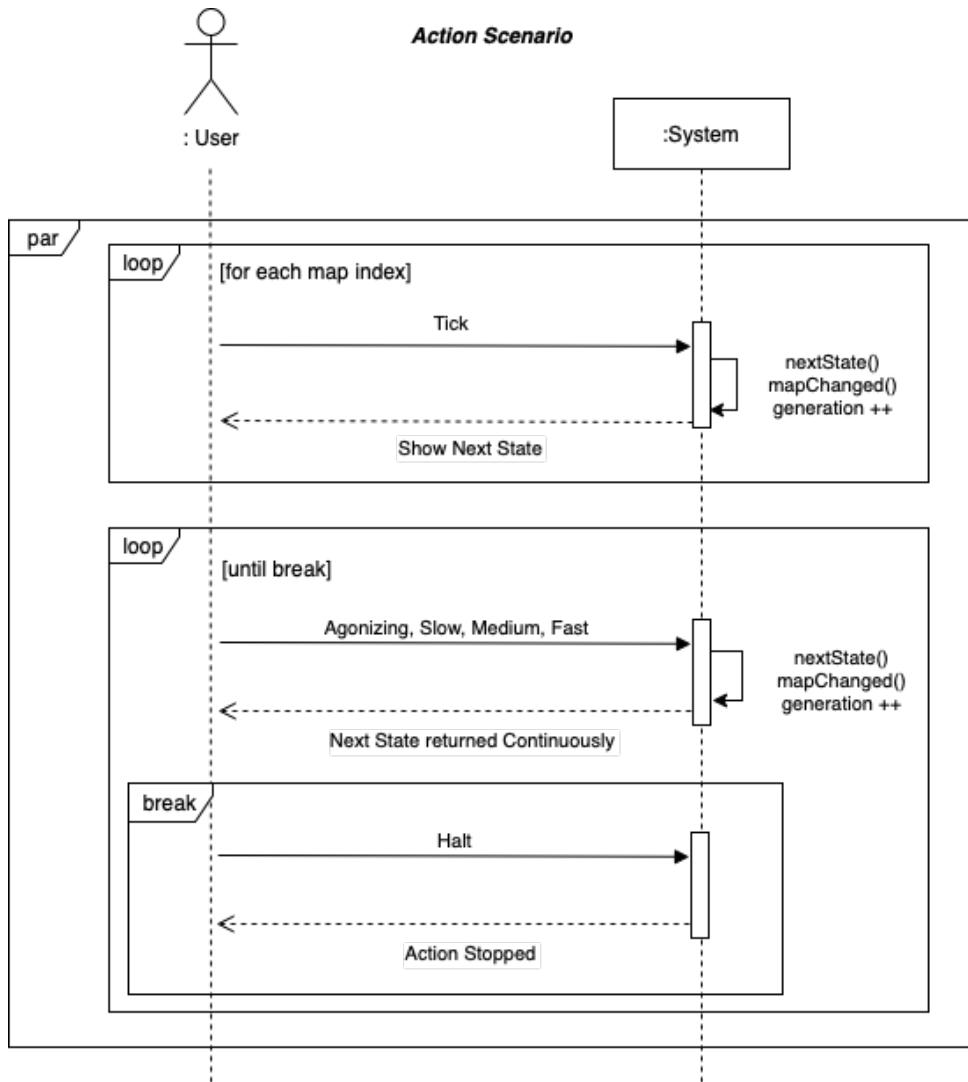
▲ [picture 08] Sequence Diagram (File I/O Scenario)



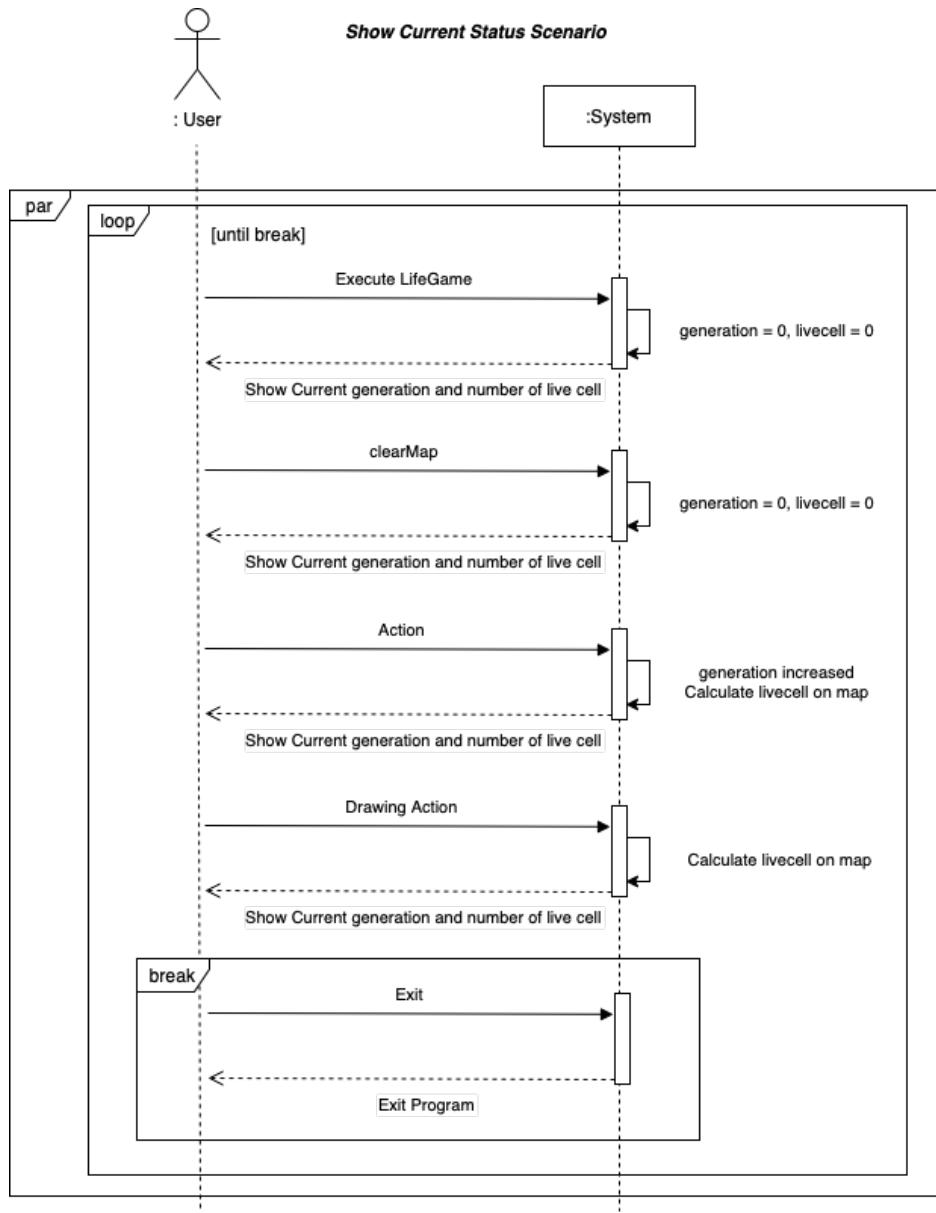
▲ [picture 09] Sequence Diagram (Toggle Scenario)



▲ [picture 10] Sequence Diagram (Draw Pattern Using Palette Scenario)



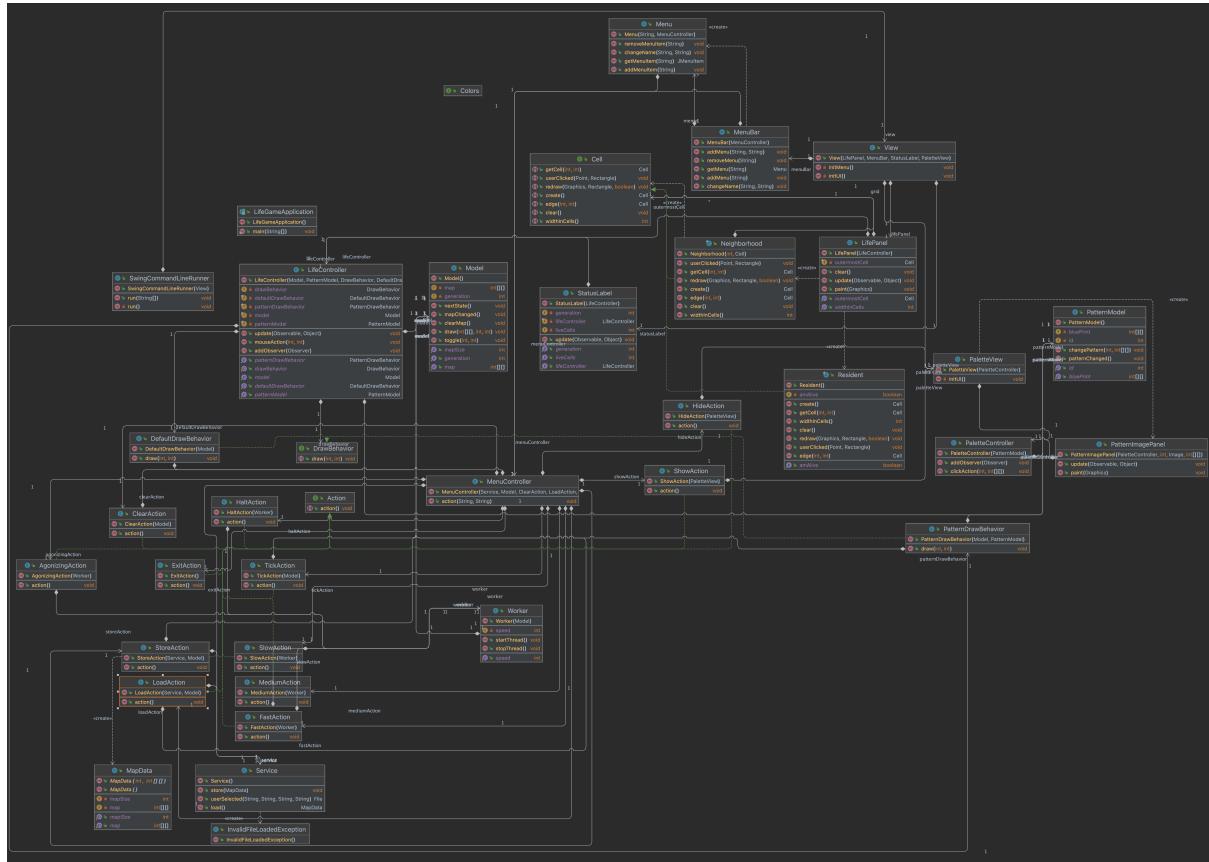
▲ [picture 11] Sequence Diagram (ActionScenario)



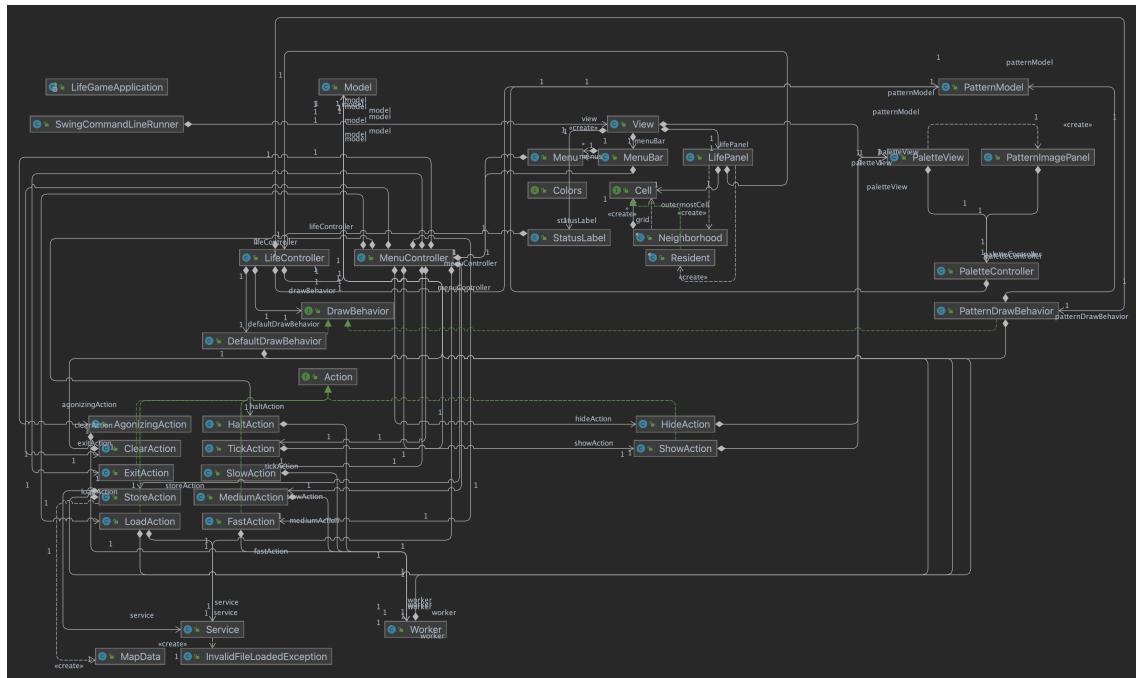
▲ [picture 12] Sequence Diagram (Show Current Status Scenario)

4. 개발

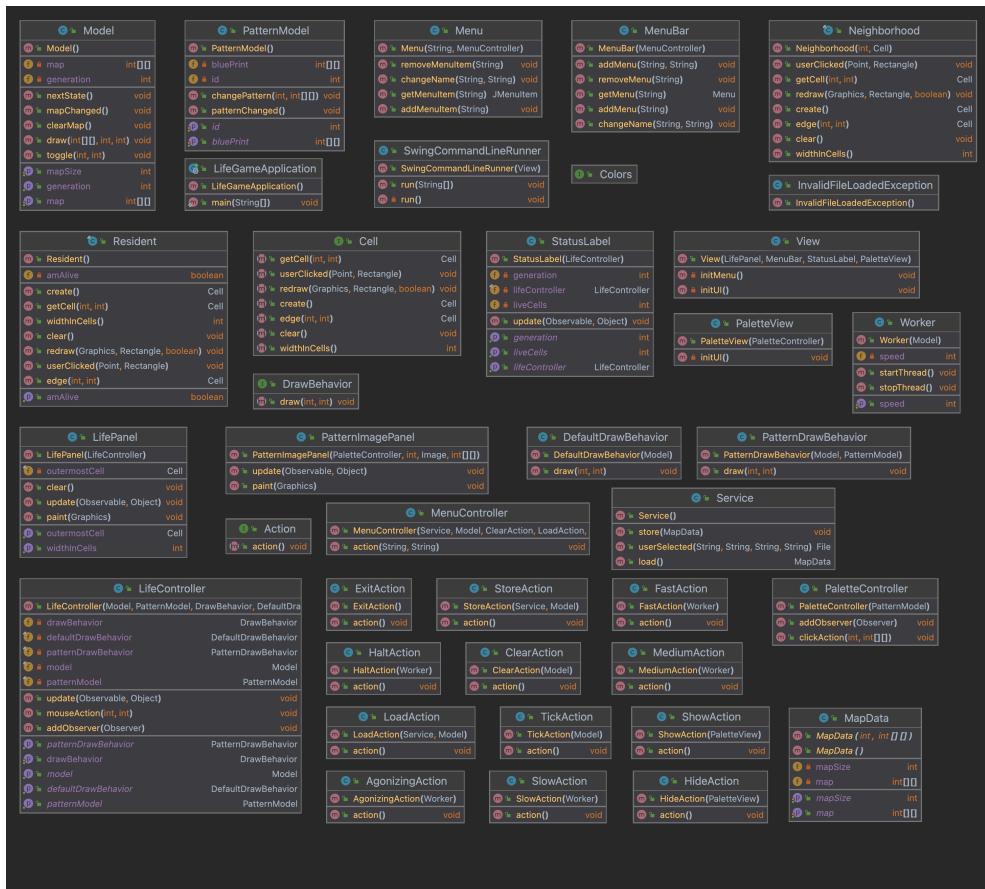
4.1. Class Diagram



▲ [picture 13] Class Diagram



▲ [picture 14] Summarized Class Diagram



▲ [picture 15] Classes

4.2. Gradle

Gradle을 이용해 빌드하고 패키징하였다.

```

plugins {
    id 'java'
    id 'org.springframework.boot' version '2.7.5'
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.fasterxml.jackson.core:jackson-databind'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'org.springframework.restdocs:spring-restdocs-mockMvc'
    testImplementation group: 'org.mockito', name: 'mockito-inline', version: '4.5.1'
}

tasks.named('test') {
    useJUnitPlatform()
}

```

▲ [picture 16] build.gradle

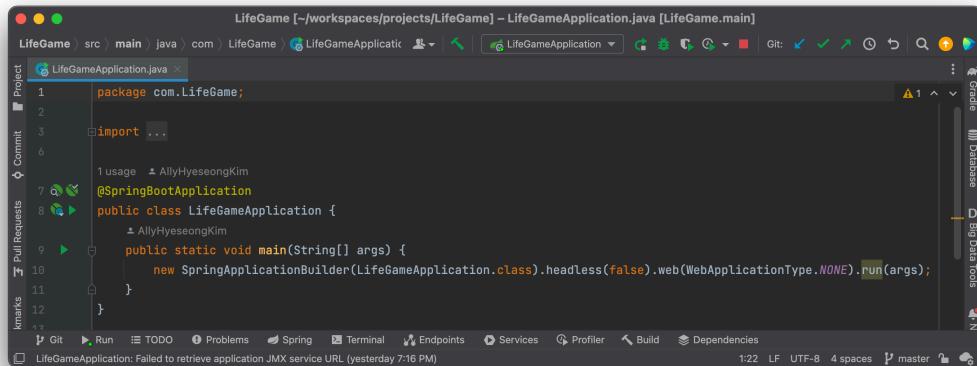
Spring Framework를 이용하기 위해 Spring에 관련된 의존성을 추가하였고, Getter, Setter를 편하게 추가하기 위해 lombok을 의존성에 추가하였다.

Load 및 Save 기능을 구현하기 위해 Java Object를 JSON으로 marshal/unmarshal할 수 있는 ObjectMapper를 이용하기 위한 Jackson의 의존성을 추가하였다.

JUnit을 이용해 테스트할 때 Spring으로 개발한 장점을 살려 쉽게 Mock 객체를 주입할 수 있도록 하기 위해 Mockito 의존성을 추가하였다.

4.3. Spring

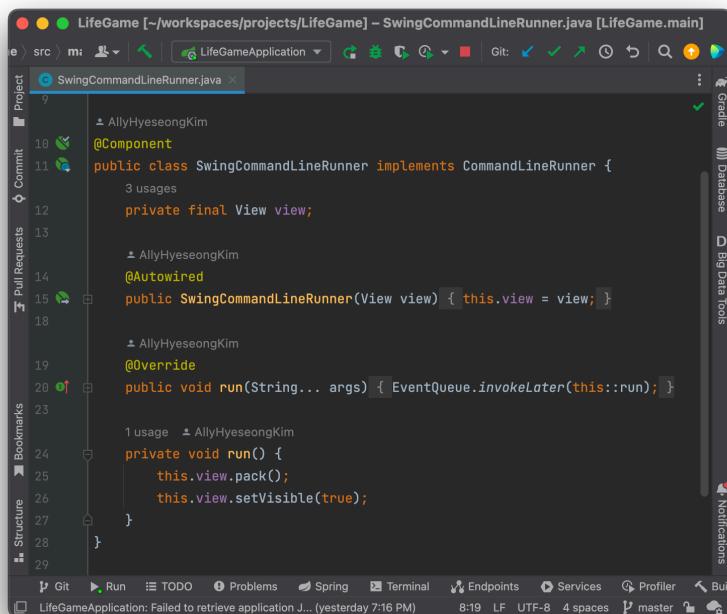
MVC Pattern으로 프로젝트를 리팩토링하면서 객체 간 느슨한 결합과 IoC를 위해 Spring Framework를 활용하여 개발하였다.



```
LifeGame [-/workspaces/projects/LifeGame] - LifeGameApplication.java [LifeGame.main]
package com.LifeGame;
import ...
@usage 盟 AllyHyeseongKim
@SpringBootApplication
public class LifeGameApplication {
    @AllyHyeseongKim
    public static void main(String[] args) {
        new SpringApplicationBuilder(LifeGameApplication.class).headless(false).web(WebApplicationType.NONE).run(args);
    }
}
```

▲ [picture 17] LifeGameApplication class

@SpringBootApplication을 main이 있는 클래스에 붙여주고 main에서 Application을 run 해준다. Spring Web을 사용하지 않기 때문에 WebApplicationType.NONE을 설정해 주었다.



```
LifeGame [-/workspaces/projects/LifeGame] - SwingCommandLineRunner.java [LifeGame.main]
@Component
public class SwingCommandLineRunner implements CommandLineRunner {
    private final View view;
    @Autowired
    public SwingCommandLineRunner(View view) { this.view = view; }

    @Override
    public void run(String... args) { EventQueue.invokeLater(this::run); }

    private void run() {
        this.view.pack();
        this.view.setVisible(true);
    }
}
```

▲ [picture 18] LifeGameApplication class

SwingCommandLineRunner는 CommandLineRunner를 구현하는 클래스이다. 모든 컴포넌트들이 초기화되고 난 후에 메인 JFrame의 setVisible method를 실행하기 위해 작성되었다.

Spring Framework에서 SpringBootApplication이 실행되고 나면 CommandLineRunner를 구현한 컴포넌트를 찾아 run(String... args) method를 찾아 실행시킨다. EventQueue.invokeLater method를 이용해 Spring Component 초기화 작업이 완료되고 난 뒤에 View Component의 setVisible method를 실행해 Swing 화면을 표시한다.

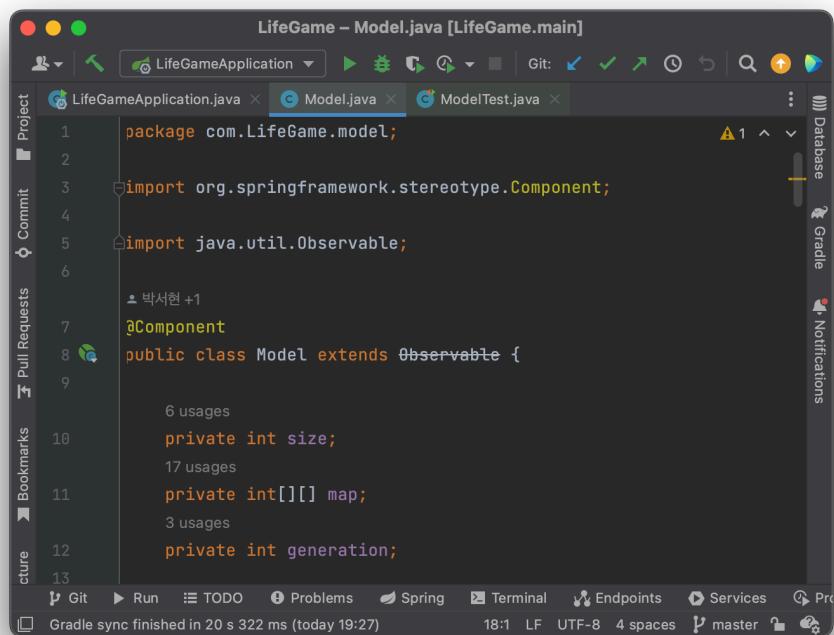
4.4. MVC Architecture Pattern

4.4.1. Model

model은 cell의 정보와 기능을 담고 있는 Model class와 palette에 들어가는 pattern에 대한 정보와 기능을 담고 있는 PatternModel class로 구성되어 있다. 각 model은 Observable class를 상속받아 Observer pattern의 subject 역할을 하여 각 observer에 update된 정보가 notify된다.

4.4.1.1. [Model] Model class

Model class는 Observer pattern의 subject 역할을 하며 Observable class를 상속받고 cell의 정보와 기능을 담고 있다.



```
LifeGame - Model.java [LifeGame.main]
-----
1 package com.LifeGame.model;
2
3 import org.springframework.stereotype.Component;
4
5 import java.util.Observable;
6
7 @Component
8 public class Model extends Observable {
9
10    private int size;
11    private int[][] map;
12    private int generation;
13}
```

▲ [picture 19] Model class (1)

Model class는 2차원 배열 형태의 map과 int형의 size, int형의 generation 변수를 가진다.

```

13
14     12 usages ▲ AllyHyeseongKim
15     public void mapChanged() {
16         setChanged();
17         notifyObservers();
18     }
19     8 usages ▲ AllyHyeseongKim +1
20     public void clearMap() { //map 초기화
21         this.map = new int[this.size][this.size];
22         this.generation = 0;
23         this.mapChanged();
24     }
25     5 usages ▲ AllyHyeseongKim +1
26     public int getMapSize() {
27         return size;
28     }
29     19 usages ▲ 박서현
30     public int[][] getMap() {
31         return this.map;
32     }
33     3 usages ▲ 박서현
34     public int getGeneration() {
35         return this.generation;
36     }

```

▲ [picture 20] Model class (2)

mapChanged를 통해 map이 바뀐 경우 Observer에게 notify할 수 있다.

clearMap을 통해 현재 map과 generation을 초기화시키고 mapchanged를 호출한다. getMapSize에서는 현재 map의 size를 return하고, getMap을 통해 현재 map을 2차원 배열 형태로 반환한다. getGeneration을 통해 현재 세포의 세대 정보를 얻는다.

```

37     public void setMapSize(int n) {
38         this.size = n;
39         this.map = new int[n][n];
40     }
41
42     13 usages ▲ 박서현 +1
43     public void toggle(int x, int y) {
44         // 좌표 설정 방식 - 그냥 배열이랑 똑같이 간다고 가정
45         if (this.map[x][y] == 0) {
46             this.map[x][y] = 1;
47         } else {
48             this.map[x][y] = 0;
49         }
50
51         this.mapChanged();
52     }
53
54     9 usages ▲ AllyHyeseongKim +1
55     public void setMap(int[][] map) {
56         this.map = map;
57
58         this.mapChanged();
59     }
60
61     public void draw(int[][] bluePrint, int x, int y) {
62         for (int i = 0; i < Math.min(bluePrint.length, this.size - x); i++) {
63             System.arraycopy(bluePrint[i], 0, this.map[x + i], y, Math.min(bluePrint[0].length, this.size - y));
64         }
65         this.mapChanged();
66     }

```

▲ [picture 21] Model class (3)

setMapSize로 map의 크기를 설정할 수 있다.

toggle을 통해서 map을 클릭했을 때 살아있는 세포(1)가 있었다면 0으로, 죽은 세포가 있었다면 1로 상태를 변경한다. setMap으로 map을 임의로 설정할 수 있다. draw 통해 팔레트의 패턴을 map에 그릴 수 있다. toggle, setMap, draw는 모두 mapChanged를 호출하여 상태 변화를 notify시킨다.

The screenshot shows the Android Studio interface with the Model.java file open in the main editor. The code implements the nextState() method for the Game of Life. It uses nested loops to iterate through the map, applying rules to each cell based on its neighbors. The code includes logic for birth (state 3), survival (state 2), and death (state 0). It also handles boundary conditions and updates the map and generation count. The Android Studio toolbar and various tool tabs are visible at the bottom.

```
public void nextState() {
    int[] dx = {-1, -1, -1, 0, 1, 1, 1, 0};
    int[] dy = {-1, 0, 1, 1, 0, -1, -1, -1};
    int n = this.map.length;
    int m = this.map[0].length;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            int state = 0;

            for (int k = 0; k < 8; k++) {
                int x = i + dx[k];
                int y = j + dy[k];
                if (x < 0 || x >= n || y < 0 || y >= m) continue;
                if (this.map[x][y] == 1 || this.map[x][y] == 2) state++;
            }

            if (this.map[i][j] == 0 && state == 3) this.map[i][j] = 3;
            if (this.map[i][j] == 1 && (state < 2 || state > 3)) this.map[i][j] = 2;
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            this.map[i][j] %= 2;
        }
    }

    this.generation += 1;
    this.mapChanged();
}
```

▲ [picture 22] Model class (4)

nextState를 호출하여 현재 map에 있는 세포들을 다음 상태로 변화시킨다. 죽은 세포의 이웃 중 세 개가 있으면 그 세포는 살아나고, 살아 있는 세포의 이웃 중 두 개나 세 개가 살아 있다면 해당 세포는 살아있는 상태를 유지한다. 2차원 배열을 순회하며 모든 방향에 대해 탐색하며, 탐색이 끝나고 해당하는 배열의 인덱스에 변화가 필요하다면 0이나 1을 사용하는 대신 2와 3을 사용한 후 모든 탐색이 끝나고 2로 나머지 연산을 한다. nextState를 할 때마다 generation이 증가하고, mapChanged가 호출된다.

4.4.1.2. [Model] PatternModel class

PatternModel class는 Observer pattern의 subject로 Observable class를 상속받고 현재 선택된 패턴의 id와 그림 정보를 가지고 있다.

package com.LifeGame.model;

import lombok.Getter;

import lombok.Setter;

import org.springframework.stereotype.Component;

import java.util.Observable;

▲ AllyHyeseongKim

● @Component

● @Setter @Getter

public class PatternModel extends Observable {

3 usages

private int id = 0;

1 usage

private int[][] bluePrint;

▲ [picture 23] PatternModel class (1)

이 클래스를 Singleton 객체로 생성하기 위해 Spring의 @Component annotation을 통해 Spring Bean에 등록하였고, lombok의 @Getter @Setter annotation을 통해 controller에서 현재 선택된 field들을 설정하고 observer에서 현재 선택된 field 값을 확인할 수 있도록 하였다.

Project

File

View

Code

Toolbars

Help

LifeGame Application

PatternModel.java

4 usages ▲ AllyHyeseongKim

```
public void patternChanged() {
    setChanged();
    notifyObservers();
}
```

4 usages ▲ AllyHyeseongKim

```
public void changePattern(int id, int[][] bluePrint) {
    if (this.id == id) {
        this.id = 0;
    } else {
        this.id = id;
        this.bluePrint = bluePrint;
    }

    this.patternChanged();
}
```

1 A 1 ▾

Gradle

Database

Big Data Tools

Notifications

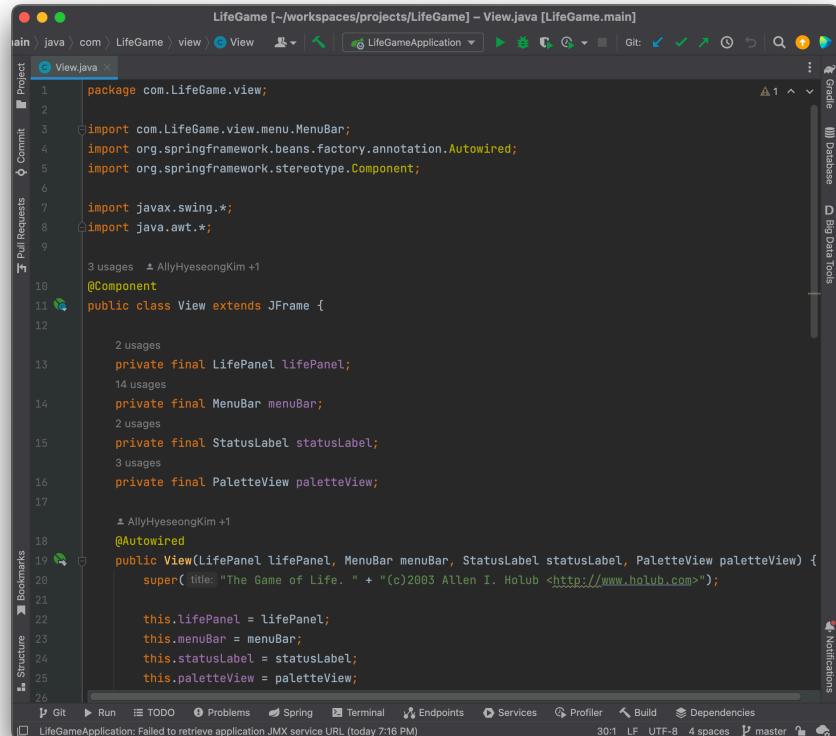
▲ [picture 24] PatternModel class (2)

changePattern method를 호출하여 현재 선택된 패턴을 등록하면 patternChanged method를 호출하여 observer에게 notify하도록 하였다.

4.4.2. View

4.4.2.1. [View] View class

View class는 JFrame을 상속받는다. View 생성 시 menu bar를 등록하고, LifePanel, StatusLabel을 등록하도록 하였다.



The screenshot shows the IntelliJ IDEA interface with the View.java file open. The code defines a View class that extends JFrame. It imports com.LifeGame.view.menu.MenuBar, org.springframework.beans.factory.annotation.Autowired, org.springframework.stereotype.Component, javax.swing.* (specifically JButton), and java.awt.*. The class has a constructor that takes four parameters: LifePanel, MenuBar, StatusLabel, and PaletteView. Inside the constructor, it initializes instance variables: lifePanel, menuBar, statusLabel, and paletteView. The code uses the @Component and @Autowired annotations to register the View class as a Spring Bean and to automatically inject instances of LifePanel, MenuBar, StatusLabel, and PaletteView.

```
package com.LifeGame.view;

import com.LifeGame.view.menu.MenuBar;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.swing.*;
import java.awt.*;

public class View extends JFrame {

    private final LifePanel lifePanel;
    private final MenuBar menuBar;
    private final StatusLabel statusLabel;
    private final PaletteView paletteView;

    public View(LifePanel lifePanel, MenuBar menuBar, StatusLabel statusLabel, PaletteView paletteView) {
        super("The Game of Life. " + "(c)2003 Allen I. Holub <http://www.holub.com>");

        this.lifePanel = lifePanel;
        this.menuBar = menuBar;
        this.statusLabel = statusLabel;
        this.paletteView = paletteView;
    }
}
```

▲ [picture 25] View class (1)

o] 클래스를 Singleton 객체로 생성하기 위해 Spring의 @Component annotation을 통해 Spring Bean에 등록하였고, View에서 사용하는 LifePanel, MenuBar, StatusLabel, PaletteView는 Spring의 @Autowired annotation을 통해 Bean에서 자동으로 주입받도록 하였다.

```

    this.initMenu();
    this.initUI();
}

1 usage  ± AllyHyeseongKim
private void initMenu() {
    this.menuBar.addMenu( name: "Grid", menutem: "Clear");
    this.menuBar.addMenu( name: "Grid", menutem: "Load");
    this.menuBar.addMenu( name: "Grid", menutem: "Store");
    this.menuBar.addMenu( name: "Grid", menutem: "Exit");

    this.menuBar.addMenu( name: "Go", menutem: "Halt");
    this.menuBar.addMenu( name: "Go", menutem: "Tick (Single Step)");
    this.menuBar.addMenu( name: "Go", menutem: "Agonizing");
    this.menuBar.addMenu( name: "Go", menutem: "Slow");
    this.menuBar.addMenu( name: "Go", menutem: "Medium");
    this.menuBar.addMenu( name: "Go", menutem: "Fast");

    this.menuBar.addMenu( name: "Palette", menutem: "Show");
    this.menuBar.addMenu( name: "Palette", menutem: "Hide");
}

```

▲ [picture 26] View class (2)

MenuBar에 들어갈 Menu들을 View 생성 시 만들어지도록 하였다.

```

    private void initUI() {
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.getContentPane().setLayout(new BorderLayout());
        this.setJMenuBar(this.menuBar);
        this.getContentPane().add(this.lifePanel, BorderLayout.CENTER);
        this.getContentPane().add(this.statusLabel, BorderLayout.SOUTH);

        Point point = this.getLocation();
        Dimension dimension = this.getPreferredSize();
        point.x += dimension.width;
        this.paletteView.setLocation(point);
        dimension.width = 100;
        dimension.height = 100 * 6;
        this.paletteView.setPreferredSize(dimension);
    }
}

```

▲ [picture 27] View class (3)

JFrame을 상속받는 View에 메뉴를 가지는 menu bar, cell들을 그리는 LifePanel, 현재 cell의 정보를 가지는 StatusLabel을 등록하였다. 또한, View가 그려지는 창의 바로 오른쪽에 PaletteView가 띄워지도록 설정하였다.

4.4.2.2. [View] LifePanel class

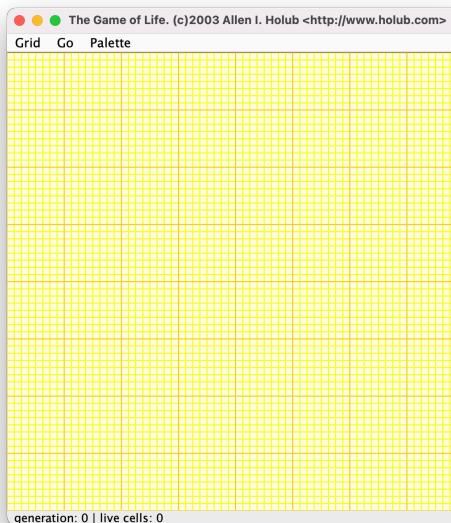
LifePanel class는 JPanel을 상속받고 Observer interface를 구현한다. Cell들이 있는 판을 그리는 panel로 cell을 가지고 있고, LifePanel에서 발생하는 로직은 LifeController로 위임한다.

```

17 8 usages  AllyHyeseongKim
18  @Component
19  public class LifePanel extends JPanel implements Observer {
20      2 usages
21      private final LifeController lifeController;
22      8 usages
23      private final Cell outermostCell;
24
25      AllyHyeseongKim
26      @Autowired
27      public LifePanel(LifeController lifeController) {
28
29          this.lifeController = lifeController;
30
31          this.lifeController.addObserver( this );
32
33          /**
34          * The default height and width of a Neighborhood in cells.
35          * If it's too big, you'll run too slowly because
36          * you have to update the entire block as a unit, so there's more
37          * to do. If it's too small, you have too many blocks to check.
38          * I've found that 8 is a good compromise.
39
40      }
41
42      int DEFAULT_GRID_SIZE = 8;
43      this.outermostCell = new Neighborhood(DEFAULT_GRID_SIZE, new Neighborhood(DEFAULT_GRID_SIZE, new Resident()));

```

▲ [picture 28] LifePanel class (1)



▲ [picture 29] LifeGame

이 클래스를 Singleton 객체로 생성하기 위해 Spring의 `@Component` annotation을 통해 Spring Bean에 등록하였고, `LifePanel`에서 사용하는 `LifeController`, `Cell`은 Spring의 `@Autowired` annotation을 통해 Bean에서 자동으로 주입받도록 하였다.

Cell들이 있는 판은 8*8개의 `Resident` cell이 있는 작은 `Neighborhood` cell이 8*8개 있는 형태로 이루어져 있다.

```
41 * The size of the smallest "atomic" cell---a Resident object.
42 * This size is extrinsic to a Resident (It's passed into the
43 * Resident's "draw yourself" method.
44 */
45
46
47 int DEFAULT_CELL_SIZE = 8;
48 final Dimension PREFERRED_SIZE = new Dimension( width: this.outermostCell.widthInCells() * DEFAULT_CELL_SIZE,
49
50     this.addMouseListener((MouseAdapter) mousePressed(e) -> {
51         Rectangle bounds = getBounds();
52         bounds.x = 0;
53         bounds.y = 0;
54         int pixelsPerCell = bounds.width / (DEFAULT_GRID_SIZE * DEFAULT_GRID_SIZE);
55         int row = e.getPoint().y / pixelsPerCell;
56         int column = e.getPoint().x / pixelsPerCell;
57
58         lifeController.mouseAction(row, column);
59     });
60
61     this.addComponentListener((ComponentAdapter) componentResized(e) -> {
62         Rectangle bounds = getBounds();
63         bounds.height /= getWidthInCells();
64         bounds.height *= getWidthInCells();
65         bounds.width = bounds.height;
66         setBounds(bounds);
67     });
68
69     setBackground(Color.white);
70     setPreferredSize(PREFERRED_SIZE);
71     setMaximumSize(PREFERRED_SIZE);
72     setMinimumSize(PREFERRED_SIZE);
73     setOpaque(true);
74
75 }
76
77
78
79 }
```

▲ [picture 30] LifePanel class (2)

LifePanel에서 Cell을 누를 경우, MouseListener로 눌린 Cell의 좌표 위치를 계산하여 LifeController의 mouseAction method를 호출하도록 하였다. 모델의 조작 및 실제 로직은 LifeController가 담당하도록 하여 책임을 분리하고 느슨한 결합 구조로 구성하였다.

마우스로 JFrame 창의 크기를 조절하는 경우 LifePanel 그래픽 크기도 자동으로 조절해 주기 위해 ComponentListener를 붙이고 ComponentAdapter를 이용해 componentResized 이벤트가 발생할 때마다 bound를 재설정하도록 하였다.

```

    /**
     * Override paint to ask the outermost Neighborhood
     * (and any subcells) to draw themselves recursively.
     * All knowledge of screen size is also encapsulated.
     * (The size is passed into the outermost <code>Cell</code>.)
     */
    public void paint(Graphics g) {
        Rectangle panelBounds = getBounds();

        // The panel bounds is relative to the upper-left
        // corner of the screen. Pretend that it's at (0,0)
        panelBounds.x = 0;
        panelBounds.y = 0;
        this.outermostCell.redraw(g, panelBounds, drawAll: true); //={Universe.redraw}
    }

    public Cell getOutermostCell() { return this.outermostCell; }

    public int getWidthInCells() { return this.outermostCell.widthInCells(); }

    public void clear() { this.outermostCell.clear(); }

```

▲ [picture 31] LifePanel class (3)

paint method는 기존에 있던 코드를 그대로 사용하였다. DFS 알고리즘을 이용해 각 Resident의 redraw method를 실행하여 화면을 업데이트한다. clear, getWidthInCells method도 마찬가지로 DFS 알고리즘을 이용하는 방식으로 동작한다.

```

    @Override
    public void update(Observer o, Object arg) {
        if (o instanceof Model) {
            Rectangle bounds = this.getBounds();
            bounds.x = 0;
            bounds.y = 0;
            int pixelsPerCell = bounds.width / 64;
            this.clear();
            int[][] cells = ((Model) o).getMap();
            for (int i = 0; i < 64; i++) {
                for (int j = 0; j < 64; j++) {
                    if (cells[i][j] == 1) {
                        this.outermostCell.userClicked(new Point(x * pixelsPerCell, y * i * pixelsPerCell), bounds);
                    }
                }
            }
            this.repaint();
        }
    }

```

▲ [picture 32] LifePanel class (4)

java.util.Observer interface를 구현하여 override한 method로, Model이 업데이트 될 때마다 notify 받고 화면에 Model대로 그려주도록 하였다. Model과 View에서 map 배열을 저장하는 방식이 다르기 때문에 이를 변환하는 연산이 포함되며, 화면에 그리는 기능은 원래 Cell interface에 정의되어 있던 userClicked method를 이용하였다.

4.4.2.3. [View] Cell interface

기존 코드를 그대로 활용하여 개발하였으며, Neighborhood와 Resident가 Cell Interface를 구현한다.

4.2.2.4. [View] Neighborhood class

Cell interface를 구현한 클래스로, 하위 Cell들의 묶음 역할을 한다.

The screenshot shows a Java code editor with the file `Neighborhood.java` open. The code defines a class `Neighborhood` that implements the `Cell` interface. It contains fields for a grid size and a grid of cells, and methods for creating a new neighborhood and checking if it is active. The code is annotated with Javadoc comments and includes usage statistics for each method and field.

```
21 3 usages ▲ AllyHyeseongKim
22 public final class Neighborhood implements Cell {
23
24     /** Returns true only if none of the cells in the Neighborhood ...*/
25     3 usages
26     private boolean amActive = false;
27
28     /** The actual grid of Cells contained within this neighborhood. */
29     9 usages
30     private final Cell[][] grid;
31
32     /** The neighborhood is square, so gridSize is both the horizontal ...*/
33     12 usages
34     private final int gridSize;
35
36     /** Create a new Neighborhood containing gridSize-by-gridSize ...*/
37
38     3 usages ▲ AllyHyeseongKim
39     public Neighborhood(int gridSize, Cell prototype) {
40         this.gridSize = gridSize;
41         this.grid = new Cell[gridSize][gridSize];
42
43         for (int row = 0; row < gridSize; ++row)
44             for (int column = 0; column < gridSize; ++column)
45                 grid[row][column] = prototype.create();
46     }
47
48
49
50
51
52
53
54 }
```

▲ [picture 33] Neighborhood class (1)

`boolean`형 `amActive` 필드는 현재 `Cell`의 활성화 여부를 저장하며, 내부에 활성화 된 `Resident`가 하나라도 있는 경우 활성화 된 것으로 본다. 이후 `redraw` method를 호출하면 `amActive` 필드를 참조하여 화면에 표시한다.

`grid` 필드는 하위 `Cell`들을 저장하는 필드이다.

`gridSize` 필드는 `grid` 이차원 배열의 크기를 결정하는 필드로 생성자에서 참조한다.

생성자에서는 `gridSize`대로 `grid`를 초기화하고 `parameter`로 받은 `prototype`을 복사하여 `grid`의 모든 칸을 채운다.

```

    public void redraw(Graphics g, Rectangle here, boolean drawAll) {
        // If the current neighborhood is stable (nothing changed
        // in the last transition stage), then there's nothing
        // to do. Just return. Otherwise, update the current block
        // and all sub-blocks. Since this algorithm is applied
        // recursively to subblocks, only those blocks that actually
        // need to update will actually do so.

        int compoundWidth = here.width;
        Rectangle subcell = new Rectangle(here.x, here.y, width: here.width / gridSize, height: here.height / gridSize);

        // Check to see if we can paint. If not, just return. If
        // so, actually wait for permission (in case there's
        // a race condition, then paint.

        for (int row = 0; row < gridSize; ++row) {
            for (int column = 0; column < gridSize; ++column) {
                grid[row][column].redraw(g, subcell, drawAll); // {=Neighborhood.redraw3}
                subcell.translate(subcell.width, dy: 0);
            }
            subcell.translate(-compoundWidth, subcell.height);
        }

        g = g.create();
        g.setColor(Colors.LIGHT_ORANGE);
        g.drawRect(here.x, here.y, here.width, here.height);

        if (amActive) {
            g.setColor(Color.BLUE);
            g.drawRect(x: here.x + 1, y: here.y + 1, width: here.width - 2, height: here.height - 2);
        }

        g.dispose();
    }

```

▲ [picture 34] Neighborhood class (2)

redraw method는 LifePanel에서 자신에게 맞는 위치에 칸을 그리고 하위 Cell들을 DFS 알고리즘으로 순회하며 모든 하위 Cell들의 redraw method를 호출한다. amActive 필드의 값을 참조하여 칸 내부 색을 지정하여 Live Cell을 표시한다.

```

    public Cell edge(int row, int column) {
        assert (row == 0 || row == gridSize - 1) || (column == 0 || column == gridSize - 1) : "central cell requested from edge()";
        return grid[row][column];
    }

    /** Notification of a mouse click. The point is relative to the ... */
    public void userClicked(Point here, Rectangle surface) {
        int pixelsPerCell = surface.width / gridSize;
        int row = here.y / pixelsPerCell;
        int column = here.x / pixelsPerCell;
        int rowOffset = here.y % pixelsPerCell;
        int columnOffset = here.x % pixelsPerCell;

        Point position = new Point(columnOffset, rowOffset);
        Rectangle subcell = new Rectangle(x: 0, y: 0, pixelsPerCell, pixelsPerCell);

        grid[row][column].userClicked(position, subcell);
        amActive = true;
    }

    public int widthInCells() { return gridSize * grid[0][0].widthInCells(); }

    public void clear() {
        for (int row = 0; row < gridSize; ++row) {
            for (int column = 0; column < gridSize; ++column) {
                grid[row][column].clear();
            }
        }
        amActive = false;
    }

```

▲ [picture 35] Neighborhood class (3)

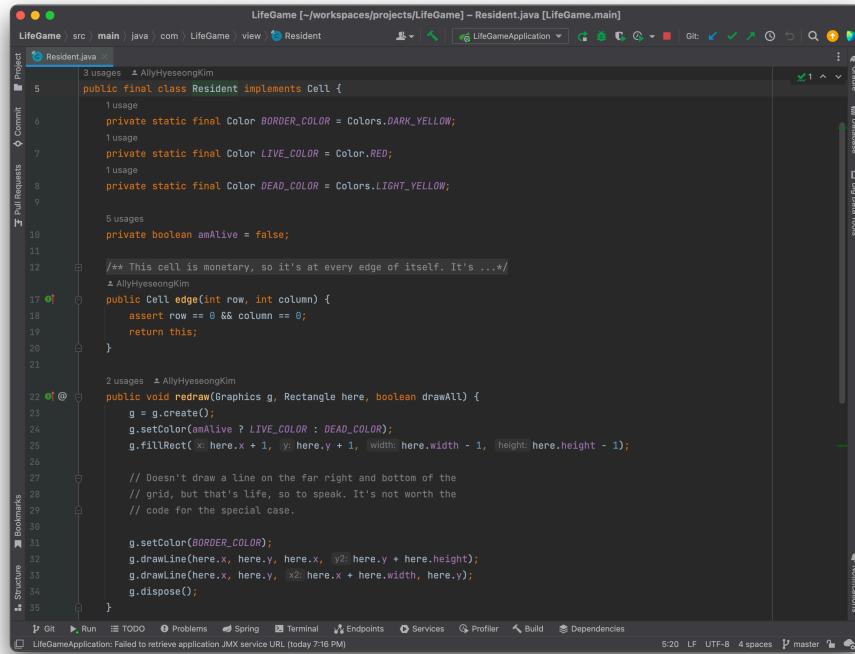
edge method는 row, column에 위치한 하위 Cell 인스턴스를 반환하는 getter로서 기능한다.

userClicked method는 화면에 클릭 이벤트가 발생했을 때 화면의 굱셀값을 토대로 활성화해야 하는 Resident를 찾아 해당 Resident의 amActive를 true로 설정한다.

widthInCells method는 해당 Cell이 화면에서 차지하는 width 굱셀 값을 반환한다.

clear method는 DFS 알고리즘을 이용해 모든 하위 Cell들을 순회하며 amActive 값을 false로 업데이트한다.

4.4.2.5. [View] Resident class



```
public final class Resident implements Cell {
    private static final Color BORDER_COLOR = Colors.DARK_YELLOW;
    private static final Color LIVE_COLOR = Color.RED;
    private static final Color DEAD_COLOR = Colors.LIGHT_YELLOW;

    private boolean amAlive = false;

    /** This cell is monitory, so it's at every edge of itself. It's ...*/
    public Cell edge(int row, int column) {
        assert row == 0 && column == 0;
        return this;
    }

    public void redraw(Graphics g, Rectangle here, boolean drawAll) {
        g = g.create();
        g.setColor(amAlive ? LIVE_COLOR : DEAD_COLOR);
        g.fillRect(here.x + 1, here.y + 1, width: here.width - 1, height: here.height - 1);

        // Doesn't draw a line on the far right and bottom of the
        // grid, but that's life, so to speak. It's not worth the
        // code for the special case.

        g.setColor(BORDER_COLOR);
        g.drawLine(here.x, here.y, here.x, y2: here.y + here.height);
        g.drawLine(here.x, here.y, x2: here.x + here.width, here.y);
        g.dispose();
    }

    public void clear() {
        clear(this);
    }
}
```

▲ [picture 36] Resident class (1)

edge method는 자기 자신을 반환한다.

redraw method는 amAlive 필드 값에 따라 화면의 적절한 위치에 붉은색 혹은 노란색 정사각형을 그리는 기능을 한다.

4.4.2.7. [View] Menu class

Menu class는 JMenu class를 상속받는 클래스로, 상단 메뉴바의 메뉴 내 기능 하나하나를 표시한다.

```

public class Menu extends JMenu {
    private String name;
    private final MenuController menuController;
    private final HashMap<String, JMenuItem> menuItems = new HashMap<>();

    public Menu(String name, MenuController menuController) {
        super(name);
        this.name = name;
        this.menuController = menuController;
    }

    public void addMenuItem(String name) {
        JMenuItem menuItem = new JMenuItem(name);
        menuItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JMenuItem menuItem = (JMenuItem) e.getSource();
                JPopupMenu popupMenu = (JPopupMenu) menuItem.getParent();

                menuController.action(((JMenu) popupMenu.getInvoker()).getActionCommand(), e.getActionCommand());
            }
        });
        this.add(menuItem);
        this.menuItems.put(name, menuItem);
    }
}

```

▲ [picture 37] Menu class (1)

addMenuItem은 메뉴 내부에 JMenuItem을 등록하는 기능을 한다. 각 JMenuItem은 ActionListener를 통해 클릭 이벤트를 받고, MenuController에게 기능을 위임한다.

```

public void removeMenuItem(String name) {
    this.remove(this.menuItems.get(name));
    this.menuItems.remove(name);
}

public void changeName(String name, String newName) {
    this.menuItems.get(name).setText(newName);
    this.menuItems.put(newName, this.menuItems.get(name));
    this.menuItems.remove(name);
}

public JMenuItem getMenuItem(String name) { return this.menuItems.get(name); }
}

```

▲ [picture 38] Menu class (2)

removeMenuItem, changeName, getMenuItem은 각각 메뉴 내부 JMenuItem을 제거, 수정, get 하는 method 들이며, 기존 코드에서 추가로 작성한 기능이다.

4.4.2.8. [View] MenuBar class

MenuBar Class는 JMenuBar 클래스를 상속한다.

```
10 usages ▲ AllyHyeseongKim
11 ④ @Component
12
13 public class MenuBar extends JMenuBar {
14
15     private final HashMap<String, Menu> menus = new HashMap<>();
16
17     private final MenuController menuController;
18
19     public MenuBar(MenuController menuController) {
20         super();
21         this.menuController = menuController;
22     }
23
24     public void addMenu(String name) {
25         Menu menu = new Menu(name, menuController);
26         this.add(menu);
27         this.menus.put(name, menu);
28     }
29
30     public void addMenu(String name, String menuItem) {
31         try {
32             this.menus.get(name).addMenuItem(menuItem);
33         } catch (NullPointerException e) {
34             Menu menu = new Menu(name, menuController);
35             menu.addMenuItem(menuItem);
36             this.add(menu);
37             this.menus.put(name, menu);
38         }
39     }
40 }
```

▲ [picture 39] MenuBar class (1)

menus 필드는 하위 Menu들을 담고 있는 필드이다.

이 클래스를 Singleton 객체로 생성하기 위해 Spring의 @Component annotation을 통해 Spring Bean에 등록하였고, menuController는 MenuController Bean을 주입 받는 필드이다.

addMenu method는 method overloading을 이용해 작성하였고, Menu만 생성하여 추가하는 method와 Menu 내부에 MenuItem을 추가하는 method 두 가지가 존재한다.

```

1 usage 盟 AllyHyeseongKim
public void removeMenu(String name) {
    this.remove(this.menus.get(name));
    this.menus.remove(name);
}

1 usage 盟 AllyHyeseongKim
public void changeName(String name, String newName) {
    this.menus.get(name).setText(newName);
    this.menus.put(newName, this.menus.get(name));
    this.menus.remove(name);
}

6 usages 盟 AllyHyeseongKim
public Menu getMenu(String name) { return this.menus.get(name); }
}

```

▲ [picture 40] MenuBar class (2)

Menu 클래스와 마찬가지로, removeMenu, changeName, getMenu method를 갖고 있고, 각각 메뉴 제거, 메뉴 수정, get 기능을 수행한다.

4.4.2.9. [View] PaletteView class

PaletteView Class는 JFrame을 상속받는다.

```

14 盟 AllyHyeseongKim
@Component
public class PaletteView extends JFrame {

15 盟 AllyHyeseongKim
16
17 2 usages
private final PaletteController paletteController;

18
19 盟 AllyHyeseongKim
@Autowired
20 public PaletteView(PaletteController paletteController) throws IOException {
21     super("Palette");
22
23     this.paletteController = paletteController;
24
25     this.initUI();
26 }

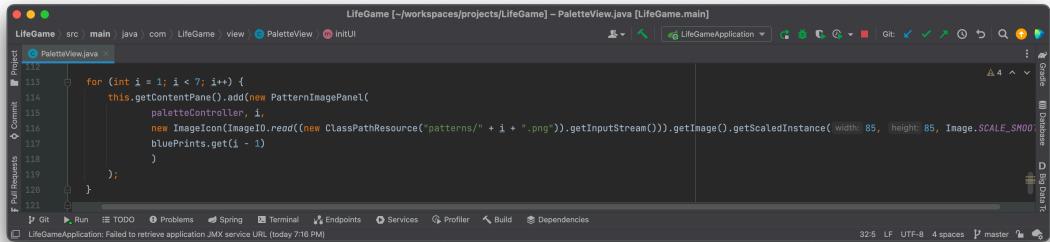
27
28 1 usage 盟 AllyHyeseongKim
private void initUI() throws IOException {
29     this.setLayout(new GridLayout( rows: 6, cols: 1));
30
31     ArrayList<int[][]> bluePrints = new ArrayList<>();
32     bluePrints.add(new int[][]{
33         {0, 1, 0, 0, 1, 0},
34         {0, 0, 1, 1, 0, 0},
35         {1, 0, 1, 1, 1, 0}
36     });
37     bluePrints.add(new int[][]{
38         {0, 1, 0, 0, 1, 1},
39         {0, 0, 0, 1, 0, 0},
40         {0, 1, 0, 0, 0, 0}
41     });
42     bluePrints.add(new int[][]{
43         {0, 0, 0, 0, 0, 1, 1, 0, 0, 0}
44     });

```

▲ [picture 41] PaletteView class (1)

@Component Annotation을 이용해 Singleton으로 만들어 Spring Bean Container에 Bean으로 등록하고, @Autowired Annotation을 통해 PaletteController의 존성을 주입받는다.

InitUI method에서 int형 2차원 배열을 담은 ArrayList를 작성하였다.



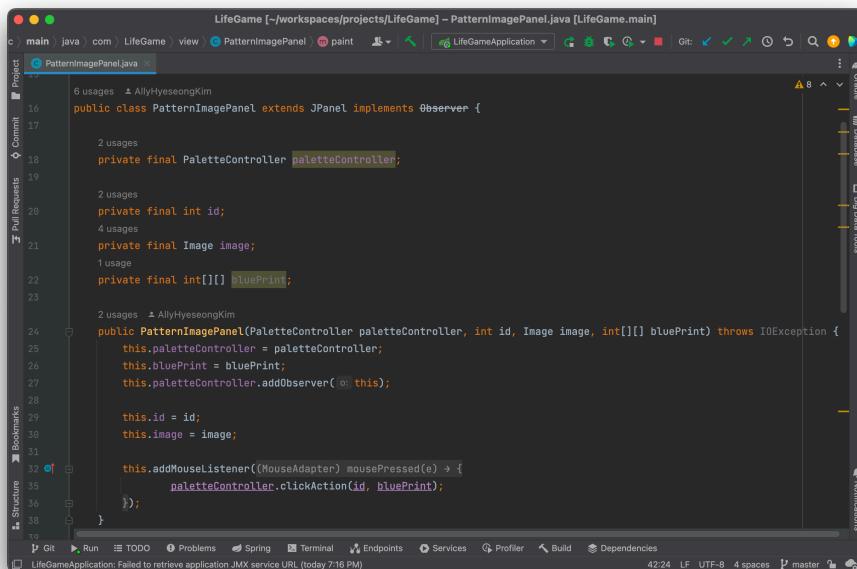
```
for (int i = 1; i < 7; i++) {
    this.getContentPane().add(new PatternImagePanel(
        paletteController, i,
        new ImageIcon(new ClassPathResource("patterns/" + i + ".png").getInputStream()).getImage().getScaledInstance(width: 85, height: 85, Image.SCALE_SMooth));
    bluePrints.get(i - 1)
}
```

▲ [picture 42] PaletteView class (2)

bluePrints ArrayList내 각각의 2차원배열 패턴과 패턴에 대응하는 이미지를 parameter로 넘겨 PatternImagePanel을 생성하도록 하였다.

4.4.2.10. [View] PatternImagePanel class

PatternImagePanel은 JPanel을 상속하고, java.util.Observer interface를 구현한다.



```
public class PatternImagePanel extends JPanel implements Observer {
    private final PaletteController paletteController;
    private final int id;
    private final Image image;
    private final int[][] bluePrint;

    public PatternImagePanel(PaletteController paletteController, int id, Image image, int[][] bluePrint) throws IOException {
        this.paletteController = paletteController;
        this.bluePrint = bluePrint;
        this.paletteController.addObserver(this);

        this.id = id;
        this.image = image;

        this.addMouseListener((MouseAdapter) e -> {
            paletteController.clickAction(id, bluePrint);
        });
    }
}
```

▲ [picture 43] PatternImagePanel class (1)

PaletteController Bean의 존성을 주입받고, 마우스 이벤트 발생 시 작동하는 로직을 PaletteController에 위임한다.

```

    40
    41     @Override
    42     public void paint(Graphics g) {
    43         super.paint(g);
    44         Graphics2D g2d = (Graphics2D) g;
    45         int x = (this.getWidth() - image.getWidth(observer: null)) / 2;
    46         int y = (this.getHeight() - image.getHeight(observer: null)) / 2;
    47         g2d.drawImage(image, x, y, observer: null);
    48     }
    49
    50     @Override
    51     public void update(Observer o, Object arg) {
    52         if (o instanceof PatternModel) {
    53             if (((PatternModel) o).getId() == this.id) {
    54                 this.setBorder(BorderFactory.createLineBorder(Color.PINK, thickness: 3));
    55             } else {
    56                 this.setBorder(null);
    57             }
    58         }
    59     }

```

▲ [picture 44] PatternImagePanel class (2)

paint method는 화면에 패널을 그리는 기능을 수행한다.

update method는 PatternModel이 변화할 때마다 호출되어 해당 Pattern이 활성화되었는지를 체크하여 Border에 반영한다.

4.4.2.11. [View] StatusLabel class

StatusLabel은 JLabel을 상속하고 java.util.Observer interface를 구현한다.

```

    13
    14     @Component
    15     @Getter
    16     public class StatusLabel extends JLabel implements Observer {
    17
    18         2 usages
    19         private final LifeController lifeController;
    20
    21         3 usages
    22         private int generation = 0;
    23         4 usages
    24         private int liveCells = 0;
    25
    26         1 AllyHyeseongKim
    27         @Autowired
    28         public StatusLabel(LifeController lifeController) {
    29             this.lifeController = lifeController;
    30             this.lifeController.addObserver( o: this);
    31
    32             this.setText(" generation: " + this.generation + " | live cells: " + this.liveCells);
    33         }

```

▲ [picture 45] StatusLabel class (1)

@Component Annotation을 이용해 Spring Bean으로 등록하여 Singleton 객체로 구현하였다.

LifeController Bean의 존성을 주입받아 addObserver method를 통해 Model에 옵저버로 등록한다.

```

30
31     @Override
32     public void update(Observer o, Object arg) {
33         if (o instanceof Model) {
34             this.generation = ((Model) o).getGeneration();
35             this.liveCells = 0;
36             int[][] map = ((Model) o).getMap();
37             for (int i = 0; i < ((Model) o).getMapSize(); i++) {
38                 for (int j = 0; j < ((Model) o).getMapSize(); j++) {
39                     this.liveCells += map[i][j];
40                 }
41             }
42             this.setText(" generation: " + this.generation + " | live cells: " + this.liveCells);
43         }
44     }

```

▲ [picture 46] StatusLabel class (2)

update method는 Model이 변화할 때마다 실행되어 Model의 map에서 Live Cell의 개수를 count하고, generation 정보와 함께 Text를 업데이트한다.

4.4.3. Controller

4.4.3.1. [Controller] Action interface

```

1 package com.LifeGame.controller.action;
2
3 public interface Action {
4     void action();
5 }

```

▲ [picture 47] Action interface

Command Pattern을 이용해 Menu를 클릭할 때 동작하는 Command들의 프로토타입을 정의한 interface이다. 이후 다양성을 활용해 MenuController에서 적절히 로직을 수행한다.

4.4.3.2. [Controller] ClearAction class

ClearAction class는 Action interface를 구현한다.

```

1  Action.java x  ClearAction.java x
2
3  4 usages  AllyHyeseongKim
4  @Component
5  public class ClearAction implements Action {
6
7      2 usages
8      private final Model model;
9
10     AllyHyeseongKim
11     @Autowired
12     public ClearAction(Model model) { this.model = model; }
13
14     AllyHyeseongKim
15     @Override
16     public void action() { this.model.clearMap(); }
17
18  }
19
20
21
22

```

▲ [picture 48] ClearAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Model Bean을 주입받았다.

action method를 실행하면 map의 clearModel method를 호출하여 map을 초기화한다.

4.4.3.3. [Controller] LoadAction class

LoadAction class는 Action interface를 구현한다.

```

1  controller > action > LoadAction x  LifeGameApplication x
2
3  4 usages  AllyHyeseongKim +1 *
4  @Component
5  public class LoadAction implements Action {
6
7      2 usages
8      private final Service service;
9      5 usages
10     private final Model model;
11
12     AllyHyeseongKim
13     @Autowired
14     public LoadAction(Service service, Model model) {
15         this.service = service;
16         this.model = model;
17     }
18
19     AllyHyeseongKim +1 *
20     @Override
21     public void action() {
22         try {
23             MapData mapData = this.service.load();
24             this.model.clearMap();
25             this.model.setMapSize(mapData.getMapSize());
26             this.model.setMap(mapData.getMap());
27         } catch (InvalidFileLoadedException e) {
28             this.model.clearMap();
29             JOptionPane.showMessageDialog(null, "Read Failed!", "The Game of Life", JOptionPane.ERROR_MESSAGE);
30         }
31     }
32
33
34
35
36
37
38
39

```

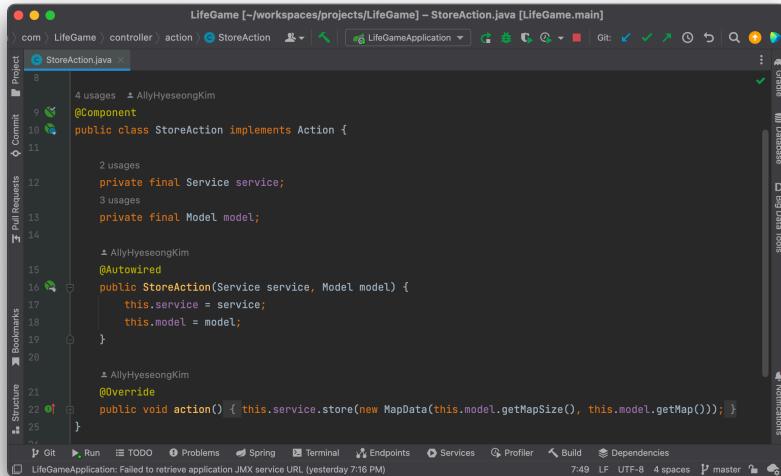
▲ [picture 49] LoadAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Model, Service Bean을 주입받았다.

action method를 실행하면 Service로부터 Map Data를 읽어 Model에 업데이트한다.

4.4.3.4. [Controller] StoreAction class

StoreAction class는 Action interface를 구현한다.



```
LifeGame [-/workspaces/projects/LifeGame] - StoreAction.java [LifeGame.main]
com.LifeGame.controller.action.StoreAction
  ↳ LifeGameApplication

StoreAction.x
Project Commit Pull Requests Bookmarks Structure Git Run TODO Problems Spring Terminal Endpoints Services Profiler Build Dependencies Notifications
8
  9 4 usages ▾ AllyHyeseongKim
  10 @Component
  11   public class StoreAction implements Action {
  12
  13     2 usages
  14     private final Service service;
  15     3 usages
  16     private final Model model;
  17
  18     ▾ AllyHyeseongKim
  19     @Autowired
  20     public StoreAction(Service service, Model model) {
  21       this.service = service;
  22       this.model = model;
  23     }
  24
  25     ▾ AllyHyeseongKim
  26     @Override
  27     public void action() { this.service.store(new MapData(this.model.getMapSize(), this.model.getMap())); }
  28
  29
  30
  31
  32
  33
  34
  35
  36
  37
  38
  39
  40
  41
  42
  43
  44
  45
  46
  47
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
  1000
  1001
  1002
  1003
  1004
  1005
  1006
  1007
  1008
  1009
  1010
  1011
  1012
  1013
  1014
  1015
  1016
  1017
  1018
  1019
  1020
  1021
  1022
  1023
  1024
  1025
  1026
  1027
  1028
  1029
  1030
  1031
  1032
  1033
  1034
  1035
  1036
  1037
  1038
  1039
  1040
  1041
  1042
  1043
  1044
  1045
  1046
  1047
  1048
  1049
  1050
  1051
  1052
  1053
  1054
  1055
  1056
  1057
  1058
  1059
  1060
  1061
  1062
  1063
  1064
  1065
  1066
  1067
  1068
  1069
  1070
  1071
  1072
  1073
  1074
  1075
  1076
  1077
  1078
  1079
  1080
  1081
  1082
  1083
  1084
  1085
  1086
  1087
  1088
  1089
  1090
  1091
  1092
  1093
  1094
  1095
  1096
  1097
  1098
  1099
  1100
  1101
  1102
  1103
  1104
  1105
  1106
  1107
  1108
  1109
  1110
  1111
  1112
  1113
  1114
  1115
  1116
  1117
  1118
  1119
  1120
  1121
  1122
  1123
  1124
  1125
  1126
  1127
  1128
  1129
  1130
  1131
  1132
  1133
  1134
  1135
  1136
  1137
  1138
  1139
  1140
  1141
  1142
  1143
  1144
  1145
  1146
  1147
  1148
  1149
  1150
  1151
  1152
  1153
  1154
  1155
  1156
  1157
  1158
  1159
  1160
  1161
  1162
  1163
  1164
  1165
  1166
  1167
  1168
  1169
  1170
  1171
  1172
  1173
  1174
  1175
  1176
  1177
  1178
  1179
  1180
  1181
  1182
  1183
  1184
  1185
  1186
  1187
  1188
  1189
  1190
  1191
  1192
  1193
  1194
  1195
  1196
  1197
  1198
  1199
  1200
  1201
  1202
  1203
  1204
  1205
  1206
  1207
  1208
  1209
  1210
  1211
  1212
  1213
  1214
  1215
  1216
  1217
  1218
  1219
  1220
  1221
  1222
  1223
  1224
  1225
  1226
  1227
  1228
  1229
  1230
  1231
  1232
  1233
  1234
  1235
  1236
  1237
  1238
  1239
  1240
  1241
  1242
  1243
  1244
  1245
  1246
  1247
  1248
  1249
  1250
  1251
  1252
  1253
  1254
  1255
  1256
  1257
  1258
  1259
  1260
  1261
  1262
  1263
  1264
  1265
  1266
  1267
  1268
  1269
  1270
  1271
  1272
  1273
  1274
  1275
  1276
  1277
  1278
  1279
  1280
  1281
  1282
  1283
  1284
  1285
  1286
  1287
  1288
  1289
  1290
  1291
  1292
  1293
  1294
  1295
  1296
  1297
  1298
  1299
  1300
  1301
  1302
  1303
  1304
  1305
  1306
  1307
  1308
  1309
  1310
  1311
  1312
  1313
  1314
  1315
  1316
  1317
  1318
  1319
  1320
  1321
  1322
  1323
  1324
  1325
  1326
  1327
  1328
  1329
  1330
  1331
  1332
  1333
  1334
  1335
  1336
  1337
  1338
  1339
  1340
  1341
  1342
  1343
  1344
  1345
  1346
  1347
  1348
  1349
  1350
  1351
  1352
  1353
  1354
  1355
  1356
  1357
  1358
  1359
  1360
  1361
  1362
  1363
  1364
  1365
  1366
  1367
  1368
  1369
  1370
  1371
  1372
  1373
  1374
  1375
  1376
  1377
  1378
  1379
  1380
  1381
  1382
  1383
  1384
  1385
  1386
  1387
  1388
  1389
  1390
  1391
  1392
  1393
  1394
  1395
  1396
  1397
  1398
  1399
  1400
  1401
  1402
  1403
  1404
  1405
  1406
  1407
  1408
  1409
  1410
  1411
  1412
  1413
  1414
  1415
  1416
  1417
  1418
  1419
  1420
  1421
  1422
  1423
  1424
  1425
  1426
  1427
  1428
  1429
  1430
  1431
  1432
  1433
  1434
  1435
  1436
  1437
  1438
  1439
  1440
  1441
  1442
  1443
  1444
  1445
  1446
  1447
  1448
  1449
  1450
  1451
  1452
  1453
  1454
  1455
  1456
  1457
  1458
  1459
  1460
  1461
  1462
  1463
  1464
  1465
  1466
  1467
  1468
  1469
  1470
  1471
  1472
  1473
  1474
  1475
  1476
  1477
  1478
  1479
  1480
  1481
  1482
  1483
  1484
  1485
  1486
  1487
  1488
  1489
  1490
  1491
  1492
  1493
  1494
  1495
  1496
  1497
  1498
  1499
  1500
  1501
  1502
  1503
  1504
  1505
  1506
  1507
  1508
  1509
  1510
  1511
  1512
  1513
  1514
  1515
  1516
  1517
  1518
  1519
  1520
  1521
  1522
  1523
  1524
  1525
  1526
  1527
  1528
  1529
  1530
  1531
  1532
  1533
  1534
  1535
  1536
  1537
  1538
  1539
  1540
  1541
  1542
  1543
  1544
  1545
  1546
  1547
  1548
  1549
  1550
  1551
  1552
  1553
  1554
  1555
  1556
  1557
  1558
  1559
  1560
  1561
  1562
  1563
  1564
  1565
  1566
  1567
  1568
  1569
  1570
  1571
  1572
  1573
  1574
  1575
  1576
  1577
  1578
  1579
  1580
  1581
  1582
  1583
  1584
  1585
  1586
  1587
  1588
  1589
  1590
  1591
  1592
  1593
  1594
  1595
  1596
  1597
  1598
  1599
  1600
  1601
  1602
  1603
  1604
  1605
  1606
  1607
  1608
  1609
  1610
  1611
  1612
  1613
  1614
  1615
  1616
  1617
  1618
  1619
  1620
  1621
  1622
  1623
  1624
  1625
  1626
  1627
  1628
  1629
  1630
  1631
  1632
  1633
  1634
  1635
  1636
  1637
  1638
  1639
  1640
  1641
  1642
  1643
  1644
  1645
  1646
  1647
  1648
  1649
  1650
  1651
  1652
  1653
  1654
  1655
  1656
  1657
  1658
  1659
  1660
  1661
  1662
  1663
  1664
  1665
  1666
  1667
  1668
  1669
  1670
  1671
  1672
  1673
  1674
  1675
  1676
  1677
  1678
  1679
  1680
  1681
  1682
  1683
  1684
  1685
  1686
  1687
  1688
  1689
  1690
  1691
  1692
  1693
  16
```

```

    1  package com.allyhyeseong.lifegame;
    2
    3  import org.springframework.stereotype.Component;
    4
    5  @Component
    6  public class HaltAction implements Action {
    7
    8      private Worker worker;
    9
   10     @Autowired
   11     public HaltAction(Worker worker) { this.worker = worker; }
   12
   13     @Override
   14     public void action() { this.worker.stopThread(); }
   15
   16 }

```

▲ [picture 52] HaltAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Worker Bean을 주입받았다.

action method를 실행하면 Worker의 현재 실행 중인 스레드를 종료하는 stopThread method를 호출한다.

4.4.3.7. [Controller] TickAction class

TickAction class는 Action interface를 구현한다.

```

    1  package com.allyhyeseong.lifegame;
    2
    3  import org.springframework.stereotype.Component;
    4
    5  @Component
    6  public class TickAction implements Action {
    7
    8      private final Model model;
    9
   10     @Autowired
   11     public TickAction(Model model) { this.model = model; }
   12
   13     @Override
   14     public void action() { this.model.nextState(); }
   15
   16 }

```

▲ [picture 53] TickAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Model Bean을 주입받았다.

action method를 실행하면 Model을 다음 상태로 업데이트하는 nextState method를 호출한다.

4.4.3.8. [Controller] AgonizingAction class

AgonizingAction class는 Action interface를 구현한다.

```

    4 usages ▲ AllyHyeseongKim
    @Component
    public class AgonizingAction implements Action {
        private Worker worker;
        public AgonizingAction(Worker worker) { this.worker = worker; }
        @Override
        public void action() {
            this.worker.setSpeed(500);
            this.worker.startThread();
        }
    }

```

▲ [picture 54] AgonizingAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Worker Bean을 주입 받았다.

action method를 실행하면 Worker의 스레드 업데이트 주기를 500으로 설정하고 스레드를 실행한다.

4.4.3.9. [Controller] SlowAction class

SlowAction class는 Action interface를 구현한다.

```

    4 usages ▲ AllyHyeseongKim
    @Component
    public class SlowAction implements Action {
        private Worker worker;
        public SlowAction(Worker worker) { this.worker = worker; }
        @Override
        public void action() {
            this.worker.setSpeed(150);
            this.worker.startThread();
        }
    }

```

▲ [picture 55] SlowAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Worker Bean을 주입 받았다.

action method를 실행하면 Worker의 스레드 업데이트 주기를 150으로 설정하고 스레드를 실행한다.

4.4.3.10. [Controller] MediumAction class

MediumAction class는 Action interface를 구현한다.

```

4 usages ▲ AllyHyeseongKim
@Component
public class MediumAction implements Action {

    3 usages
    private Worker worker;

    ▲ AllyHyeseongKim
    @Autowired
    public MediumAction(Worker worker) { this.worker = worker; }

    ▲ AllyHyeseongKim
    @Override
    public void action() {
        this.worker.setSpeed(70);
        this.worker.startThread();
    }
}

```

▲ [picture 56] MediumAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Worker Bean을 주입 받았다.

action method를 실행하면 Worker의 스레드 업데이트 주기를 70으로 설정하고 스레드를 실행한다.

4.4.3.11. [Controller] FastAction class

FastAction class는 Action interface를 구현 한다.

```

4 usages ▲ AllyHyeseongKim
@Component
public class FastAction implements Action {

    3 usages
    private Worker worker;

    ▲ AllyHyeseongKim
    @Autowired
    public FastAction(Worker worker) { this.worker = worker; }

    ▲ AllyHyeseongKim
    @Override
    public void action() {
        this.worker.setSpeed(30);
        this.worker.startThread();
    }
}

```

▲ [picture 57] FastAction class

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 구현하였고, 생성자에서 @Autowired를 이용해 Worker Bean을 주입 받았다.

action method를 실행하면 Worker의 스레드 업데이트 주기를 30으로 설정하고 스레드를 실행한다.

4.4.3.12. [Controller] DrawBehavior interface

LifePanel에서 발생한 MouseEvent를 LifeController가 처리할 때 Strategy Pattern을 활용하여 기본 그리기와 패턴 그리기 중 선택하여 처리할 수 있게 개발하였다. DrawBehavior interface는 Strategy들의 프로토 타입을 정의한 인터페이스이다.

```
1 package com.LifeGame.controller.drawBehavior;
2
3 public interface DrawBehavior {
4     void draw(int x, int y);
5 }
6
```

▲ [picture 58] DrawBehavior interface

draw method는 x, y(각각 Model의 row, column)을 입력 받아 draw 로직을 수행한다.

4.4.3.13. [Controller] DefaultDrawBehavior class

DefaultDrawBehavior class는 DrawBehavior interface를 구현한다.

```
1 package com.LifeGame.controller.drawBehavior;
2
3 import org.springframework.stereotype.Component;
4 import org.springframework.stereotype.Primary;
5
6 @Component
7 @Primary
8 public class DefaultDrawBehavior implements DrawBehavior {
9
10     private final Model model;
11
12     @Autowired
13     public DefaultDrawBehavior(Model model) { this.model = model; }
14
15     @Override
16     public void draw(int x, int y) { this.model.toggle(x, y); }
17
18 }
```

▲ [picture 59] DefaultDrawBehavior class

draw method를 실행하면 기존 동작하던 기능처럼 클릭한 위치의 칸을 toggle하는 기능을 수행한다.

4.4.3.14. [Controller] PatternDrawBehavior class

PatternDrawBehavior class는 DrawBehavior interface를 구현한다.

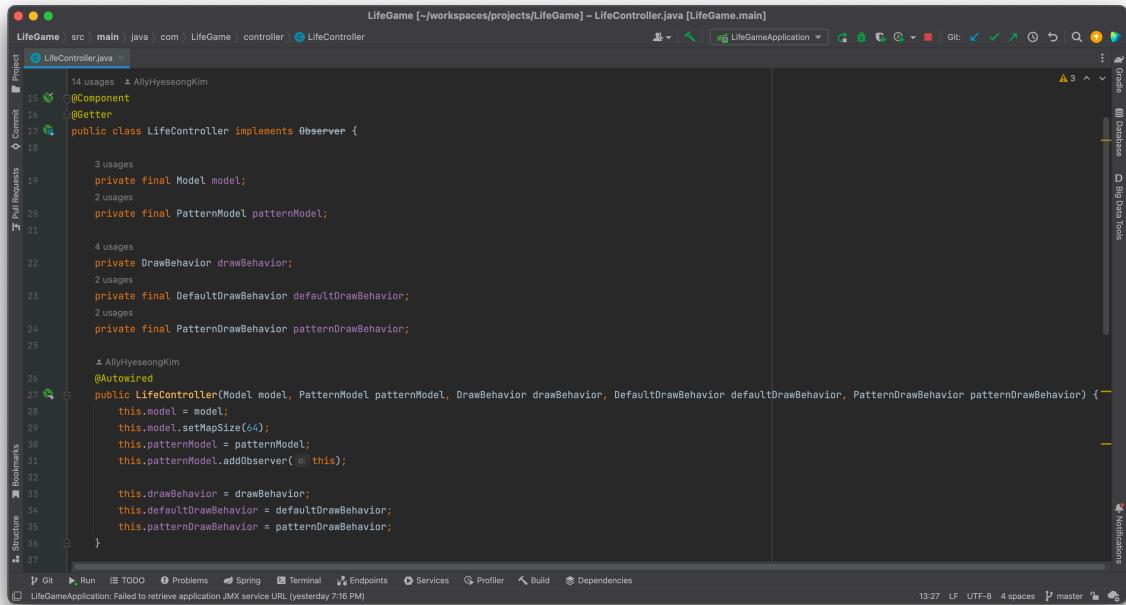
```
1 package com.LifeGame.controller.drawBehavior;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class PatternDrawBehavior implements DrawBehavior {
7
8     private final Model model;
9     private final PatternModel patternModel;
10
11     @Autowired
12     public PatternDrawBehavior(Model model, PatternModel patternModel) {
13         this.model = model;
14         this.patternModel = patternModel;
15     }
16
17     @Override
18     public void draw(int x, int y) { this.model.draw(this.patternModel.getBluePrint(), x, y); }
19
20 }
```

▲ [picture 60] PatternDrawBehavior class

draw method를 실행하면 Model의 x, y파라미터에 해당하는 위치에 patternModel에 등록된 pattern을 그리게 된다.

4.4.3.15. [Controller] LifeController class

LifeController class는 java.util.Observer를 구현한다. LifePanel에서 발생하는 이벤트를 핸들링하는 역할을 갖고 있으며, @Component Annotation을 이용해 Spring Bean으로 등록하고 Singleton으로 동작한다.



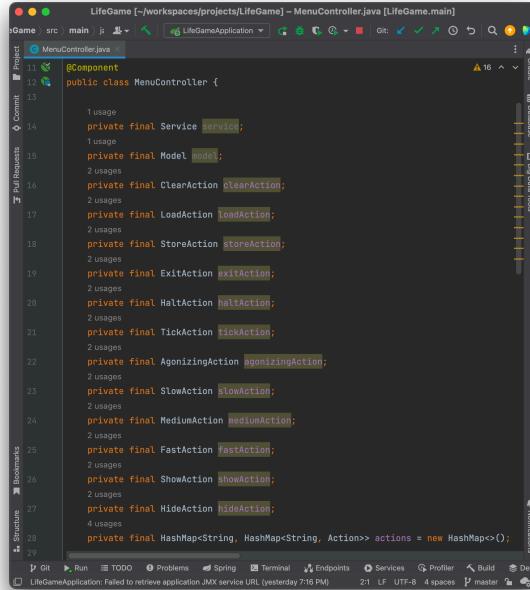
```
LifeGame [~/workspaces/projects/LifeGame] - LifeController.java [LifeGame.main]
Project  LifeController.java
14 usages  AllyHyeseongKim
15  @Component
16  @Getter
17  public class LifeController implements Observer {
18
19      3 usages
20      private final Model model;
21      2 usages
22      private final PatternModel patternModel;
23
24      4 usages
25      private DrawBehavior drawBehavior;
26      2 usages
27      private final DefaultDrawBehavior defaultDrawBehavior;
28      2 usages
29      private final PatternDrawBehavior patternDrawBehavior;
30
31      1 AllyHyeseongKim
32      @Autowired
33      public LifeController(Model model, PatternModel patternModel, DrawBehavior drawBehavior, DefaultDrawBehavior defaultDrawBehavior, PatternDrawBehavior patternDrawBehavior) {
34          this.model = model;
35          this.model.setMapSize(64);
36          this.patternModel = patternModel;
37          this.patternModel.addObserver(this);
38
39          this.drawBehavior = drawBehavior;
40          this.defaultDrawBehavior = defaultDrawBehavior;
41          this.patternDrawBehavior = patternDrawBehavior;
42
43      }
44
45      2 usages  AllyHyeseongKim
46      public void setDrawBehavior(DrawBehavior drawBehavior) { this.drawBehavior = drawBehavior; }
47
48      3 usages  AllyHyeseongKim
49      public DrawBehavior getDrawBehavior() { return this.drawBehavior; }
50
51      2 usages  AllyHyeseongKim
52      public void addObserver(Observer o) { this.model.addObserver(o); }
53
54      2 usages  AllyHyeseongKim
55      @Override
56      public void update( Observable o, Object arg) {
57          if (o instanceof PatternModel) {
58              if (((PatternModel) o).getId() == 0) {
59                  this.setDrawBehavior(this.defaultDrawBehavior);
60              } else {
61                  this.setDrawBehavior(this.patternDrawBehavior);
62              }
63          }
64      }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
784
785
786
786
787
787
788
788
789
789
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1
```

Strategy Pattern에서 Strategy를 설정하기 위한 setDrawBehavior method가 정의되어 있고, Model이 Observer를 추가하는 addObserver, mouseAction을 핸들링하여 적절한 Strategy를 실행하는 mouseAction method가 정의되어 있다.

PatternModel이 변할 때 마다 update method가 실행되고, 그 때마다 적절한 Strategy를 선택한다.

4.4.3.16. [Controller] MenuController class

MenuController는 Menu 관련 뷰에서 발생하는 모든 이벤트를 핸들링 한다.



```

11  * @Component
12  public class MenuController {
13
14      private final Service service;
15      private final Model model;
16      private final ClearAction clearAction;
17      private final LoadAction loadAction;
18      private final StoreAction storeAction;
19      private final ExitAction exitAction;
20      private final HaltAction haltAction;
21      private final TickAction tickAction;
22      private final AgonizingAction agonizingAction;
23      private final SlowAction slowAction;
24      private final MediumAction mediumAction;
25      private final FastAction fastAction;
26      private final ShowAction showAction;
27      private final HideAction hideAction;
28
29      private final HashMap<String, HashMap<String, Action>> actions = new HashMap<>();
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
969
970
971
972
973
974
975
975
976
977
977
978
979
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1044
1045
1046
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1054
1055
1056
1056
1057
1058
1058
1059
1060
1061
1062
1063
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1071
1072
1073
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1081
1082
1083
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1091
1092
1093
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1101
1102
1103
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887

```

생성자에서 `@Autowired` Annotation을 이용해 의존성을 주입받고, `HashMap`에 Menu와 Action을 등록한다.

```

57     HashMap<String, Action> goMenuItems = new HashMap<>();
58     goMenuItems.put("Halt", this.haltAction);
59     goMenuItems.put("Tick (Single Step)", this.tickAction);
60     goMenuItems.put("Agonizing", this.agonizingAction);
61     goMenuItems.put("Slow", this.slowAction);
62     goMenuItems.put("Medium", this.mediumAction);
63     goMenuItems.put("Fast", this.fastAction);
64     this.actions.put("Go", goMenuItems);
65
66     HashMap<String, Action> paletteMenuItems = new HashMap<>();
67     paletteMenuItems.put("Show", this.showAction);
68     paletteMenuItems.put("Hide", this.hideAction);
69     this.actions.put("Palette", paletteMenuItem);
70 }
71
72 /**
73 * @param AllyHyeseongKim
74 */
75 public void action(String menu, String menuItem) { this.actions.get(menu).get(menuItem).action(); }
76

```

▲ [picture 65] MenuController class (3)

Menu에서 발생하는 모든 액션을 핸들링하는 action method를 갖고 있고, 파라미터로 넘겨받은 menu, menuItem 이름을 `HashMap`에 조회하여 적절한 Action을 실행한다.

4.4.3.17. [Controller] PaletteController class

`PaletteController`는 `PatternPanel` 뷰에서 발생하는 모든 이벤트를 핸들링한다.

```

10 usages ▲ AllyHyeseongKim
11
12 @Component
13
14 public class PaletteController {
15
16     private final PatternModel patternModel;
17
18     @AllyHyeseongKim
19     @Autowired
20     public PaletteController(PatternModel patternModel) { this.patternModel = patternModel; }
21
22     public void addObserver(Observer o) { this.patternModel.addObserver(o); }
23
24     public void clickAction(int id, int[][] bluePrint) { this.patternModel.changePattern(id, bluePrint); }
25
26
27

```

▲ [picture 66] PaletteController class

`@Component` Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 동작한다.

생성자에서는 `@Autowired` Annotation을 이용해 `PaletteModel`의 존성을 주입받는다.

`addObserver` method를 이용해 view에서 `PaletteModel`에 옵저버로 등록할 수 있다.

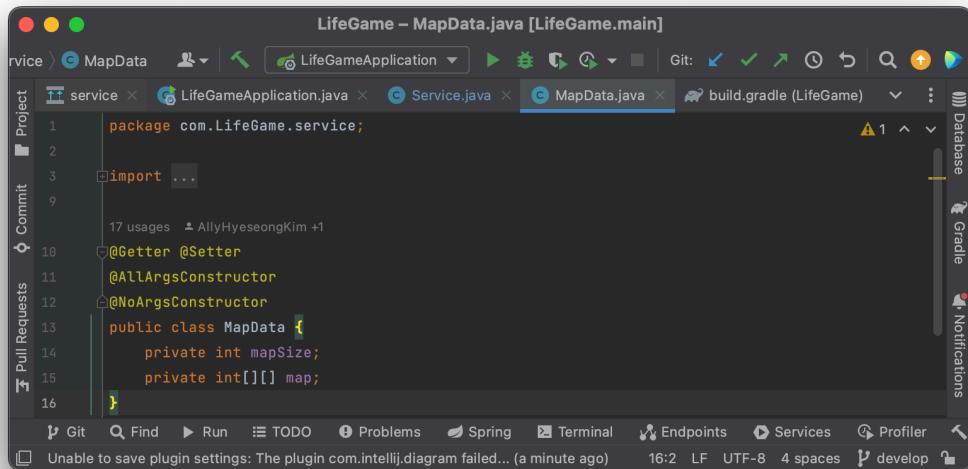
`clickAction` method는 `PatternPanel`에서 발생한 클릭 이벤트들을 핸들링하며, 파라미터로 전달받은 `bluePrint`를 모델에 적용한다.

4.5. Service

Controller가 view에서 넘어오는 매개변수를 이용하여 service 객체를 호출할 수 있다. Service는 view에 종속적인 코드가 없어 필요한 매개변수만 받으면 재사용이 가능하다. `MapData` 객체를 매개변수로 받아서 맵의 상태를 저장, 불러오기하는 기능을 service에서 담당한다.

4.5.1. [Service] MapData class

전달받는 매개변수는 MapData 객체로 int 형의 mapSize 와 실제 toggle 된 좌표의 2차원 배열을 담고 있는 map 변수를 갖고 있다.



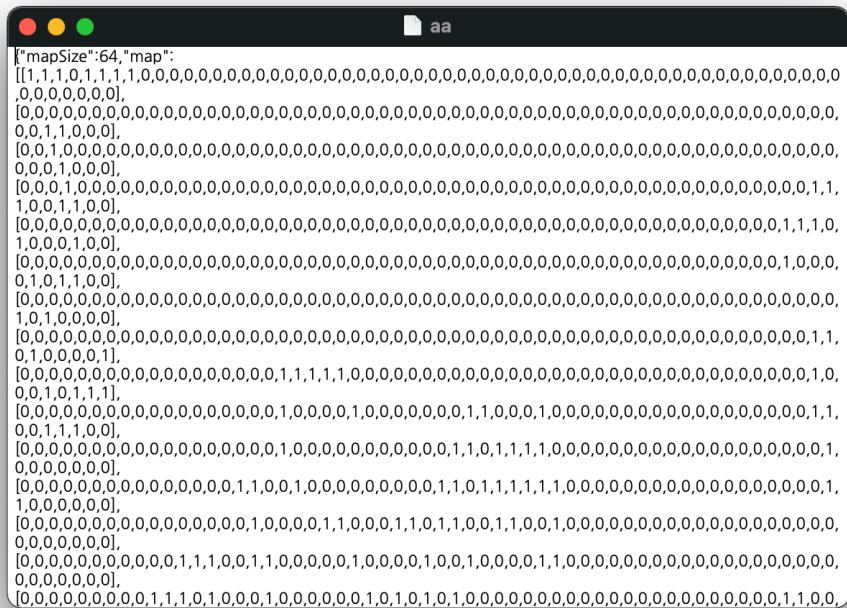
```
package com.LifeGame.service;
import ...;

@GETTER @SETTER
@ALLARGSCONSTRUCTOR
@NOARGSCONSTRUCTOR
public class MapData {
    private int mapSize;
    private int[][] map;
}
```

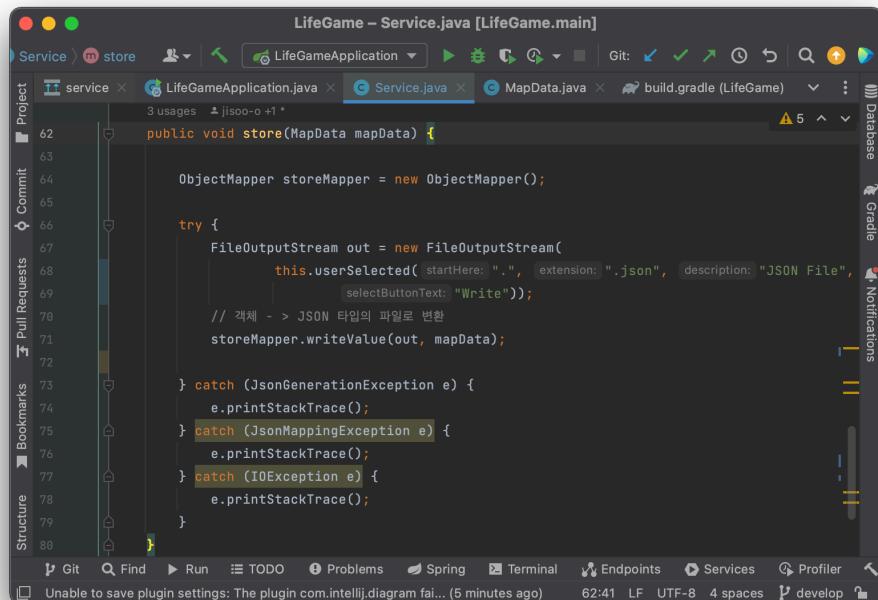
▲ [picture 67] MapData class

4.5.2. [Service] Service class

store, load 메소드를 구현하는 클래스로 store 는 MapData 객체를 전달 받아서 해당 객체를 Json 파일로 변환하여 저장한다. load 는 저장했던 Json 타입의 파일을 다시 MapData 객체로 변환하여 맵에 불러온다. 타입 변환은 ObjectMapper를 활용한다. 파일 선택 UI에서 user 가 선택한 파일, 경로를 userSelected 로 받아 해당 파일을 불러오거나 경로에 저장한다.



▲ [picture 68] 변환되어 저장된 결과물



▲ [picture 69] Service class (1)

정보를 store할 파일의 경로를 filechooser UI를 통해 userSelected로 받아 FileOutputStream로 지정 한다. ObjectMapper인 storeMapper의 writeValue 메소드를 이용하여 전달받은 mapData 객체를 Json 타입 파일로 변환하여 저장한다.

```
42
43     public MapData load() throws InvalidFileLoadedException {
44         ObjectMapper objectMapper = new ObjectMapper();
45
46         try {
47             // JSON 파일의 파일 -> 객체로 변환
48             MapData mapData = objectMapper.readValue(new FileInputStream(
49                 this.userSelected(startHere, extension, description, "JSON File",
50                     selectButtonText: "Load"), MapData.class);
51             return mapData;
52         } catch (JsonGenerationException e) {
53             e.printStackTrace();
54         } catch (JsonMappingException e) {
55             e.printStackTrace();
56         } catch (IOException e) {
57             e.printStackTrace();
58         }
59         throw new InvalidFileLoadedException();
60     }
```

▲ [picture 70] Service class (2)

load하고자 하는 파일의 경로를 같은 방법으로 받아 FileInputStream으로 지정하고 ObjectMapper의 readValue 메소드를 이용하여 다시 MapData 객체로 변환시켜서 int mapSize 와 int[] map 변수값을 받아온다.

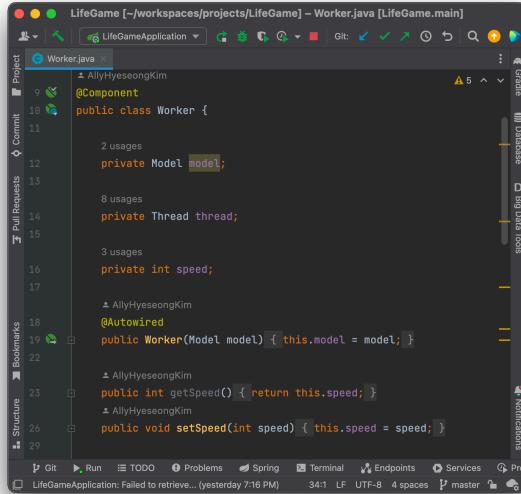
```
1 package com.LifeGame.service;
2
3 import ...
4
5 @Component
6 public class Service {
7
8     public File userSelected(final String startHere,
9             final String extension,
10            final String description,
11            final String selectButtonText)
12     throws FileNotFoundException {
13
14         FileFilter filter =
15             new FileFilter() {
16                 public boolean accept(File f) {
17                     return f.isDirectory()
18                         || (extension != null
19                             && f.getName().endsWith(extension));
20
21                 }
22
23                 public String getDescription() { return description; }
24             };
25
26
27         JFileChooser chooser = new JFileChooser(startHere);
28         chooser.setFileFilter(filter);
29
30         int result = chooser.showDialog( parent: null, selectButtonText);
31         if (result == JFileChooser.APPROVE_OPTION)
32             return chooser.getSelectedFile();
33
34         throw new FileNotFoundException("No file selected by user");
35     }
36 }
```

▲ [picture 71] Service class (3)

4.6. Worker

4.6.1. [Worker] Worker class

Worker class는 Model의 map을 설정한 주기로 업데이트하는 스레드를 관리하는 클래스이다.



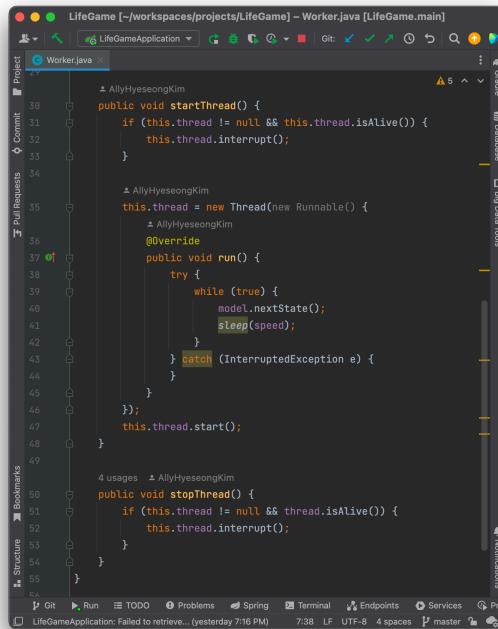
```
LifeGame [-/workspaces/projects/LifeGame] - Worker.java [LifeGame.main]
Worker.java x
  AllyHyeseongKim
    @Component
    public class Worker {
        2 usages
        private Model model;
        8 usages
        private Thread thread;
        3 usages
        private int speed;
        AllyHyeseongKim
        @Autowired
        public Worker(Model model) { this.model = model; }
        AllyHyeseongKim
        public int getSpeed() { return this.speed; }
        AllyHyeseongKim
        public void setSpeed(int speed) { this.speed = speed; }
    }

Project Commit Pull Requests Bookmarks Structure Git Run TODO Problems Spring Terminal Endpoints Services Profiler
LifeGameApplication: Failed to retrieve... (yesterday 7:16 PM) 34:1 LF UTF-8 4 spaces master
```

▲ [picture 72] Worker class (1)

@Component Annotation을 이용해 Spring Bean으로 등록해 Singleton으로 동작한다.

setSpeed method를 이용해 스레드가 모델을 업데이트하는 주기를 설정한다.



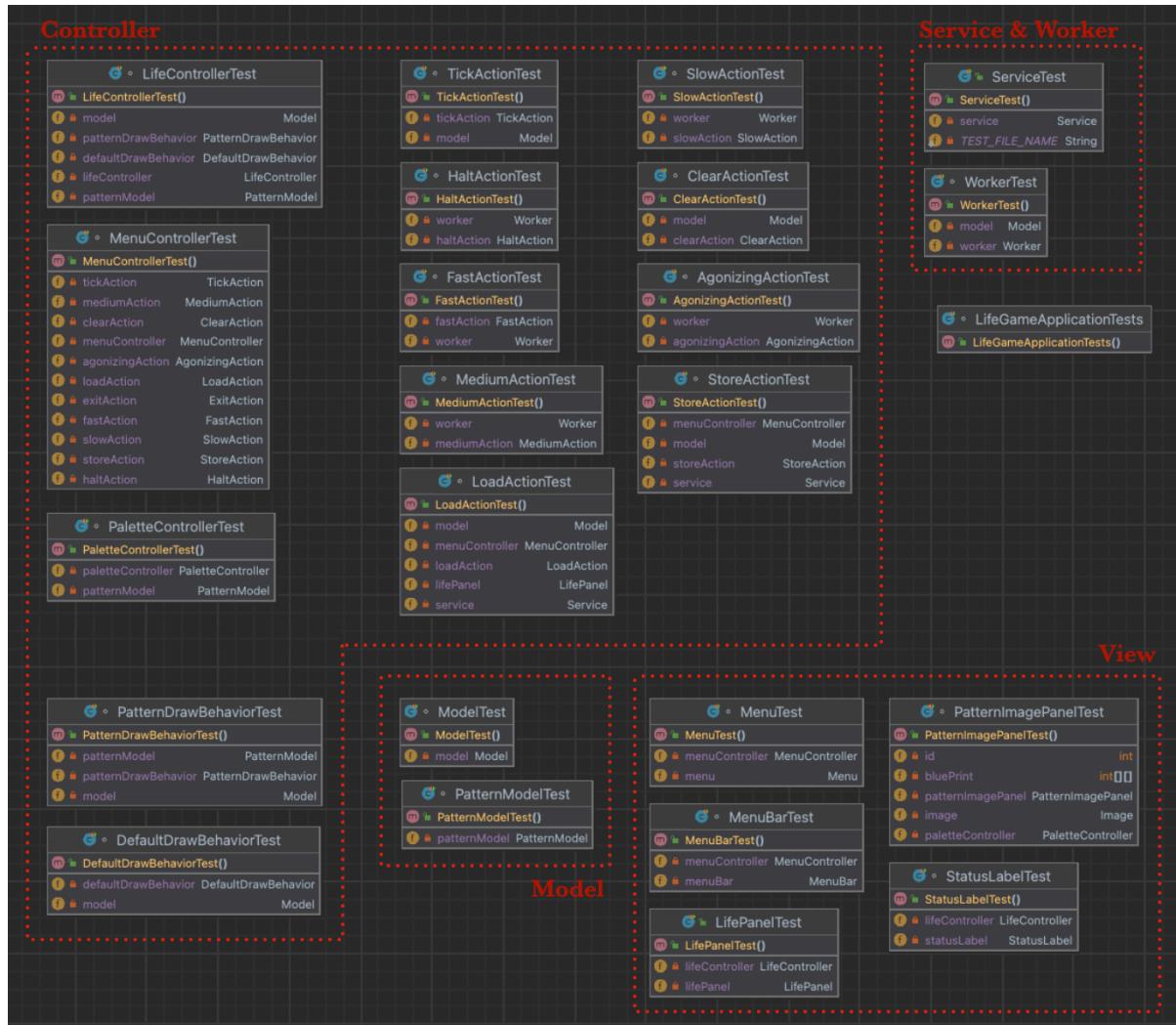
```
LifeGame [-/workspaces/projects/LifeGame] - Worker.java [LifeGame.main]
Worker.java x
  AllyHyeseongKim
    public void startThread() {
        if (this.thread != null && this.thread.isAlive()) {
            this.thread.interrupt();
        }
    }
    AllyHyeseongKim
    this.thread = new Thread(new Runnable() {
        AllyHyeseongKim
        @Override
        public void run() {
            try {
                while (true) {
                    model.nextState();
                    sleep(speed);
                }
            } catch (InterruptedException e) {
            }
        }
    });
    this.thread.start();
}
AllyHyeseongKim
public void stopThread() {
    if (this.thread != null && thread.isAlive()) {
        this.thread.interrupt();
    }
}
```

▲ [picture 73] Worker class (2)

startThread method를 이용해 설정한 주기마다 nextState를 호출하는 스레드를 동작시킨다. 스레드가 sleep하는 동안 클릭 이벤트에 의해 Model이 업데이트 되어야 하기 때문에 nextState method는 synchronized로 동작한다.

4.7. Test Cases

JUnit을 활용하여 모든 class에 대하여 테스트를 수행하였다.



▲ [picture 74] classes of test cases

4.7.1. [Test] ModelTest

Model class를 테스트하는 class이다. UI와는 분리하여 Model class에 대한 기능만을 테스트한다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – ModelTest.java [LifeGame.test]
- Project Tree:** model > ModelTest > draw1Test
- Code Editor:** ModelTest.java (highlighted)
- Imports:**

```
1 package com.LifeGame.model;
2
3 import com.LifeGame.view.LifePanel;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.extension.ExtendWith;
6 import org.junit.jupiter.api.DisplayName;
7 import org.junit.jupiter.api.Test;
8 import org.mockito.Spy;
9 import org.mockito.junit.jupiter.MockitoExtension;
10
11 import static org.junit.jupiter.api.Assertions.*;
12 import static org.mockito.ArgumentMatchers.any;
13 import static org.mockito.Mockito.mock;
14 import static org.mockito.Mockito.verify;
15 import static org.mockito.Mockito.times;
```
- Annotations:** `@ExtendWith(MockitoExtension.class)`, `@Spy`, `@BeforeEach`, `@Test`
- Structure View:** Shows `setUp()` method.
- Toolbars and Status Bar:** Git, Run, TODO, Problems, Spring, Terminal, Endpoints, Services, Profiler, Build, Dependencies. Status bar shows "Gradle sync finished in 20 s 322 ms (today 19:27)" and "144:1 LF UTF-8 4 spaces master".

▲ [picture 75] Test cases of Model class (1)

필요한 부분에만 stub하여 사용하기 위해 model을 mock 대신 `@Spy`로 선언한다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – ModelTest.java [LifeGame.test]
- Project Tree:** model > ModelTest > draw1Test
- Code Editor:** ModelTest.java (highlighted)
- Code:**

```
26
27     ▲ 박서현 +1
28     @Test
29     @DisplayName("[clearMap] clear map 호출시 mapChanged 호출되는지 test")
30     void clearMapTest() {
31         //given
32         this.model.setMapSize(3);
33         //when
34         this.model.clearMap();
35         //then
36         verify(this.model).mapChanged();
37     }
```
- Annotations:** `@Test`, `@DisplayName`, `verify`
- Toolbars and Status Bar:** Git, Run, TODO, Problems, Spring, Terminal, Endpoints, Services, Profiler, Build, Dependencies. Status bar shows "Gradle sync finished in 20 s 322 ms (today 19:27)" and "144:1 LF UTF-8 4 spaces master".

▲ [picture 76] Test cases of Model class (2)

`clearMapTest`로 `clear map`을 호출했을 때 `mapChanged`가 잘 호출되는지 `verify`를 활용하여 테스트한다.

```
java > com > LifeGame > model > ModelTest > draw1Te | LifeGameApplication > Model.java > ModelTest.java [LifeGame.test]
Project Pull Requests Bookmarks Structure
37
38     ↳ 박서현 +1
39     @Test
40     @DisplayName("[SetMapSize] 3*3 test")
41     void setMapSize1Test() {
42         //given
43         this.model.setMapSize(3);
44
45         //when
46         int[][] ans = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
47
48         //then
49         assertEquals(ans, this.model.getMap());
50     }
51
52     ↳ 박서현 +1
53     @Test
54     @DisplayName("[SetMapSize] 5*5 test")
55     void setMapSize2Test() {
56         //given
57         this.model.setMapSize(5);
58
59         //when
60         int[][] ans = {{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}};
61
62         //then
63         assertEquals(ans, this.model.getMap());
64     }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
928
929
929
930
931
932
933
934
935
936
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
958
958
959
959
960
961
962
963
964
965
966
967
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
202
```

```

java > com > LifeGame > model > ModelTest > draw1T
Project: LifeGameApplication.java > Model.java > ModelTest.java [LifeGame.test]
64
65     @Test
66     @DisplayName("[toggle] 3*3 test")
67     void toggle1Test() {
68
69         //given
70         this.model.setMapSize(3);
71
72         //when
73         this.model.toggle(x: 0, y: 0);
74         int[][] ans = {{1, 0, 0}, {0, 0, 0}, {0, 0, 0}};
75
76         //then
77         assertEquals(ans, this.model.getMap());
78     }
79
80     @Test
81     @DisplayName("[toggle] 1 -> 0 test")
82     void toggle2Test() {
83
84         //given
85         this.model.setMapSize(4);
86         this.model.toggle(x: 1, y: 3);
87
88         //when
89         this.model.toggle(x: 1, y: 3);
90         int[][] ans = {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};
91
92         //then
93         assertEquals(ans, this.model.getMap());
94     }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 144:1 LF UTF-8 4 spaces master

▲ [picture 78] Test cases of Model class (4)

toggle1Test로 3*3 크기의 map에서 좌표를 지정했을 때 값이 바뀌는지 확인하고, toggle2Test에서 4*4 크기의 map에서 값이 1인 인덱스가 0으로 바뀌는지 assertEquals로 확인한다.

```

java > com > LifeGame > model > ModelTest > draw1T
Project: LifeGameApplication.java > Model.java > ModelTest.java [LifeGame.test]
93
94     @Test
95     @DisplayName("[toggle] 5*5 test")
96     void toggle3Test() {
97
98         //given
99         this.model.setMapSize(5);
100
101        //when
102        this.model.toggle(x: 4, y: 2);
103        int[][] ans = {{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 1, 0, 0}};
104
105        //then
106        assertEquals(ans, this.model.getMap());
107    }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 144:1 LF UTF-8 4 spaces master

▲ [picture 79] Test cases of Model class (5)

toggle3Test로 5*5 크기의 map에서 지정한 좌표의 값이 바뀌는지 assertEquals로 확인한다.

```

java > com > LifeGame > model > ModelTest > draw1Test > Model.java > ModelTest.java
Project Commit Pull Requests Bookmarks Structure
108    ↳ 박서현 +1
109    @Test
110    @DisplayName("[toggle] toggle 호출시 mapChanged 호출되는지 test")
111    void toggleTest() {
112        //given
113        this.model.setMapSize(3);
114        //when
115        this.model.toggle(x: 1, y: 1);
116        //then
117        verify(this.model).mapChanged();
118    }
119    ↳ 박서현 +1
120    @Test
121    @DisplayName("[setMap] setMap 호출시 mapChanged 호출되는지 test")
122    void setMapTest() {
123        //given
124        int[][] arr = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
125        //when
126        this.model.setMap(arr);
127        //then
128        verify(this.model).mapChanged();
    }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 144:1 LF UTF-8 4 spaces master

▲ [picture 80] Test cases of Model class (6)

toggleTest와 setMapTest로 toggle이 호출되면 내부의 mapChanged가 호출되는지 verify를 통해 확인 한다.

```

java > com > LifeGame > model > ModelTest > draw1Test > Model.java > ModelTest.java
Project Commit Pull Requests Bookmarks Structure
130    ↳ AllyHyeseongKim
131    @Test
132    @DisplayName("[draw] draw 기능 테스트")
133    void draw1Test() {
134        //given
135        this.model.setMapSize(3);
136
137        int[][] bluePrint = new int[][] {{0, 1}, {1, 0}};
138        int x = 0;
139        int y = 0;
140
141        //when
142        this.model.draw(bluePrint, x, y);
143        //then
144        assertEquals(new int[][] {{0, 1, 0}, {1, 0, 0}, {0, 0, 0}}, this.model.getMap());
145        verify(this.model).mapChanged();
146    }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 128:6 LF UTF-8 4 spaces master

▲ [picture 81] Test cases of Model class (7)

draw1Test로 draw 기능이 잘 동작하는지 assertEquals로 확인하고, mapChanged가 호출되는지 verify로 확인한다.

```

com > LifeGame > model > ModelTest > setMapTest > ModelTest.java [LifeGame.test]
Project Commit Pull Requests Bookmarks Structure
  ^ AllyHyeoseongKim
  @Test
  @DisplayName("[draw] 그림이 그려져야 할 때 draw 기능 테스트")
  void draw2Test() {
    //given
    this.model.setMapSize(3);
    int[][] bluePrint = new int[][] {{0, 1, 1, 0}, {1, 0, 1, 1}};
    int x = 1;
    int y = 1;
    //when
    this.model.draw(bluePrint, x, y);
    //then
    assertEquals(new int[][] {{0, 0, 0}, {0, 0, 1}, {0, 1, 0}}, this.model.getMap());
    verify(this.model).mapChanged();
  }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 128:6 LF UTF-8 4 spaces master

▲ [picture 82] Test cases of Model class (8)

Model class 테스트하는 class이다.

draw2Test로 크기나 벽이 있어 패턴이 모두 들어가지 못할 때 draw 기능이 동작하는지 확인하기 위해 assertEquals로 비교하고 verify로 mapChanged가 호출되는지 테스트한다.

```

com > LifeGame > model > ModelTest > draw2Test > ModelTest.java [LifeGame.test]
Project Commit Pull Requests Bookmarks Structure
  @Test
  @DisplayName("[nextState] 3*3 test")
  void nextState1Test() {
    //given
    int[][] arr = {{1, 1, 1}, {1, 0, 0}, {0, 0, 1}};
    this.model.setMap(arr);
    int[][] ans = {{1, 1, 0}, {1, 0, 1}, {0, 0, 0}};
    //when
    this.model.nextState();
    //then
    assertEquals(ans, this.model.getMap());
  }
  ^ 박서현 +1 *
  @Test
  @DisplayName("[nextState] 4*4 test")
  void nextState2Test() {
    //given
    int[][] arr = {{0, 0, 0, 0}, {1, 0, 1, 0}, {0, 1, 1, 0}, {0, 1, 0, 0}};
    this.model.setMap(arr);
    int[][] ans = {{0, 0, 0, 0}, {0, 0, 1, 0}, {1, 0, 1, 0}, {0, 1, 1, 0}};
    //when
    this.model.nextState();
    //then
    assertEquals(ans, this.model.getMap());
  }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 164:6 LF UTF-8 4 spaces master

▲ [picture 83] Test cases of Model class (9)

`nextState1Test`와 `nextState2Test`로 현재 배열에서 다음 상태로 변한 값을 `assertArrayEquals`로 비교한다.

The screenshot shows a Java IDE interface with the following details:

- Project Bar:** com > LifeGame > model > ModelTest.java
- Editor:** The code for `ModelTest.java` is displayed. It contains two test methods: `nextState3Test()` and `nextStateTest()`.
- Code Content:**

```
194
195     @Test
196     @DisplayName("nextState : 5*5 test")
197     void nextState3Test() {
198         //given
199         int[][] arr = {{0, 0, 0, 0, 0}, {0, 1, 1, 1, 0}, {0, 1, 0, 1, 0}, {0, 1, 1, 1, 0}, {0, 0, 0, 0, 0}};
200         this.model.setMap(arr);
201         int[][] ans = {{0, 0, 1, 0, 0}, {0, 1, 0, 1, 0}, {1, 0, 0, 0, 1}, {0, 1, 0, 1, 0}, {0, 0, 1, 0, 0}};
202
203         //when
204         this.model.nextState();
205
206         //then
207         assertEquals(ans, this.model.getMap());
208     }
209
210     @Test
211     @DisplayName("[nextState] nextState 호출시 mapChanged 2번 호출되는지 test")
212     void nextStateTest() {
213         //given
214         int[][] arr = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
215         this.model.setMap(arr);
216         //when
217         this.model.nextState();
218         verify(this.model, times(wantedNumberOfInvocations: 2)).mapChanged();
219     }
}
```
- Toolbars and Status Bar:** Includes Git status, build progress, and terminal access.

▲ [picture 84] Test cases of Model class (10)

nextState3Test로 다음 상태의 값을 assertArrayEquals로 비교하고, nextStateTest에서 mapChanged가 잘 호출되는지 verify로 확인한다.

```

com > LifeGame > model > ModelTest > draw2Test > LifeGameApplication > Model.java > ModelTest.java
Project Commit Pull Requests Bookmarks Structure
221
222
223 @Test
224 @DisplayName("[Model] 통합 테스트")
225 void modelTest() {
226     this.model.setMapSize(3);
227     this.model.toggle(x: 0, y: 0);
228     this.model.toggle(x: 0, y: 1);
229     this.model.toggle(x: 0, y: 2);
230     this.model.toggle(x: 1, y: 0);
231     this.model.toggle(x: 2, y: 2);
232     int[][] arr = this.model.getMap();
233     int[][] ans = {{1, 1, 1}, {1, 0, 0}, {0, 0, 1}};
234     assertEquals(ans, arr);
235
236     this.model.nextState();
237     int[][] ans2 = {{1, 1, 0}, {1, 0, 1}, {0, 0, 0}};
238     int[][] arr2 = this.model.getMap();
239     assertEquals(ans2, arr2);
240
241     this.model.toggle(x: 2, y: 2);
242     int[][] arr3 = this.model.getMap();
243     this.model.nextState();
244     int[][] ans3 = {{1, 1, 0}, {1, 0, 1}, {0, 1, 0}};
245     assertEquals(ans3, arr3);
246 }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 164:6 LF UTF-8 4 spaces master

▲ [picture 85] Test cases of Model class (11)

modelTest로 setMapSize, toggle, getMap, nextState의 기능이 있어서 동작하는지 assertEquals로 확인한다.

```

com > LifeGame > model > ModelTest > draw2Test > LifeGameApplication > Model.java > ModelTest.java
Project Commit Pull Requests Bookmarks Structure
246
247
248 @Test
249 @DisplayName("[mapChanged] mapChanged 호출 시 LifePanel에 update가 잘 호출되는지 테스트")
250 void mapChangedTest() {
251
252     //given
253     LifePanel lifePanel = mock(LifePanel.class);
254
255     this.model.addObserver(lifePanel);
256
257     //when
258     this.model.mapChanged();
259
260     //then
261     verify(lifePanel).update(any(), any());
262 }

```

Gradle sync finished in 20 s 322 ms (today 19:27) 164:6 LF UTF-8 4 spaces master

▲ [picture 86] Test cases of Model class (12)

mapChangedTest로 mock 객체를 만들고 함수가 호출되었을 때 update가 호출되는지 verify로 확인한다.

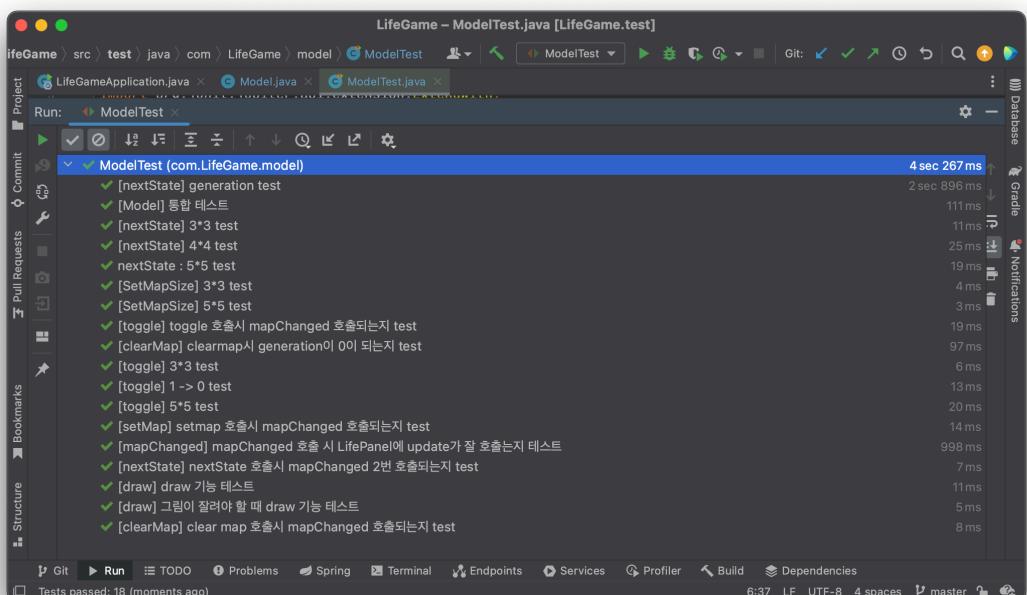
```

261     * 박서현
262     @Test
263     @DisplayName("[nextState] generation 테스트")
264     void nextState_gen() {
265         //given
266         int[][] arr = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
267         this.model.setMap(arr);
268         //when
269         this.model.nextState();
270         this.model.nextState();
271         //then
272         assertEquals(expected: 2, this.model.getGeneration());
273     }
274
275     * 박서현
276     @Test
277     @DisplayName("[clearMap] clearMap 시 generation이 0이 되는지 테스트")
278     void clearmap_gen() {
279         int[][] arr = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
280         this.model.setMap(arr);
281         //when
282         this.model.nextState();
283         this.model.nextState();
284         this.model.clearMap();
285         //then
286         assertEquals(expected: 0, this.model.getGeneration());
287     }

```

▲ [picture 87] Test cases of Model class (13)

nextstate_gen으로 nextState가 호출되었을 때 generation이 증가하는지, clearmap_gen으로 clearMap이 호출되었을 때 generation이 초기화 되는지 assertEquals로 비교한다.

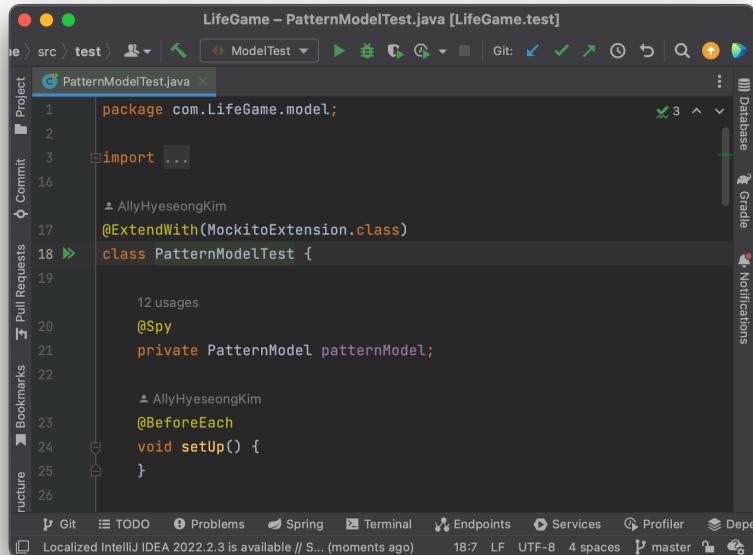


▲ [picture 88] ModelTest passed

모든 ModelTest가 통과한 것을 볼 수 있다.

4.7.2. [Test] PatternModelTest

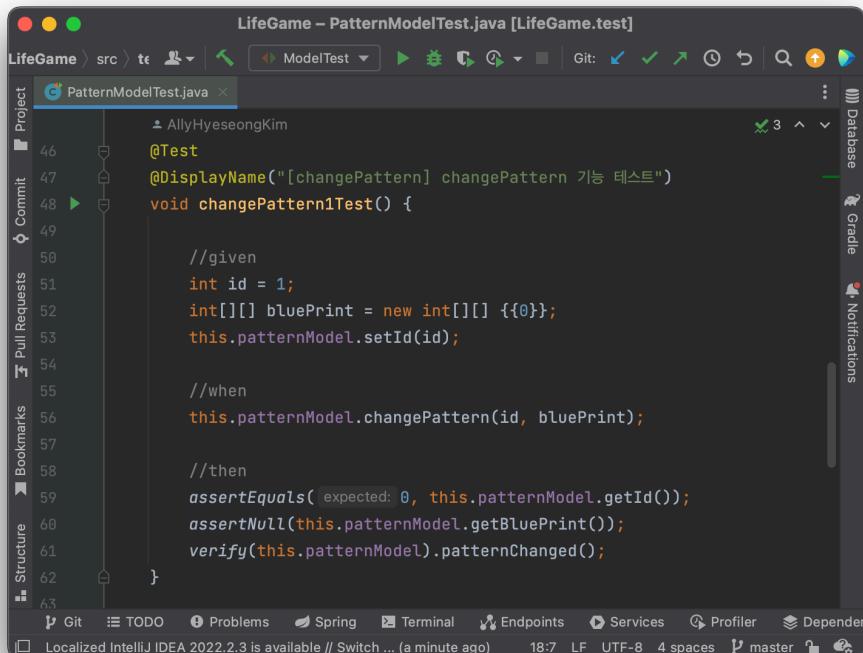
PatternModel class 테스트하는 class이다.



```
LifeGame - PatternModelTest.java [LifeGame.test]
1 package com.LifeGame.model;
2
3 import ...
4
5 @ExtendWith(MockitoExtension.class)
6 class PatternModelTest {
7
8     @Spy
9     private PatternModel patternModel;
10
11    @Test
12    void setUp() {
13
14    }
15}
```

▲ [picture 89] Test cases of PatternModel class (1)

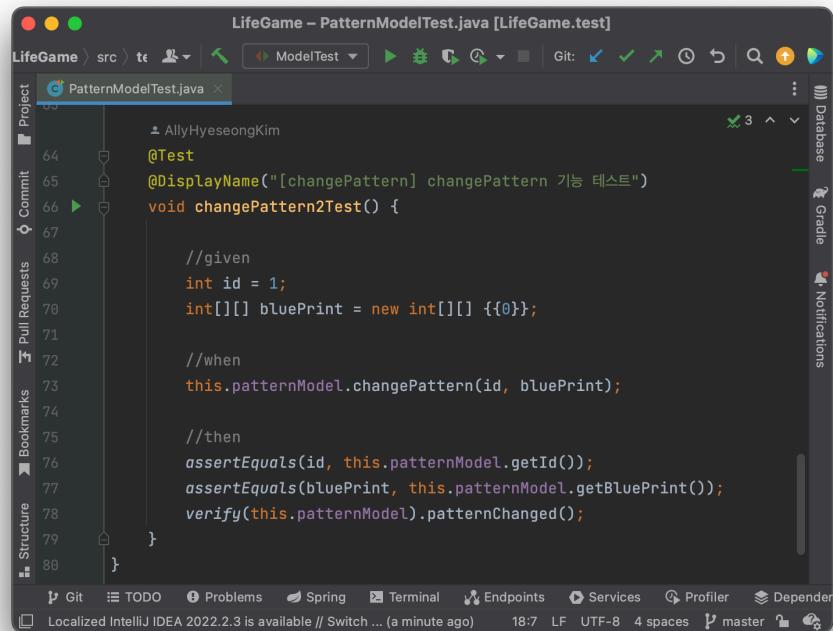
PatternModel의 @Spy Annotation을 이용해 verify와 when을 이용할 수 있도록 하였다.



```
LifeGame - PatternModelTest.java [LifeGame.test]
1 package com.LifeGame.model;
2
3 import org.junit.jupiter.api.DisplayName;
4 import org.junit.jupiter.api.Test;
5
6 import static org.junit.jupiter.api.Assertions.*;
7
8 class PatternModelTest {
9
10    @Test
11    @DisplayName("[changePattern] changePattern 기능 테스트")
12    void changePattern1Test() {
13
14        //given
15        int id = 1;
16        int[][] bluePrint = new int[][] {{0}};
17        this.patternModel.setId(id);
18
19        //when
20        this.patternModel.changePattern(id, bluePrint);
21
22        //then
23        assertEquals(0, this.patternModel.getId());
24        assertNull(this.patternModel.getBluePrint());
25        verify(this.patternModel).patternChanged();
26    }
27}
```

▲ [picture 90] Test cases of PatternModel class (2)

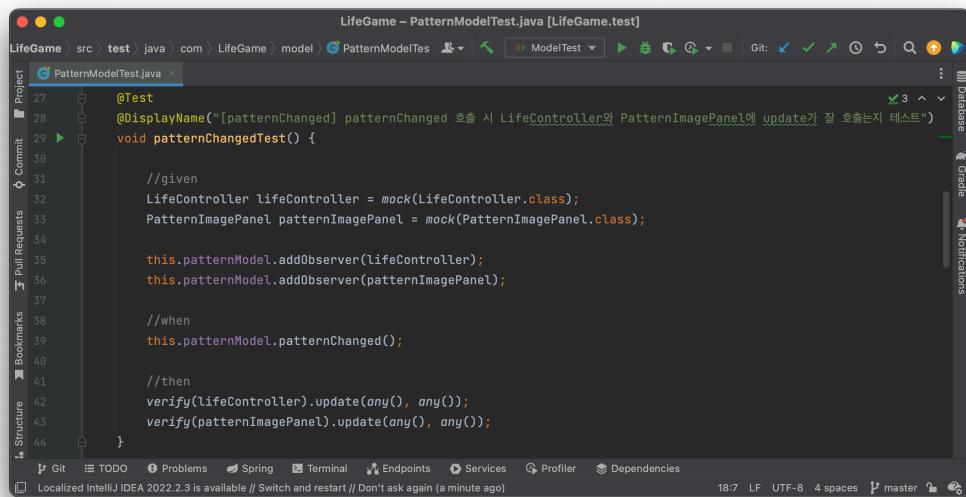
changePattern1Test는 changePattern method를 실행했을 때 이미 설정된 id로 패턴이 설정되는 경우 bluePrint를 Null로 설정하고 id를 0으로 잘 설정하는지 테스트한다.



```
LifeGame - PatternModelTest.java [LifeGame.test]
...
64     @Test
65     @DisplayName("[changePattern] changePattern 기능 테스트")
66     void changePattern2Test() {
67
68         //given
69         int id = 1;
70         int[][] bluePrint = new int[][] {{0}};
71
72         //when
73         this.patternModel.changePattern(id, bluePrint);
74
75         //then
76         assertEquals(id, this.patternModel.getId());
77         assertEquals(bluePrint, this.patternModel.getBluePrint());
78         verify(this.patternModel).patternChanged();
79     }
80 }
```

▲ [picture 91] Test cases of Pattern Model class (3)

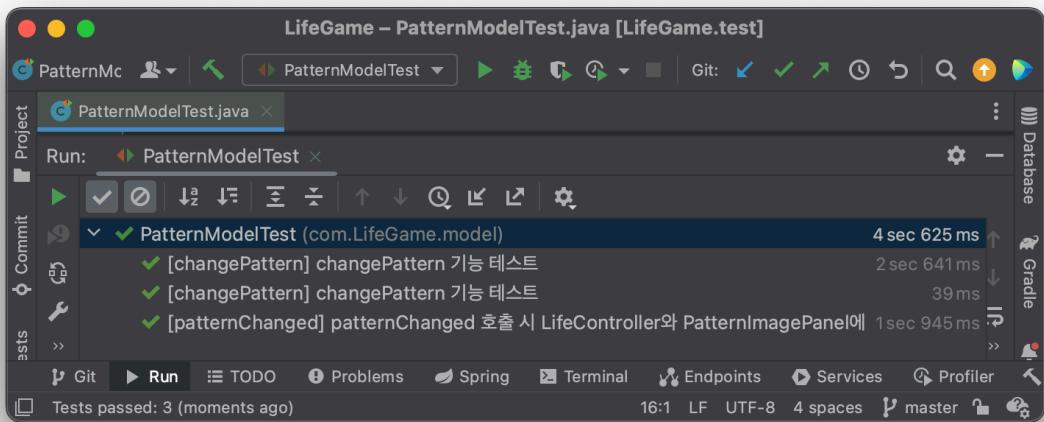
changePattern2Test는 기존 id와 다른 id를 설정할 때 설정한 id 및 bluePrint가 잘 설정 되는지, patternChanged가 잘 호출되는지 테스트한다.



```
LifeGame - PatternModelTest.java [LifeGame.test]
...
27     @Test
28     @DisplayName("[patternChanged] patternChanged 호출 시 LifeController와 PatternImagePanel에 update가 잘 호출되는지 테스트")
29     void patternChangedTest() {
30
31         //given
32         LifeController lifeController = mock(LifeController.class);
33         PatternImagePanel patternImagePanel = mock(PatternImagePanel.class);
34
35         this.patternModel.addObserver(lifeController);
36         this.patternModel.addObserver(patternImagePanel);
37
38         //when
39         this.patternModel.patternChanged();
40
41         //then
42         verify(lifeController).update(any(), any());
43         verify(patternImagePanel).update(any(), any());
44     }
45 }
```

▲ [picture 92] Test cases of PatternModel class (4)

patternChangedTest는 patternChanged를 호출하는 경우 Observer들의 Update가 잘 호출되는지 테스트한다.



▲ [picture 93] PatternModelTest Passed

Test가 모두 통과하였음을 확인 할 수 있다.

4.7.3. [Test] MenuTest

Menu class 테스트하는 class이다.

```

package com.LifeGame.view.menu;

import ...;

@ExtendWith(MockitoExtension.class)
class MenuTest {

    @Mock
    private MenuController menuController;
    11 usages
    @InjectMocks
    private Menu menu;

    @BeforeEach
    void setUp() {
    }
}

```

▲ [picture 94] Test cases of Menu class (1)

Menu를 독립적으로 테스트하기 위해 Mockito의 `@Mock`을 이용해 MenuController를 Mocking하고 Menu에 주입하였다.

```
LifeGame – MenuTest.java [LifeGame.test]
change PatternModelTest Git: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

@Project
@Commit
@Pull Requests
@Bookmarks
@Structure

@DisplayName("addMenuItem: JMenuItem 추가 테스트")
void addMenuItemTest() {
    //given
    String name = "test menu item";
    //when
    this.menu.addMenuItem(name);
    //then
    assertEquals(name, this.menu.getMenuItem(name).getText());
}

▲ AllyHyeseongKim
@Test
@DisplayName("removeMenuItem: JMenuItem 삭제 테스트")
void removeMenuItemTest() {
    //given
    String name = "test menu item";
    this.menu.addMenuItem(name);
    //when
    this.menu.removeMenuItem(name);
    //then
    assertNull(this.menu.getMenuItem(name));
}

Tests passed: 3 (4 minutes ago)
55:10 LF UTF-8 4 spaces master
```

▲ [picture 95] Test cases of Menu class (2)

addMenuItemTest는 addMenuItem을 호출했을 때 MenuItem이 잘 생성되었는지 Text를 비교하여 검증하는 테스트이다.

removeMenuItemTest는 removeItem을 호출했을 때 MenuItem을 잘 삭제하는지 테스트한다. getMenuItem을 호출했을 때 Null이 반환되는지 확인한다.

```

55     @Test
56     @DisplayName("changeName: JMenuItem 이름 변경 테스트")
57     void changeNameTest() {
58
59         //given
60         String name = "test menu item";
61         this.menu.addMenuItem(name);
62
63         //when
64         String newName = "changed name";
65         this.menu.changeName(name, newName);
66
67         //then
68         assertNull(this.menu.getMenuItem(name));
69         assertEquals(newName, this.menu.getMenuItem(newName).getText());
70     }
71
72     @Test
73     @DisplayName("getMenuItem: JMenuItem getter 테스트")
74     void getMenuItemTest() {
75
76         //given
77         String name = "test menu item";
78
79         //when
80         this.menu.addMenuItem(name);
81
82         //then
83         assertEquals(name, this.menu.getMenuItem(name).getText());
84     }

```

Tests passed: 3 (4 minutes ago) 55:10 LF UTF-8 4 spaces master

▲ [picture 96] Test cases of Menu class (3)

changeNameTest는 changeName을 호출했을 때 MenuItem의 이름이 잘 변하는지 테스트한다. changeName을 호출하고 기존 이름의 MenuItem을 get 했을 때 Null인지, 새로운 이름으로 get 했을 때 설정한 Text가 적용되었는지 검증한다.

Test Case	Time
addMenuItem: JMenuItem 추가 테스트	2 sec 509 ms
removeMenuItem: JMenuItem 삭제 테스트	15 ms
changeName: JMenuItem 이름 변경 테스트	15 ms
getMenuItem: JMenuItem getter 테스트	7 ms

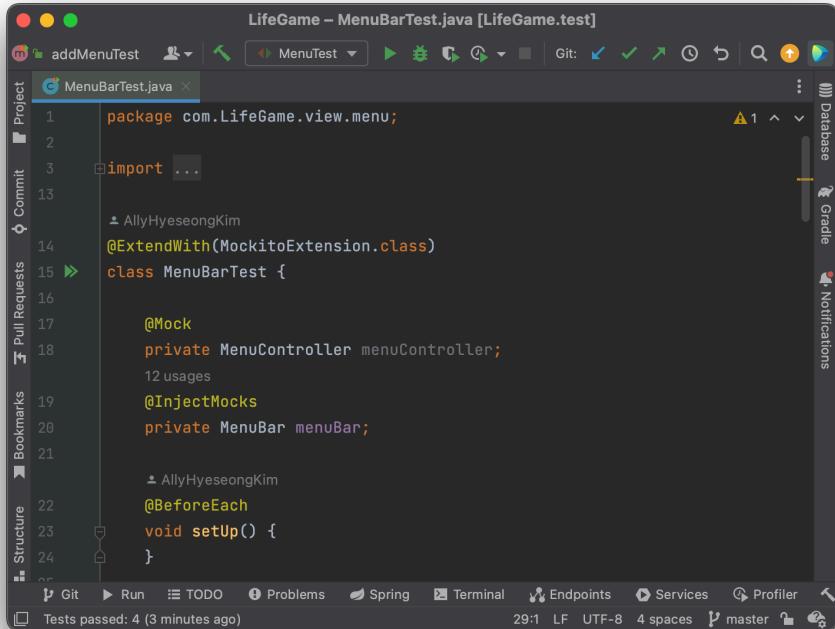
Tests passed: 4 (moments ago) 13:1 LF UTF-8 4 spaces master

▲ [picture 97] MenuTest Passed

모든 Test가 통과하였음을 알 수 있다.

4.7.4. [Test] MenuBarTest

MenuBar class 테스트하는 class이다.



```
package com.LifeGame.view.menu;

import ...  
import AllyHyeseongKim  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
  
@ExtendWith(MockitoExtension.class)  
public class MenuBarTest {  
  
    @Mock  
    private MenuController menuController;  
    @InjectMocks  
    private MenuBar menuBar;  
  
    @BeforeEach  
    void setUp() {  
    }  
}
```

▲ [picture 98] Test cases of MenuBar class (1)

MenuBar를 독립적으로 테스트하기 위해 Mockito @Mock을 이용해 MenuController를 Mocking하고 MenuBar에 주입하여 테스트 하였다.

The screenshot shows an IDE interface with the following details:

- Title Bar:** LifeGame – MenuBarTest.java [LifeGame.test]
- Toolbar:** Includes icons for BarTest, set up, Git, and various navigation and search functions.
- Left Sidebar:** Shows a project tree with 'MenuBarTest.java' selected, and sections for Commit, Pull Requests, Bookmarks, and Structure.
- Code Editor:** Displays the following Java code:

```
57     @Test
58     @DisplayName("changeName: JMenu 이름 변경 테스트")
59     void changeNameTest() {
60
61         //given
62         String name = "test menu";
63         this.menuBar.addMenu(name);
64
65         //when
66         String newName = "changed name";
67         this.menuBar.changeName(name, newName);
68
69         //then
70         assertNull(this.menuBar.getMenu(name));
71         assertEquals(newName, this.menuBar.getMenu(newName).getText());
72     }
73
74    盟 AllyHyeseongKim
75     @Test
76     @DisplayName("getMenu: JMenu getter 테스트")
77     void getMenuTest() {
78
79         //given
80         String name = "test menu";
81
82         //when
83         this.menuBar.addMenu(name);
84
85         //then
86         assertEquals(name, this.menuBar.getMenu(name).getText());
87     }

```

- Bottom Status Bar:** Shows 'Tests passed: 4 (3 minutes ago)', system information (24:6 LF UTF-8 4 spaces), and a master branch indicator.

▲ [picture 99] Test cases of MenuBar class (2)

changeNameTest는 changeName method를 실행하였을 때 기존 name의 메뉴가 null인지, 바뀐 이름의 메뉴가 존재하는지 테스트한다.

getMENutest는 getMenu method를 실행하였을 때 올바른 Menu가 반환되는지 테스트한다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – MenuBarTest.java [LifeGame.test]
- Toolbar:** Includes icons for file operations, Git status, and search.
- Project View:** Shows a single file named "MenuBarTest.java".
- Code Editor:** Displays two test methods:
 - addMenuTest()**:

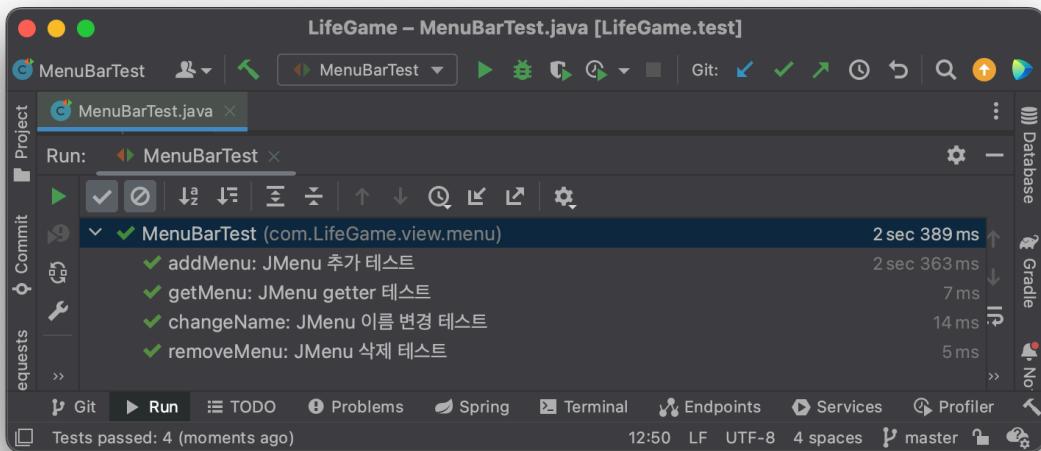
```
26     @Test
27     @DisplayName("addMenu: JMenu 추가 테스트")
28     void addMenuTest() {
29
30         //given
31         String name = "test menu";
32         String menuItem = "test menu item";
33
34         //when
35         this.menuBar.addMenu(name, menuItem);
36
37         //then
38         assertEquals(name, this.menuBar.getMenu(name).getText());
39         assertEquals(menuItem, this.menuBar.getMenu(name).getMenuItem(menuI
40     }
41
42     ▲ AllyHyeseongKim
43     @Test
44     @DisplayName("removeMenu: JMenu 삭제 테스트")
45     void removeMenuTest() {
46
47         //given
48         String name = "test menu";
49         this.menuBar.addMenu(name);
50
51         //when
52         this.menuBar.removeMenu(name);
53
54         //then
55         assertNull(this.menuBar.getMenu(name));
    }
```
 - removeMenuTest()**:

```
26     @Test
27     @DisplayName("removeMenu: JMenu 삭제 테스트")
28     void removeMenuTest() {
29
30         //given
31         String name = "test menu";
32         this.menuBar.addMenu(name);
33
34         //when
35         this.menuBar.removeMenu(name);
36
37         //then
38         assertNull(this.menuBar.getMenu(name));
    }
```
- Side Panels:** Includes "Pull Requests", "Bookmarks", and "Structure" panels.
- Bottom Status Bar:** Shows "Tests passed: 4 (3 minutes ago)", system information (24:6 LF UTF-8 4 spaces), and a terminal tab.

▲ [picture 100] Test cases of MenuBar class (3)

addMenuTest는 addMenu를 호출하였을 때 설정한 name과 menuItem 인스턴스가 설정되었는지 assertEquals를 이용해 검증한다.

removeMenuTest는 removeMenu를 호출하였을 때 name에 해당하는 menu가 삭제되어 get했을 때 null이 반환되는지 assertNull로 체크한다.



▲ [picture 101] MenuBarTest Passed

MenuBarTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.5. [Test] LifePanelTest

LifePanel class 테스트하는 class이다

```

package com.LifeGame.view;

import ...;

@ExtendWith(MockitoExtension.class)
public class LifePanelTest {

    @Mock
    private LifeController lifeController;
    4 usages

    @Spy
    @InjectMocks
    private LifePanel lifePanel;

    @BeforeEach
    void setUp() {
    }
}

```

▲ [picture 102] Test cases of LifePanel class (1)

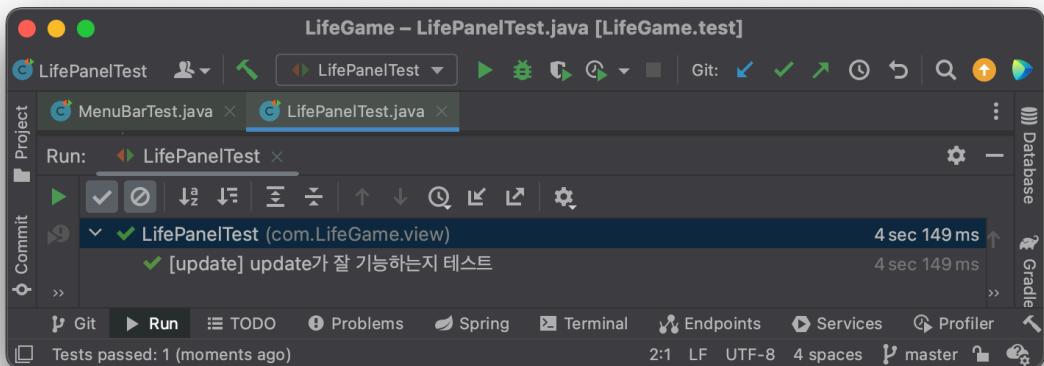
LifePanel을 독립적으로 테스트하기 위해 LifeController를 Mocking 하여 주입하였다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – LifePanelTest.java [LifeGame.test]
- Toolbar:** Includes icons for Test, setUp, MenuBarTest, Git, and various navigation and search tools.
- Project View:** Shows a tree structure with nodes like Project, Commit, Pull Requests, Bookmarks, and Structure.
- Code Editor:** Displays the `LifePanelTest.java` file content. The code is a JUnit test for the `update` method of the `LifePanel` class. It uses Mockito to mock the `Model` and `LifePanel` classes. The test sets up initial cell values, updates the model, and then asserts that all cells remain alive (value 1) and calls the `repaint` method on the panel.
- Bottom Status Bar:** Shows "Tests passed: 4 (3 minutes ago)", the current time (29:16), and the git status (master).

▲ [picture 103] Test cases of LifePanel class (2)

updateTest는 update가 호출되었을 때 올바른 Resident의 amAlive가 1로 설정 되었는지, repaint가 한번만 잘 실행되었는지 검증한다.



▲ [picture 104] LifePanelTest Passed

모든 LifePanelTest가 통과하였음을 확인할 수 있다.

4.7.6. [Test] PatternImagePanelTest

PatternImagePanel class 테스트하는 class이다.

```

package com.LifeGame.view;

import ...;

public class PatternImagePanelTest {
    private PaletteController paletteController;
    private PatternImagePanel patternImagePanel;
    private int id;
    private Image image;
    private int[][] bluePrint;

    @BeforeEach
    void setUp() throws IOException {
        this.paletteController = Mockito.mock(PaletteController.class);

        this.id = 1;
        this.image = Mockito.mock(Image.class);
        this.bluePrint = new int[][] {{0}};

        this.patternImagePanel = new PatternImagePanel(this.paletteController, this.id, this.image, this.bluePrint);
    }
}

```

▲ [picture 105] Test cases of PatternImagePanel class (1)

PaletteController, Image를 Mockito로 Mocking하고, 테스트에 필요한 PatternImagePanel을 @BeforeEach Annotation이 붙은 메소드에 적용하여, 각 테스트 이전에 초기화하도록 하였다. PatternImagePanel은 Spring Bean이 아니기 때문에 생성자로 생성하였다.

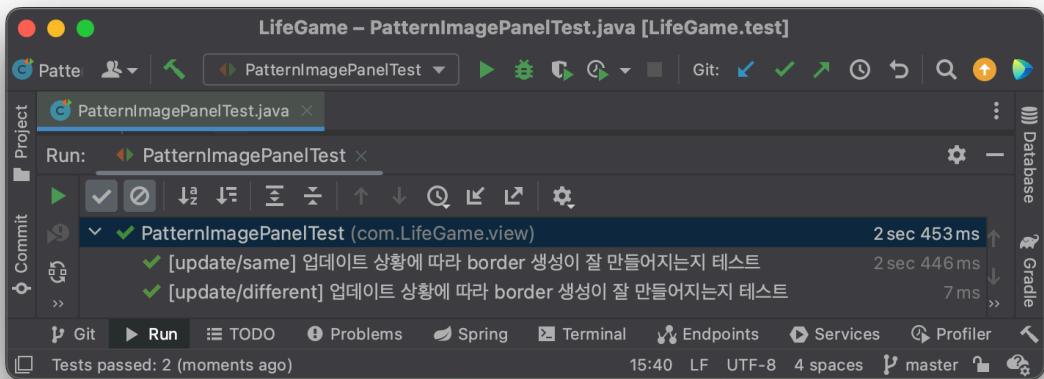
```
38     @Test
39     @DisplayName("[update/same] 업데이트 상황에 따라 border 생성이 잘 만들어지는지 테스트")
40     void updateSameTest() {
41
42         //given
43         PatternModel patternModel = Mockito.mock(PatternModel.class);
44         when(patternModel.getId()).thenReturn(this.id);
45
46         //when
47         this.patternImagePanel.update(patternModel, arg: null);
48
49         //then
50         assertTrue(this.patternImagePanel.getBorder() instanceof LineBorder);
51     }
52
53    盟 AllyHyeseongKim
54     @Test
55     @DisplayName("[update/different] 업데이트 상황에 따라 border 생성이 잘 만들어지는지 테스트")
56     void updateDifferentTest() {
57
58         //given
59         PatternModel patternModel = Mockito.mock(PatternModel.class);
60         when(patternModel.getId()).thenReturn( value: 2);
61
62         //when
63         this.patternImagePanel.update(patternModel, arg: null);
64
65         //then
66         assertNull(this.patternImagePanel.getBorder());
67     }

```

▲ [picture 106] Test cases of PatternImagePanel class (2)

updateSameTest는 update를 호출했을 때 PatternModel의 id값이 PatternImagePanel과 같은 경우, Border가 LineBorder로 잘 설정되는지 검증한다.

updateDifferentTest는 update를 호출했을 때 PatternModel의 id값이 PatternImagePanel과 다른 경우, Border가 Null로 잘 설정되는지 검증한다.



▲ [picture 107] PatternImagePanelTest Passed

PatternImagePanelTest가 모두 잘 통과하였음을 확인 할 수 있다.

4.7.7. [Test] StatusLabelTest

StatusLabel class 테스트하는 class이다.

```

1 package com.LifeGame.view;
2
3 import ...
4
5 // Mockito annotations
6 import org.junit.jupiter.api.extension.ExtendWith;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9
10 // Test class
11 @ExtendWith(MockitoExtension.class)
12 class StatusLabelTest {
13
14     @Mock
15     private LifeController lifeController;
16
17     @InjectMocks
18     private StatusLabel statusLabel;
19
20     // Setup
21     @BeforeEach
22     void setUp() {
23     }
24
25 }

```

The screenshot shows the IntelliJ IDEA interface with the title "LifeGame – StatusLabelTest.java [LifeGame.test]". The code editor displays the StatusLabelTest.java file. The code uses Mockito annotations: `@ExtendWith(MockitoExtension.class)` at the class level, `@Mock` and `@InjectMocks` at the field level, and `@BeforeEach` with a `setUp()` method. The status bar at the bottom right shows "Tests passed: 2 (2 minutes ago)" at "35:16".

▲ [picture 108] Test cases of StatusLabel class (1)

LifeController를 Mocking하여 주입하여 독립적으로 테스트하였다.

```

    31  @Test
    32  @DisplayName("[update] 업데이트 시 label이 잘 업데이트되는지 테스트")
    33  void updatePatternTest() {
    34
    35      //given
    36      int[][] map = {{0, 1, 0}, {0, 0, 0}, {1, 1, 1}};
    37      Model model = Mockito.mock(Model.class);
    38      when(model.getMapSize()).thenReturn( value: 3);
    39      when(model.getMap()).thenReturn(map);
    40
    41      //when
    42      thisStatusLabel.update(model, arg: null);
    43
    44      //then
    45      assertEquals( expected: 0, thisStatusLabel.getGeneration());
    46      assertEquals( expected: 4, thisStatusLabel.getLiveCells());
    47  }

```

▲ [picture 109] Test cases of StatusLabel class (2)

updatePatternTest는 update가 호출되었을 때 generation과 liveCell이 잘 설정되었는지 검증한다.

Run Configuration	Test Case	Duration
StatusLabelTest	[update] 업데이트 시 label이 잘 업데이트되는지 테스트	2 sec 999 ms

▲ [picture 110] StatusLabelTest Passed

모든 StatusLabelTest가 통과되었음을 확인할 수 있다.

4.7.8. [Test] LifeControllerTest

LifeController class 테스트하는 class이다.

```
1 package com.LifeGame.controller;
2
3 import ...
4
5 ▲ AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class LifeControllerTest {
8
9     @Mock
10    private Model model;
11    4 usages
12    @Mock
13    private PatternModel patternModel;
14
15    1 usage
16    @Mock
17    private DefaultDrawBehavior defaultDrawBehavior;
18    1 usage
19    @Mock
20    private PatternDrawBehavior patternDrawBehavior;
21
22    6 usages
23    @InjectMocks
24    private LifeController lifeController;
25
26    ▲ AllyHyeseongKim
27    @BeforeEach
28    void setUp() {
29    }
30
31
32
33
34
35
36
37 }
```

▲ [picture 111] test cases of LifeController class (1)

Model, PatternModel, DefaultDrawBehavior, PatternDrawBehavior를 Mocking하여 LifeController에 주입하였다.

```
LifeGame – LifeControllerTest.java [LifeGame.test]
@Project
@Commit
@Pull Requests
@Bookmarks
@LifeControllerTest.java
@AllyHyeseongKim
@Test
@DisplayName("[mouseAction] draw 메서드가 잘 호출되는지 테스트")
void mouseActionTest() {
    //given
    int x = 0;
    int y = 0;

    //when
    this.lifeController.mouseAction(x, y);

    //then
    verify(this.lifeController.getDrawBehavior()).draw(x, y);
}
```

Tests passed: 1 (6 minutes ago) 30:6 LF UTF-8 4 spaces ⚡ develop

▲ [picture 112] test cases of LifeController class (2)

mouseActionTest는 mouseAction이 호출되었을 때 drawBehavior을 이용해 draw가 호출되었는지 검증한다.

```
LifeGame – LifeControllerTest.java [LifeGame.test]
@Project
@Commit
@Pull Requests
@Bookmarks
@LifeControllerTest.java
@AllyHyeseongKim
@Test
@DisplayName("[update/default] 업데이트 상황에 따라 setDrawBehavior 메서드가 잘 호출되는지 테스트")
void updateDefaultTest() {
    //given
    when(this.patternModel.getId()).thenReturn(0);

    //when
    this.lifeController.update(this.patternModel, null);

    //then
    assertEquals(this.defaultDrawBehavior, this.lifeController.getDrawBehavior());
}
```

Tests passed: 1 (8 minutes ago) 83:17 LF UTF-8 4 spaces ⚡ develop

▲ [picture 113] test cases of LifeController class (3)

updateDefaultTest는 PatternModel의 id가 0일 때 drawBehavior이 0으로 잘 업데이트 되었는지 확인하는 테스트이다.

```

    70  @Test
    71  @DisplayName("[update/pattern] 업데이트 상황에 따라 setDrawBehavior 메서드가 " +
    72      "잘 호출되는지 테스트")
    73  void updatePatternTest() {
    74
    75      //given
    76      when(this.patternModel.getId()).thenReturn( value: 1);
    77
    78      //when
    79      this.lifeController.update(this.patternModel, arg: null);
    80
    81      //then
    82      assertEquals(this.patternDrawBehavior, this.lifeController
    83          .getDrawBehavior());
    84  }

```

Tests passed: 1 (8 minutes ago)

▲ [picture 114] test cases of LifeController class (4)

updatePatternTest는 lifeController의 update가 PatternModel에 의해 호출되었을 때 id가 0이 아닌 경우 drawBehavior Strategy가 patternDrawBehavior로 잘 설정되는지 검증한다.

Test Case	Test Description	Time
LifeControllerTest	[mouseAction] draw 메서드가 잘 호출되는지 테스트	66 ms
LifeControllerTest	[update/default] 업데이트 상황에 따라 setDrawBehavior 메서드가 잘 호출되는지 테스트	1sec 742 ms
LifeControllerTest	[update/pattern] 업데이트 상황에 따라 setDrawBehavior 메서드가 잘 호출되는지 테스트	7 ms

Tests passed: 3 (moments ago)

▲ [picture 115] LifeControllerTest passed

LifeControllerTest 모두가 통과되었음을 확인할 수 있다.

4.7.9. [Test] MenuControllerTest

MenuController class 테스트하는 class이다.

```
LifeGame – MenuControllerTest.java [LifeGame.test]
load | user | PalettesControllerTest | Run | Git: ⌛ ✓ 🔍 ⚡ ⚡ ⚡ ⚡ ⚡ ⚡ Database | ... | Notifications
Project | Pull Requests | Commit | Pull Requests | Bookmarks | Project Structure | Commit | ...
1 package com.LifeGame.controller;
2
3 import ...
4
5 * AllyHyeseongKim
6
7 @ExtendWith(MockitoExtension.class)
8 class MenuControllerTest {
9
10     1 usage
11     @Mock
12     private ClearAction clearAction;
13
14     1 usage
15     @Mock
16     private LoadAction loadAction;
17
18     1 usage
19     @Mock
20     private StoreAction storeAction;
21
22     1 usage
23     @Mock
24     private ExitAction exitAction;
25
26     1 usage
27     @Mock
28     private HaltAction haltAction;
29
30     1 usage
31     @Mock
32     private TickAction tickAction;
33
34     1 usage
35     @Mock
36     private AgonizingAction agonizingAction;
37
38     1 usage
39     @Mock
40     private SlowAction slowAction;
41
42     1 usage
43     @Mock
44     private MediumAction mediumAction;
45
46     2 usages
47     @InjectMocks
48     private MenuController menuController;
49
50 * AllyHyeseongKim
51
52 @BeforeEach
53 void setUp() {
54 }
55
56 * AllyHyeseongKim
57
58 @Test
59 void test() {
60     // Arrange
61     // Act
62     // Assert
63 }
64
65 * AllyHyeseongKim
66
67 @Test
68 void test() {
69     // Arrange
70     // Act
71     // Assert
72 }
73
74 * AllyHyeseongKim
75
76 @Test
77 void test() {
78     // Arrange
79     // Act
80     // Assert
81 }
82
83 * AllyHyeseongKim
84
85 @Test
86 void test() {
87     // Arrange
88     // Act
89     // Assert
90 }
91
92 * AllyHyeseongKim
93
94 @Test
95 void test() {
96     // Arrange
97     // Act
98     // Assert
99 }
100
101 * AllyHyeseongKim
102
103 @Test
104 void test() {
105     // Arrange
106     // Act
107     // Assert
108 }
109
110 * AllyHyeseongKim
111
112 @Test
113 void test() {
114     // Arrange
115     // Act
116     // Assert
117 }
118
119 * AllyHyeseongKim
120
121 @Test
122 void test() {
123     // Arrange
124     // Act
125     // Assert
126 }
127
128 * AllyHyeseongKim
129
130 @Test
131 void test() {
132     // Arrange
133     // Act
134     // Assert
135 }
136
137 * AllyHyeseongKim
138
139 @Test
140 void test() {
141     // Arrange
142     // Act
143     // Assert
144 }
145
146 * AllyHyeseongKim
147
148 @Test
149 void test() {
150     // Arrange
151     // Act
152     // Assert
153 }
154
155 * AllyHyeseongKim
156
157 @Test
158 void test() {
159     // Arrange
160     // Act
161     // Assert
162 }
163
164 * AllyHyeseongKim
165
166 @Test
167 void test() {
168     // Arrange
169     // Act
170     // Assert
171 }
172
173 * AllyHyeseongKim
174
175 @Test
176 void test() {
177     // Arrange
178     // Act
179     // Assert
180 }
181
182 * AllyHyeseongKim
183
184 @Test
185 void test() {
186     // Arrange
187     // Act
188     // Assert
189 }
190
191 * AllyHyeseongKim
192
193 @Test
194 void test() {
195     // Arrange
196     // Act
197     // Assert
198 }
199
200 * AllyHyeseongKim
201
202 @Test
203 void test() {
204     // Arrange
205     // Act
206     // Assert
207 }
208
209 * AllyHyeseongKim
210
211 @Test
212 void test() {
213     // Arrange
214     // Act
215     // Assert
216 }
217
218 * AllyHyeseongKim
219
220 @Test
221 void test() {
222     // Arrange
223     // Act
224     // Assert
225 }
226
227 * AllyHyeseongKim
228
229 @Test
230 void test() {
231     // Arrange
232     // Act
233     // Assert
234 }
235
236 * AllyHyeseongKim
237
238 @Test
239 void test() {
240     // Arrange
241     // Act
242     // Assert
243 }
244
245 * AllyHyeseongKim
246
247 @Test
248 void test() {
249     // Arrange
250     // Act
251     // Assert
252 }
253
254 * AllyHyeseongKim
255
256 @Test
257 void test() {
258     // Arrange
259     // Act
260     // Assert
261 }
262
263 * AllyHyeseongKim
264
265 @Test
266 void test() {
267     // Arrange
268     // Act
269     // Assert
270 }
271
272 * AllyHyeseongKim
273
274 @Test
275 void test() {
276     // Arrange
277     // Act
278     // Assert
279 }
280
281 * AllyHyeseongKim
282
283 @Test
284 void test() {
285     // Arrange
286     // Act
287     // Assert
288 }
289
290 * AllyHyeseongKim
291
292 @Test
293 void test() {
294     // Arrange
295     // Act
296     // Assert
297 }
298
299 * AllyHyeseongKim
300
301 @Test
302 void test() {
303     // Arrange
304     // Act
305     // Assert
306 }
307
308 * AllyHyeseongKim
309
310 @Test
311 void test() {
312     // Arrange
313     // Act
314     // Assert
315 }
316
317 * AllyHyeseongKim
318
319 @Test
320 void test() {
321     // Arrange
322     // Act
323     // Assert
324 }
325
326 * AllyHyeseongKim
327
328 @Test
329 void test() {
330     // Arrange
331     // Act
332     // Assert
333 }
334
335 * AllyHyeseongKim
336
337 @Test
338 void test() {
339     // Arrange
340     // Act
341     // Assert
342 }
343
344 * AllyHyeseongKim
345
346 @Test
347 void test() {
348     // Arrange
349     // Act
350     // Assert
351 }
352
353 * AllyHyeseongKim
354
355 @Test
356 void test() {
357     // Arrange
358     // Act
359     // Assert
360 }
361
362 * AllyHyeseongKim
363
364 @Test
365 void test() {
366     // Arrange
367     // Act
368     // Assert
369 }
370
371 * AllyHyeseongKim
372
373 @Test
374 void test() {
375     // Arrange
376     // Act
377     // Assert
378 }
379
380 * AllyHyeseongKim
381
382 @Test
383 void test() {
384     // Arrange
385     // Act
386     // Assert
387 }
388
389 * AllyHyeseongKim
390
391 @Test
392 void test() {
393     // Arrange
394     // Act
395     // Assert
396 }
397
398 * AllyHyeseongKim
399
400 @Test
401 void test() {
402     // Arrange
403     // Act
404     // Assert
405 }
406
407 * AllyHyeseongKim
408
409 @Test
410 void test() {
411     // Arrange
412     // Act
413     // Assert
414 }
415
416 * AllyHyeseongKim
417
418 @Test
419 void test() {
420     // Arrange
421     // Act
422     // Assert
423 }
424
425 * AllyHyeseongKim
426
427 @Test
428 void test() {
429     // Arrange
430     // Act
431     // Assert
432 }
433
434 * AllyHyeseongKim
435
436 @Test
437 void test() {
438     // Arrange
439     // Act
440     // Assert
441 }
442
443 * AllyHyeseongKim
444
445 @Test
446 void test() {
447     // Arrange
448     // Act
449     // Assert
450 }
451
452 * AllyHyeseongKim
453
454 @Test
455 void test() {
456     // Arrange
457     // Act
458     // Assert
459 }
460
461 * AllyHyeseongKim
462
463 @Test
464 void test() {
465     // Arrange
466     // Act
467     // Assert
468 }
469
470 * AllyHyeseongKim
471
472 @Test
473 void test() {
474     // Arrange
475     // Act
476     // Assert
477 }
478
479 * AllyHyeseongKim
480
481 @Test
482 void test() {
483     // Arrange
484     // Act
485     // Assert
486 }
487
488 * AllyHyeseongKim
489
490 @Test
491 void test() {
492     // Arrange
493     // Act
494     // Assert
495 }
496
497 * AllyHyeseongKim
498
499 @Test
500 void test() {
501     // Arrange
502     // Act
503     // Assert
504 }
505
506 * AllyHyeseongKim
507
508 @Test
509 void test() {
510     // Arrange
511     // Act
512     // Assert
513 }
514
515 * AllyHyeseongKim
516
517 @Test
518 void test() {
519     // Arrange
520     // Act
521     // Assert
522 }
523
524 * AllyHyeseongKim
525
526 @Test
527 void test() {
528     // Arrange
529     // Act
530     // Assert
531 }
532
533 * AllyHyeseongKim
534
535 @Test
536 void test() {
537     // Arrange
538     // Act
539     // Assert
540 }
541
542 * AllyHyeseongKim
543
544 @Test
545 void test() {
546     // Arrange
547     // Act
548     // Assert
549 }
550
551 * AllyHyeseongKim
552
553 @Test
554 void test() {
555     // Arrange
556     // Act
557     // Assert
558 }
559
560 * AllyHyeseongKim
561
562 @Test
563 void test() {
564     // Arrange
565     // Act
566     // Assert
567 }
568
569 * AllyHyeseongKim
570
571 @Test
572 void test() {
573     // Arrange
574     // Act
575     // Assert
576 }
577
578 * AllyHyeseongKim
579
580 @Test
581 void test() {
582     // Arrange
583     // Act
584     // Assert
585 }
586
587 * AllyHyeseongKim
588
589 @Test
590 void test() {
591     // Arrange
592     // Act
593     // Assert
594 }
595
596 * AllyHyeseongKim
597
598 @Test
599 void test() {
600     // Arrange
601     // Act
602     // Assert
603 }
604
605 * AllyHyeseongKim
606
607 @Test
608 void test() {
609     // Arrange
610     // Act
611     // Assert
612 }
613
614 * AllyHyeseongKim
615
616 @Test
617 void test() {
618     // Arrange
619     // Act
620     // Assert
621 }
622
623 * AllyHyeseongKim
624
625 @Test
626 void test() {
627     // Arrange
628     // Act
629     // Assert
630 }
631
632 * AllyHyeseongKim
633
634 @Test
635 void test() {
636     // Arrange
637     // Act
638     // Assert
639 }
640
641 * AllyHyeseongKim
642
643 @Test
644 void test() {
645     // Arrange
646     // Act
647     // Assert
648 }
649
650 * AllyHyeseongKim
651
652 @Test
653 void test() {
654     // Arrange
655     // Act
656     // Assert
657 }
658
659 * AllyHyeseongKim
660
661 @Test
662 void test() {
663     // Arrange
664     // Act
665     // Assert
666 }
667
668 * AllyHyeseongKim
669
670 @Test
671 void test() {
672     // Arrange
673     // Act
674     // Assert
675 }
676
677 * AllyHyeseongKim
678
679 @Test
680 void test() {
681     // Arrange
682     // Act
683     // Assert
684 }
685
686 * AllyHyeseongKim
687
688 @Test
689 void test() {
690     // Arrange
691     // Act
692     // Assert
693 }
694
695 * AllyHyeseongKim
696
697 @Test
698 void test() {
699     // Arrange
700     // Act
701     // Assert
702 }
703
704 * AllyHyeseongKim
705
706 @Test
707 void test() {
708     // Arrange
709     // Act
710     // Assert
711 }
712
713 * AllyHyeseongKim
714
715 @Test
716 void test() {
717     // Arrange
718     // Act
719     // Assert
720 }
721
722 * AllyHyeseongKim
723
724 @Test
725 void test() {
726     // Arrange
727     // Act
728     // Assert
729 }
730
731 * AllyHyeseongKim
732
733 @Test
734 void test() {
735     // Arrange
736     // Act
737     // Assert
738 }
739
740 * AllyHyeseongKim
741
742 @Test
743 void test() {
744     // Arrange
745     // Act
746     // Assert
747 }
748
749 * AllyHyeseongKim
750
751 @Test
752 void test() {
753     // Arrange
754     // Act
755     // Assert
756 }
757
758 * AllyHyeseongKim
759
760 @Test
761 void test() {
762     // Arrange
763     // Act
764     // Assert
765 }
766
767 * AllyHyeseongKim
768
769 @Test
770 void test() {
771     // Arrange
772     // Act
773     // Assert
774 }
775
776 * AllyHyeseongKim
777
778 @Test
779 void test() {
780     // Arrange
781     // Act
782     // Assert
783 }
784
785 * AllyHyeseongKim
786
787 @Test
788 void test() {
789     // Arrange
790     // Act
791     // Assert
792 }
793
794 * AllyHyeseongKim
795
796 @Test
797 void test() {
798     // Arrange
799     // Act
800     // Assert
801 }
802
803 * AllyHyeseongKim
804
805 @Test
806 void test() {
807     // Arrange
808     // Act
809     // Assert
810 }
811
812 * AllyHyeseongKim
813
814 @Test
815 void test() {
816     // Arrange
817     // Act
818     // Assert
819 }
820
821 * AllyHyeseongKim
822
823 @Test
824 void test() {
825     // Arrange
826     // Act
827     // Assert
828 }
829
830 * AllyHyeseongKim
831
832 @Test
833 void test() {
834     // Arrange
835     // Act
836     // Assert
837 }
838
839 * AllyHyeseongKim
840
841 @Test
842 void test() {
843     // Arrange
844     // Act
845     // Assert
846 }
847
848 * AllyHyeseongKim
849
850 @Test
851 void test() {
852     // Arrange
853     // Act
854     // Assert
855 }
856
857 * AllyHyeseongKim
858
859 @Test
860 void test() {
861     // Arrange
862     // Act
863     // Assert
864 }
865
866 * AllyHyeseongKim
867
868 @Test
869 void test() {
870     // Arrange
871     // Act
872     // Assert
873 }
874
875 * AllyHyeseongKim
876
877 @Test
878 void test() {
879     // Arrange
880     // Act
881     // Assert
882 }
883
884 * AllyHyeseongKim
885
886 @Test
887 void test() {
888     // Arrange
889     // Act
890     // Assert
891 }
892
893 * AllyHyeseongKim
894
895 @Test
896 void test() {
897     // Arrange
898     // Act
899     // Assert
900 }
901
902 * AllyHyeseongKim
903
904 @Test
905 void test() {
906     // Arrange
907     // Act
908     // Assert
909 }
910
911 * AllyHyeseongKim
912
913 @Test
914 void test() {
915     // Arrange
916     // Act
917     // Assert
918 }
919
920 * AllyHyeseongKim
921
922 @Test
923 void test() {
924     // Arrange
925     // Act
926     // Assert
927 }
928
929 * AllyHyeseongKim
930
931 @Test
932 void test() {
933     // Arrange
934     // Act
935     // Assert
936 }
937
938 * AllyHyeseongKim
939
940 @Test
941 void test() {
942     // Arrange
943     // Act
944     // Assert
945 }
946
947 * AllyHyeseongKim
948
949 @Test
950 void test() {
951     // Arrange
952     // Act
953     // Assert
954 }
955
956 * AllyHyeseongKim
957
958 @Test
959 void test() {
960     // Arrange
961     // Act
962     // Assert
963 }
964
965 * AllyHyeseongKim
966
967 @Test
968 void test() {
969     // Arrange
970     // Act
971     // Assert
972 }
973
974 * AllyHyeseongKim
975
976 @Test
977 void test() {
978     // Arrange
979     // Act
980     // Assert
981 }
982
983 * AllyHyeseongKim
984
985 @Test
986 void test() {
987     // Arrange
988     // Act
989     // Assert
990 }
991
992 * AllyHyeseongKim
993
994 @Test
995 void test() {
996     // Arrange
997     // Act
998     // Assert
999 }
1000
1001 * AllyHyeseongKim
1002
1003 @Test
1004 void test() {
1005     // Arrange
1006     // Act
1007     // Assert
1008 }
1009
1010 * AllyHyeseongKim
1011
1012 @Test
1013 void test() {
1014     // Arrange
1015     // Act
1016     // Assert
1017 }
1018
1019 * AllyHyeseongKim
1020
1021 @Test
1022 void test() {
1023     // Arrange
1024     // Act
1025     // Assert
1026 }
1027
1028 * AllyHyeseongKim
1029
1030 @Test
1031 void test() {
1032     // Arrange
1033     // Act
1034     // Assert
1035 }
1036
1037 * AllyHyeseongKim
1038
1039 @Test
1040 void test() {
1041     // Arrange
1042     // Act
1043     // Assert
1044 }
1045
1046 * AllyHyeseongKim
1047
1048 @Test
1049 void test() {
1050     // Arrange
1051     // Act
1052     // Assert
1053 }
1054
1055 * AllyHyeseongKim
1056
1057 @Test
1058 void test() {
1059     // Arrange
1060     // Act
1061     // Assert
1062 }
1063
1064 * AllyHyeseongKim
1065
1066 @Test
1067 void test() {
1068     // Arrange
1069     // Act
1070     // Assert
1071 }
1072
1073 * AllyHyeseongKim
1074
1075 @Test
1076 void test() {
1077     // Arrange
1078     // Act
1079     // Assert
1080 }
1081
1082 * AllyHyeseongKim
1083
1084 @Test
1085 void test() {
1086     // Arrange
1087     // Act
1088     // Assert
1089 }
1090
1091 * AllyHyeseongKim
1092
1093 @Test
1094 void test() {
1095     // Arrange
1096     // Act
1097     // Assert
1098 }
1099
1100 * AllyHyeseongKim
1101
1102 @Test
1103 void test() {
1104     // Arrange
1105     // Act
1106     // Assert
1107 }
1108
1109 * AllyHyeseongKim
1110
1111 @Test
1112 void test() {
1113     // Arrange
1114     // Act
1115     // Assert
1116 }
1117
1118 * AllyHyeseongKim
1119
1120 @Test
1121 void test() {
1122     // Arrange
1123     // Act
1124     // Assert
1125 }
1126
1127 * AllyHyeseongKim
1128
1129 @Test
1130 void test() {
1131     // Arrange
1132     // Act
1133     // Assert
1134 }
1135
1136 * AllyHyeseongKim
1137
1138 @Test
1139 void test() {
1140     // Arrange
1141     // Act
1142     // Assert
1143 }
1144
1145 * AllyHyeseongKim
1146
1147 @Test
1148 void test() {
1149     // Arrange
1150     // Act
1151     // Assert
1152 }
1153
1154 * AllyHyeseongKim
1155
1156 @Test
1157 void test() {
1158     // Arrange
1159     // Act
1160     // Assert
1161 }
1162
1163 * AllyHyeseongKim
1164
1165 @Test
1166 void test() {
1167     // Arrange
1168     // Act
1169     // Assert
1170 }
1171
1172 * AllyHyeseongKim
1173
1174 @Test
1175 void test() {
1176     // Arrange
1177     // Act
1178     // Assert
1179 }
1180
1181 * AllyHyeseongKim
1182
1183 @Test
1184 void test() {
1185     // Arrange
1186     // Act
1187     // Assert
1188 }
1189
1190 * AllyHyeseongKim
1191
1192 @Test
1193 void test() {
1194     // Arrange
1195     // Act
1196     // Assert
1197 }
1198
1199 * AllyHyeseongKim
1200
1201 @Test
1202 void test() {
1203     // Arrange
1204     // Act
1205     // Assert
1206 }
1207
1208 * AllyHyeseongKim
1209
1210 @Test
1211 void test() {
1212     // Arrange
1213     // Act
1214     // Assert
1215 }
1216
1217 * AllyHyeseongKim
1218
1219 @Test
1220 void test() {
1221     // Arrange
1222     // Act
1223     // Assert
1224 }
1225
1226 * AllyHyeseongKim
1227
1228 @Test
1229 void test() {
1230     // Arrange
1231     // Act
1232     // Assert
1233 }
1234
1235 * AllyHyeseongKim
1236
1237 @Test
1238 void test() {
1239     // Arrange
1240     // Act
1241     // Assert
1242 }
1243
1244 * AllyHyeseongKim
1245
1246 @Test
1247 void test() {
1248     // Arrange
1249     // Act
1250     // Assert
1251 }
1252
1253 * AllyHyeseongKim
1254
1255 @Test
1256 void test() {
1257     // Arrange
1258     // Act
1259     // Assert
1260 }
1261
1262 * AllyHyeseongKim
1263
1264 @Test
1265 void test() {
1266     // Arrange
1267     // Act
1268     // Assert
1269 }
1270
1271 * AllyHyeseongKim
1272
1273 @Test
1274 void test() {
1275     // Arrange
1276     // Act
1277     // Assert
1278 }
1279
1280 * AllyHyeseongKim
1281
1282 @Test
1283 void test() {
1284     // Arrange
1285     // Act
1286     // Assert
1287 }
1288
1289 * AllyHyeseongKim
1290
1291 @Test
1292 void test() {
1293     // Arrange
1294     // Act
1295     // Assert
1296 }
1297
1298 * AllyHyeseongKim
1299
1300 @Test
1301 void test() {
1302     // Arrange
1303     // Act
1304     // Assert
1305 }
1306
1307 * AllyHyeseongKim
1308
1309 @Test
1310 void test() {
1311     // Arrange
1312     // Act
1313     // Assert
1314 }
1315
1316 * AllyHyeseongKim
1317
1318 @Test
1319 void test() {
1320     // Arrange
1321     // Act
1322     // Assert
1323 }
1324
1325 * AllyHyeseongKim
1326
1327 @Test
1328 void test() {
1329     // Arrange
1330     // Act
1331     // Assert
1332 }
1333
1334 * AllyHyeseongKim
1335
1336 @Test
1337 void test() {
1338     // Arrange
1339     // Act
1340     // Assert
1341 }
1342
1343 * AllyHyeseongKim
1344
1345 @Test
1346 void test() {
1347     // Arrange
1348     // Act
1349     // Assert
1350 }
1351
1352 * AllyHyeseongKim
1353
1354 @Test
1355 void test() {
1356     // Arrange
1357     // Act
1358     // Assert
1359 }
1360
1361 * AllyHyeseongKim
1362
1363 @Test
1364 void test() {
1365     // Arrange
1366     // Act
1367     // Assert
1368 }
1369
1370 * AllyHyeseongKim
1371
1372 @Test
1373 void test() {
1374     // Arrange
1375     // Act
1376     // Assert
1377 }
1378
1379 * AllyHyeseongKim
1380
1381 @Test
1382 void test() {
1383     // Arrange
1384     // Act
1385     // Assert
1386 }
1387
1388 * AllyHyeseongKim
1389
1390 @Test
1391 void test() {
1392     // Arrange
1393     // Act
1394     // Assert
1395 }
1396
1397 * AllyHyeseongKim
1398
1399 @Test
1400 void test() {
1401     // Arrange
1402     // Act
1403     // Assert
1404 }
1405
1406 * AllyHyeseongKim
1407
1408 @Test
1409 void test() {
1410     // Arrange
1411     // Act
1412     // Assert
1413 }
1414
1415 * AllyHyeseongKim
1416
1417 @Test
1418 void test() {
1419     // Arrange
1420     // Act
1421     // Assert
1422 }
1423
1424 * AllyHyeseongKim
1425
1426 @Test
1427 void test() {
1428     // Arrange
1429     // Act
1430     // Assert
1431 }
1432
1433 * AllyHyeseongKim
1434
1435 @Test
1436 void test() {
1437     // Arrange
1438     // Act
1439     // Assert
1440 }
1441
1442 * AllyHyeseongKim
1443
1444 @Test
1445 void test() {
1446     // Arrange
1447     // Act
1448     // Assert
1449 }
1450
1451 * AllyHyeseongKim
1452
1453 @Test
1454 void test() {
1455     // Arrange
1456     // Act
1457     // Assert
1458 }
1459
1460 * AllyHyeseongKim
1461
1462 @Test
1463 void test() {
1464     // Arrange
1465     // Act
1466     // Assert
1467 }
1468
1469 * AllyHyeseongKim
1470
1471 @Test
1472 void test() {
1473     // Arrange
1474     // Act
1475     // Assert
1476 }
1477
1478 * AllyHyeseongKim
1479
1480 @Test
1481 void test() {
1482     // Arrange
1483     // Act
1484     // Assert
1485 }
1486
1487 * AllyHyeseongKim
1488
1489 @Test
1490 void test() {
1491     // Arrange
1492     // Act
1493     // Assert
1494 }
1495
1496 * AllyHyeseongKim
1497
1498 @Test
1499 void test() {
1500     // Arrange
1501     // Act
1502     // Assert
1503 }
1504
1505 * AllyHyeseongKim
1506
1507 @Test
1508 void test() {
1509     // Arrange
1510     // Act
1511     // Assert
1512 }
1513
1514 * AllyHyeseongKim
1515
1516 @Test
1517 void test() {
1518     // Arrange
1519     // Act
1520     // Assert
1521 }
1522
1523 * AllyHyeseongKim
1524
1525 @Test
1526 void test() {
1527     // Arrange
1528     // Act
1529     // Assert
1530 }
1531
1532 * AllyHyeseongKim
1533
1534 @Test
1535 void test() {
1536     // Arrange
1537     // Act
1538     // Assert
1539 }
1540
1541 * AllyHyeseongKim
1542
1543 @Test
1544 void test() {
1545     // Arrange
1546     // Act
1547     // Assert
1548 }
1549
1550 * AllyHyeseongKim
1551
1552 @Test
1553 void test() {
1554     // Arrange
1555     // Act
1556     // Assert
1557 }
1558
1559 * AllyHyeseongKim
1560
1561 @Test
1562 void test() {
1563     // Arrange
1564     // Act
1565     // Assert
1566 }
1567
1568 * AllyHyeseongKim
1569
1570 @Test
1571 void test() {
1572     // Arrange
1573     // Act
1574     // Assert
1575 }
1576
1577 * AllyHyeseongKim
1578
1579 @Test
1580 void test() {
1581     // Arrange
1582     // Act
1583     // Assert
1584 }
1585
1586 * AllyHyeseongKim
1587
1588 @Test
1589 void test() {
1590     // Arrange
1591     // Act
1592     // Assert
1593 }
1594
1595 * AllyHyeseongKim
1596
1597 @Test
1598 void test() {
1599     // Arrange
1600     // Act
1601     // Assert
1602 }
1603
1604 * AllyHyeseongKim
1605
1606 @Test
1607 void test() {
1608     // Arrange
1609     // Act
1610     // Assert
1611 }
1612
1613 * AllyHyeseongKim
1614
1615 @Test
1616 void test() {
1617     // Arrange
1618     // Act
1619     // Assert
1620 }
1621
1622 * AllyHyeseongKim
1623
1624 @Test
1625 void test() {
1626     // Arrange
1627     // Act
1628     // Assert
1629 }
1630
1631 * AllyHyeseongKim
1632
1633 @Test
1634 void test() {
1635     // Arrange
1636     // Act
1637     // Assert
1638 }
1639
1640 * AllyHyeseongKim
1641
1642 @Test
1643 void test() {
1644     // Arrange
1645     // Act
1646     // Assert
1647 }
1648
1649 * AllyHyeseongKim
1650
1651 @Test
1652 void test() {
1653     // Arrange
1654     // Act
1655     // Assert
1656 }
1657
1658 * AllyHyeseongKim
1659
1660 @Test
1661 void test() {
1662     // Arrange
1663     // Act
1664     // Assert
1665 }
1666
166
```

```

    @Test
    @DisplayName("[action] action 기능 테스트")
    void actionTest() {
        //given
        HashMap<String, Action> gridMenuItems = new HashMap<>();
        gridMenuItems.put("Clear", this.clearAction);
        gridMenuItems.put("Load", this.loadAction);
        gridMenuItems.put("Store", this.storeAction);
        gridMenuItems.put("Exit", this.exitAction);

        HashMap<String, Action> goMenuItems = new HashMap<>();
        goMenuItems.put("Halt", this.haltAction);
        goMenuItems.put("Tick (Single Step)", this.tickAction);
        goMenuItems.put("Agonizing", this.agonizingAction);
        goMenuItems.put("Slow", this.slowAction);
        goMenuItems.put("Medium", this.mediumAction);
        goMenuItems.put("Fast", this.fastAction);

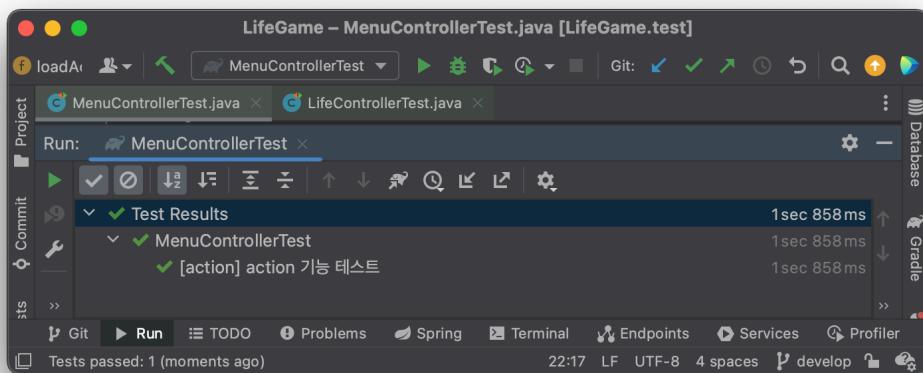
        for (String menuItem : gridMenuItems.keySet()) {
            //when
            menuController.action(menu: "Grid", menuItem);
            //then
            verify(gridMenuItems.get(menuItem)).action();
        }

        for (String menuItem : goMenuItems.keySet()) {
            //when
            menuController.action(menu: "Go", menuItem);
            //then
            verify(goMenuItems.get(menuItem)).action();
        }
    }
}

```

▲ [picture 117] test cases of MenuController class (2)

actionTest는 각 menuItem들의 action을 호출했을 때 올바른 Action이 실행되는지 검증한다.



▲ [picture 118] MenuControllerTest passed

모든 MenuControllerTest가 통과하였음을 확인할 수 있다.

4.7.10. [Test] PaletteControllerTest

PaletteController class 테스트하는 class이다.

```
1 package com.LifeGame.controller;
2
3 import ...;
4
5盟 AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class PaletteControllerTest {
8
9     1 usage
10    @Mock
11    private PatternModel patternModel;
12
13    1 usage
14    @InjectMocks
15    private PaletteController paletteController;
16
17    盟 AllyHyeseongKim
18    @BeforeEach
19    void setUp() {
20
21    }
22
23
24 }
```

Tests passed: 1 (a minute ago) 1:33 LF UTF-8 4 spaces ⚡ develop

▲ [picture 119] test cases of PaletteControl class (1)

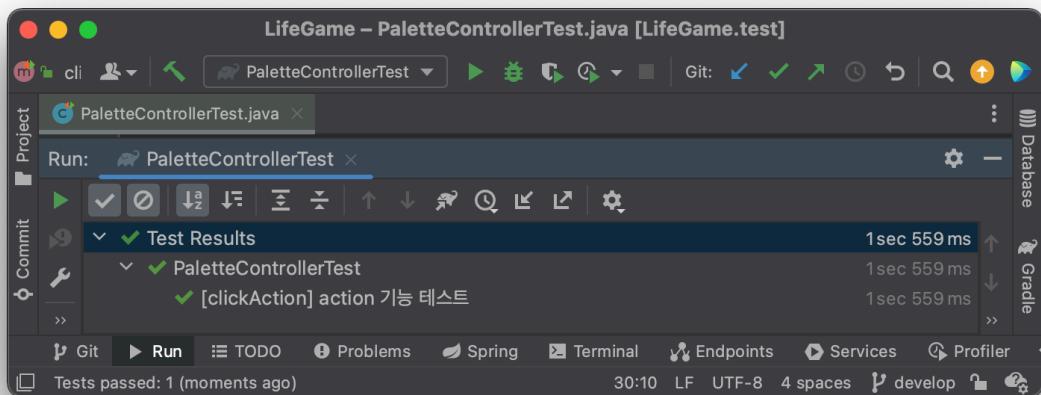
PatternModel을 Mocking하여 테스트하였다.

```
26
27    @Test
28    @DisplayName("[clickAction] action 기능 테스트")
29    void clickActionTest() {
30
31        //given
32        int id = 0;
33        int[][] bluePrint = new int[][] {{0}};
34
35        //when
36        this.paletteController.clickAction(id, bluePrint);
37
38        //then
39        verify(this.patternModel).changePattern(id, bluePrint);
40    }
41
42 }
```

Tests passed: 1 (2 minutes ago) 15:15 LF UTF-8 4 spaces ⚡ develop

▲ [picture 120] test cases of PaletteControl class (2)

clickActionTest는 clickAction이 발생했을 때 PatternModel의 changePattern이 적절하게 호출되었는지 테스트한다.



▲ [picture 121] PaletteControlTeson passed

모든 PaletteControlTest가 통과함을 확인 할 수 있다.

4.7.11. [Test] ClearActionTest

ClearAction class 테스트하는 class이다.

```
1 package com.LifeGame.controller.action;
2
3 import ...
4
5 import AllyHyeseongKim;
6 import org.junit.jupiter.api.extension.ExtendWith;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9
10 import static AllyHyeseongKim.*;
11
12 @ExtendWith(MockitoExtension.class)
13 class ClearActionTest {
14
15     @Mock
16     private Model model;
17
18     @InjectMocks
19     private ClearAction clearAction;
20
21     @BeforeEach
22     void setUp() {
23     }
24 }
```

The code block shows the content of ClearActionTest.java. It imports Model and ClearAction from the com.LifeGame.controller.action package. It uses Mockito annotations: @ExtendWith(MockitoExtension.class), @Mock, @InjectMocks, and @BeforeEach. The class name is ClearActionTest. Inside the class, there are fields for Model and ClearAction, and a setUp() method.

▲ [picture 122] test cases of ClearAction class (1)

Model을 Mocking하여 주입하였다.

```
26  @Test
27  @DisplayName("[action] action 기능 테스트")
28  void actionTest() {
29
30      //given
31      //when
32      this.clearAction.action();
33
34      //then
35      verify(this.model).clearMap();
36  }
37 }
```

Tests passed: 2 (4 minutes ago) 22:16 LF UTF-8 4 spaces ⚡ develop

▲ [picture 123] test cases of ClearAction class (2)

ClearAction을 실행했을 때 Map이 잘 초기화되는지 테스트한다.

Run: ClearActionTest

- Test Results
 - [action] action 기능 테스트 1sec 800 ms

Tests passed: 1 (moments ago)

▲ [picture 124] ClearActionTest passed

모든 ClearActionTest가 통과하였음을 확인할 수 있다.

4.7.12. [Test] LoadActionTest

LoadAction class 테스트하는 class이다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – LoadActionTest.java [LifeGame.test]
- Toolbars:** Action, StoreActionTest, Git, Database, Gradle, Notifications.
- Left Sidebar:** Project, Commit, Pull Requests, Bookmarks, Structure.
- Code Editor:** The file LoadActionTest.java is open, showing test code for the LoadAction class. It includes imports for Mockito, Service, LifePanel, Model, and MenuController, and uses @ExtendWith(MockitoExtension.class) and @InjectMocks annotations.
- Bottom Status Bar:** Tests passed: 1 (3 minutes ago), 74:24 LF, UTF-8, 2 spaces*, develop.

```
1 package com.LifeGame.controller.action;
2
3 import ...
4
5 //盟 AllyHyeseongKim +1
6 @ExtendWith(MockitoExtension.class)
7 class LoadActionTest {
8
9     2 usages
10    @Mock
11    private Service service;
12    @Mock
13    private LifePanel lifePanel;
14    3 usages
15    @Mock
16    private Model model;
17    @Mock
18    private MenuController menuController;
19    2 usages
20    @InjectMocks
21    private LoadAction loadAction;
22
23    //盟 AllyHyeseongKim
24    @BeforeEach
25    void setUp() {
26    }
27}
```

▲ [picture 125] test cases of LoadAction class (1)

Service, LifePanel, Model, MenuController를 Mocking하여 주입하였다.

The screenshot shows a Java IDE interface with the title "LifeGame – LoadActionTest.java [LifeGame.test]". The code editor displays a test class named "LoadActionTest.java" with the following content:

```
49     @Test
50     @DisplayName("[action/try] action 기능 테스트")
51     void actionTryTest() {
52         //given
53         int mapSize = 0;
54         int[][] liveCells = {{0, 0}};
55         MapData mapData = new MapData(mapSize, liveCells);
56         when(this.service.load()).thenReturn(mapData);
57
58         //when
59         this.loadAction.action();
56
59         //then
60         verify(this.model).clearMap();
61         verify(this.model).setMap(liveCells);
62     }
63 }
```

The IDE's status bar at the bottom indicates "Tests passed: 1 (3 minutes ago)".

▲ [picture 126] test cases of LoadAction class (2)

LoadAction이 동작할 때 Mocking된 Service에서 받아온 값을 Model에 잘 반영하는지 테스트한다.

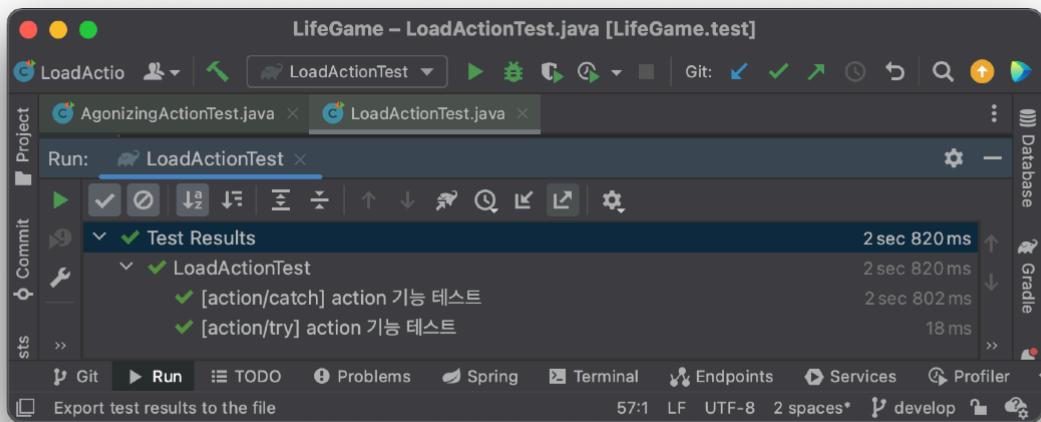
The screenshot shows a Java IDE interface with the title "LifeGame – LoadActionTest.java [LifeGame.test]". The code editor displays a test class named "LoadActionTest.java" with the following content:

```
66     @Test
67     @DisplayName("[action/catch] action 기능 테스트")
68     void actionCatchTest() throws AWTException {
69         //given
70         MockedStatic<JOptionPane> ignoreOption = Mockito.mockStatic(JOptionPane.class);
71         when(this.service.load()).thenThrow(new InvalidFileLoadedException());
72
73         //when
74         this.loadAction.action();
75         //then
76         verify(this.model).clearMap();
77
78         //cleanUp
79         ignoreOption.close();
80     }
81 }
```

The IDE's status bar at the bottom indicates "Tests passed: 1 (4 minutes ago)".

▲ [picture 127] test cases of LoadAction class (3)

actionCatchTest는 잘못된 파일을 읽으려 할 때 model이 잘 클리어 되는지 테스트한다.



▲ [picture 128] LoadActionTest passed

모든 LoadActionTest가 통과하였음을 확인 할 수 있다.

4.7.13. [Test] StoreActionTest

StoreAction class 테스트하는 class이다.

```
1 package com.LifeGame.controller.action;
2
3 import ...
4
5 ▲ AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class StoreActionTest {
8
9     1 usage
10    @Mock
11    private Service service;
12    2 usages
13    @Mock
14    private Model model;
15    @Mock
16    private MenuController menuController;
17    1 usage
18    @InjectMocks
19    private StoreAction storeAction;
20
21    ▲ AllyHyeseongKim
22    @BeforeEach
23    void setUp() {
24    }
25
26
27
28
29
30
31
32
33 }
```

Tests passed: 1 (3 minutes ago) 47:1 LF UTF-8 4 spaces develop

▲ [picture 129] test cases of StoreAction class (1)

Service, Model, MeuController를 Mocking하였다.

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** LifeGame – StoreActionTest.java [LifeGame.test]
- Toolbars:** Git, Run, TODO, Problems, Spring, Terminal, Endpoints, Services, Profiler.
- Left Sidebar:** Project, Commit, Pull Requests, Bookmarks, Structure.
- Central Area:** Code editor with the following code:

```
35      @Test
36      @DisplayName("[action] action 기능 테스트")
37      void actionTest() {
38
39          //given
40          int[][] map = {{0, 0}};
41          int mapSize = 0;
42          when(this.model.getMap()).thenReturn(map);
43          when(this.model.getMapSize()).thenReturn(mapSize);
44
45          //when
46          this.storeAction.action();
47
48          //then
49          verify(this.service).store(argThat(mapData -> {
50              Assertions.assertEquals(mapSize, mapData.getMapSize());
51              Assertions.assertEquals(map, mapData.getMap());
52              return true;
53          }));
54      }
55  }
```

- Bottom Status Bar:** Tests passed: 1 (4 minutes ago), 27:16, LF, UTF-8, 4 spaces, develop.

▲ [picture 130] test cases of StoreAction class (2)

StoreAction이 실행될 때 적절한 데이터를 Service에 요청하는지 테스트한다.

The screenshot shows a Java IDE interface with the following details:

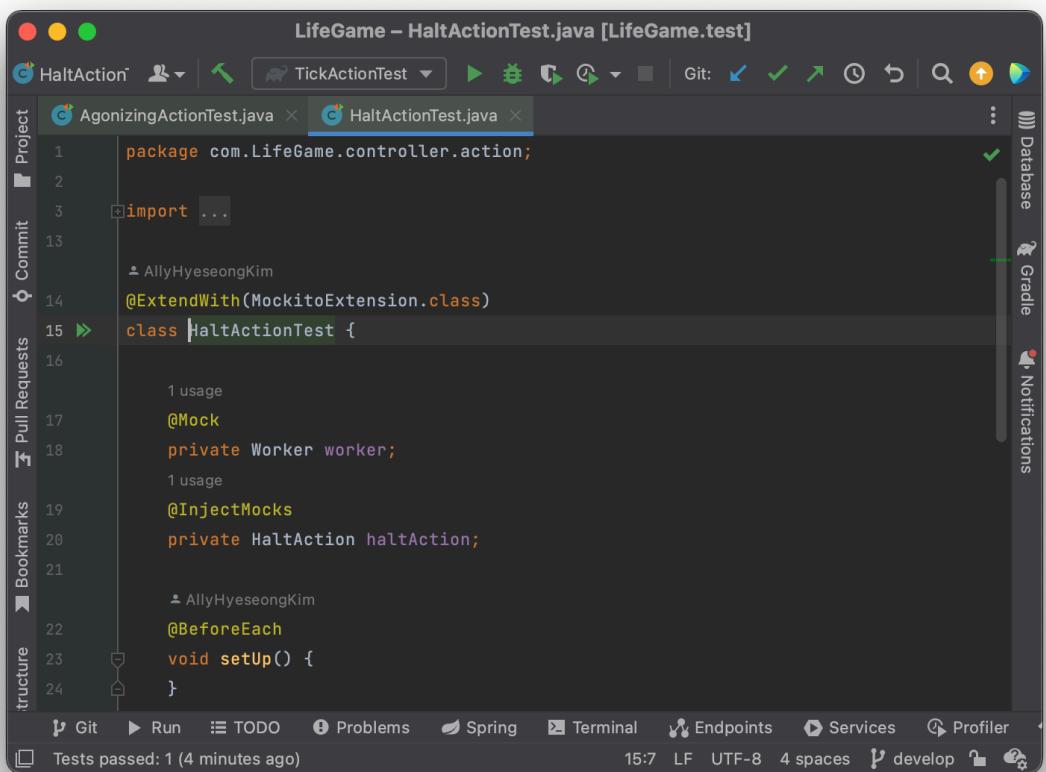
- Title Bar:** LifeGame – StoreActionTest.java [LifeGame.test]
- Toolbars:** Git, Run, TODO, Problems, Spring, Terminal, Endpoints, Services, Profiler.
- Left Sidebar:** Project, Commit, Pull Requests, Bookmarks, Structure.
- Central Area:** Run: StoreActionTest x
- Run Results:** Test Results, StoreActionTest, [action] action 기능 테스트, 1sec 812 ms, 1sec 812 ms, 1sec 812 ms.
- Bottom Status Bar:** Tests passed: 1 (moments ago), 43:18, LF, UTF-8, 4 spaces, develop.

▲ [picture 131] StoreActionTest passed

StoreActionTest의 모든 Test가 통과하는 것을 확인 할 수 있다.

4.7.14. [Test] HaltActionTest

HaltAction class 테스트하는 class이다.

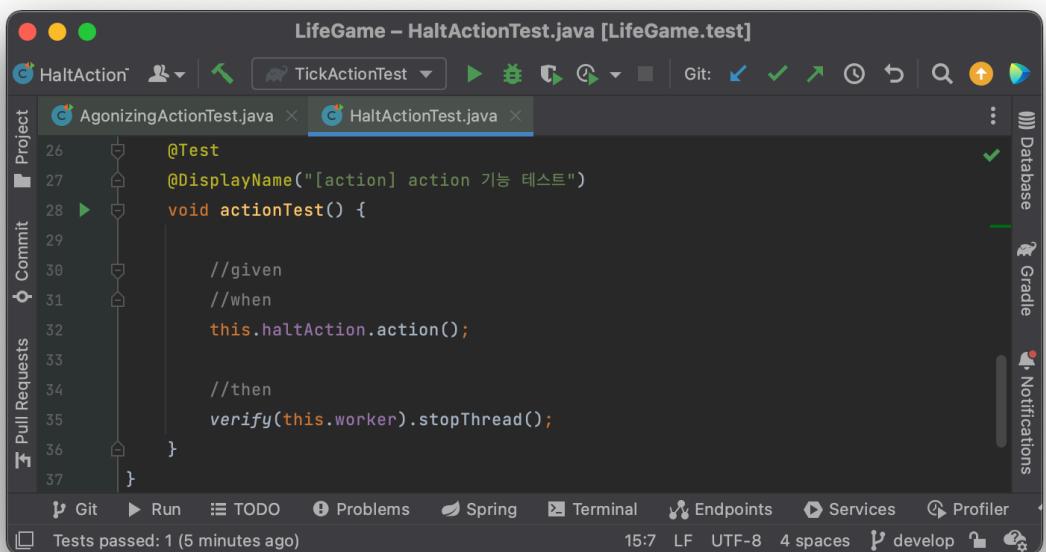


```
1 package com.LifeGame.controller.action;
2
3 import ...;
4
5 // AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class HaltActionTest {
8
9     // Mock
10    private Worker worker;
11
12    // Inject Mocks
13    private HaltAction haltAction;
14
15    // AllyHyeseongKim
16    @BeforeEach
17    void setUp() {
18    }
19
20
21
22
23
24 }
```

Tests passed: 1 (4 minutes ago)

▲ [picture 132] test cases of HaltAction class (1)

Worker를 Mocking하여 주입하여 테스트를 진행하였다.

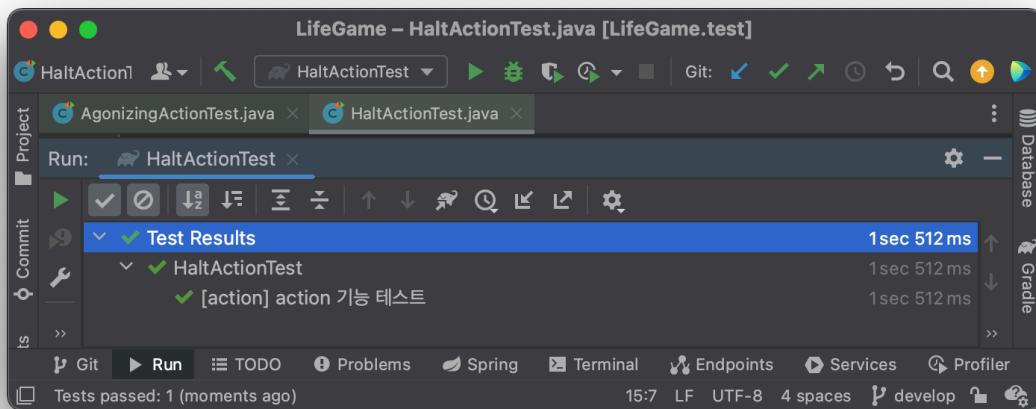


```
1 package com.LifeGame.controller.action;
2
3 import ...;
4
5 // Test
6 // DisplayName("[action] action 기능 테스트")
7 void actionTest() {
8
9     //given
10    //when
11    this.haltAction.action();
12
13    //then
14    verify(this.worker).stopThread();
15
16 }
```

Tests passed: 1 (5 minutes ago)

▲ [picture 133] test cases of HaltAction class (2)

actionTest는 HaltAction이 실행될 때 worker의 stopThread를 잘 호출하는지 테스트한다.



▲ [picture 134] HaltActionTest passed

HaltActionTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.15. [Test] TickActionTest

TickAction class 테스트하는 class이다.

A screenshot of the IntelliJ IDEA interface. The title bar says "LifeGame - TickActionTest.java [LifeGame.test]". The code editor shows the following Java code:

```
package com.LifeGame.controller.action;

import ...;

import AllyHyeseongKim;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import static AllyHyeseongKim.*;

@ExtendWith(MockitoExtension.class)
class TickActionTest {

    @Mock
    private Model model;

    @InjectMocks
    private TickAction tickAction;

    @BeforeEach
    void setUp() {
    }
}
```

The code uses Mockito annotations like @ExtendWith, @Mock, @InjectMocks, and @BeforeEach. The status bar at the bottom shows "Tests passed: 1 (2 minutes ago)" and the current time as 16:7.

▲ [picture 135] test cases of TickAction class (1)

Model을 Mocking하여 주입하여 테스트를 진행하였다.

```
LifeGame – TickActionTest.java [LifeGame.test]
AgonizingActionTest | TickActionTest.java
Project Commit Pull Requests
27
28
29
30
31
32
33
34
35
36
37
38
AllyHyeseongKim
@Test
@DisplayName("[action] action 가능 테스트")
void actionTest() {
    //given
    //when
    this.tickAction.action();

    //then
    verify(this.model).nextState();
}
ks
```

Tests passed: 1 (3 minutes ago) 21:14 LF UTF-8 4 spaces develop

▲ [picture 136] test cases of TickAction class (2)

```
LifeGame – TickActionTest.java [LifeGame.test]
ActionTest | TickActionTest.java
Project Commit Pull Requests
Run: TickActionTest
Test Results
TickActionTest
[action] action 가능 테스트 1sec 641ms
1sec 641ms
1sec 641ms
```

Tests passed: 1 (moments ago) 29:23 LF UTF-8 4 spaces develop

▲ [picture 137] TickActionTest passed

TickActionTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.16. [Test] AgonizingActionTest

AgonizingAction class 테스트하는 class이다.

```
package com.LifeGame.controller.action;

import ...;

@ExtendWith(MockitoExtension.class)
class AgonizingActionTest {

    @Mock
    private Worker worker;

    @InjectMocks
    private AgonizingAction agonizingAction;

    void setUp() {
    }

    @Test
    @DisplayName("[action] action 기능 테스트")
    void actionTest() {
        //given
        //when
        this.agonizingAction.action();

        //then
        verify(this.worker).setSpeed(500);
        verify(this.worker).startThread();
    }
}
```

▲ [picture 138] test cases of AgonizingAction class (1)

```
package com.LifeGame.controller.action;

import ...;

@ExtendWith(MockitoExtension.class)
class AgonizingActionTest {

    @Mock
    private Worker worker;

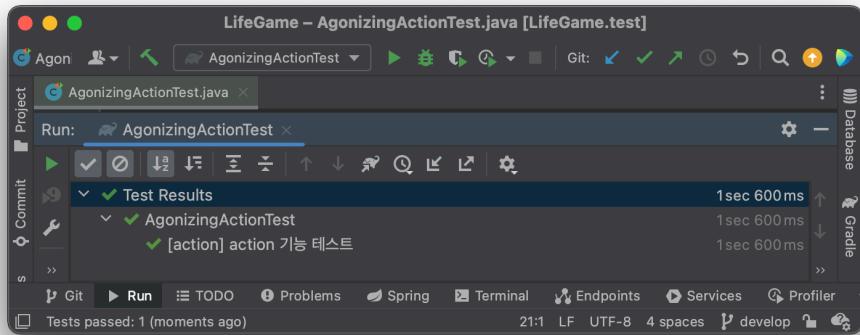
    @InjectMocks
    private AgonizingAction agonizingAction;

    void setUp() {
    }

    @Test
    @DisplayName("[action] action 기능 테스트")
    void actionTest() {
        //given
        //when
        this.agonizingAction.action();

        //then
        verify(this.worker).setSpeed(500);
        verify(this.worker).startThread();
    }
}
```

▲ [picture 139] test cases of AgonizingAction class (2)



▲ [picture 140] AgonizingActionTest passed

AgonizingActionTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.17. [Test] SlowActionTest

SlowAction class 테스트하는 class이다.

```
1 package com.LifeGame.controller.action;
2
3 import ...;
4
5 import AllyHyeseongKim;
6 import org.junit.jupiter.api.extension.ExtendWith;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9 import org.mockito.MockitoAnnotations;
10 import org.mockito.junit.jupiter.MockitoExtension;
11
12 import static AllyHyeseongKim.*;
13
14 @ExtendWith(MockitoExtension.class)
15 class SlowActionTest {
16
17     @Mock
18     private Worker worker;
19
20     @InjectMocks
21     private SlowAction slowAction;
22
23     @BeforeEach
24     void setUp() {
```

The screenshot shows the IntelliJ IDEA interface with the title bar "LifeGame – SlowActionTest.java [LifeGame.test]". The code editor displays the "SlowActionTest.java" file. The code defines a test class `SlowActionTest` with annotations `@ExtendWith(MockitoExtension.class)` and `@BeforeEach`. It imports `Worker` and `SlowAction` from the `com.LifeGame.controller.action` package, and `MockitoAnnotations` from `org.mockito.junit.jupiter`. The `setUp` method is defined to set up the `worker` and `slowAction` instances.

▲ [picture 141] test cases of SlowAction class (1)

The screenshot shows the IntelliJ IDEA interface with the project 'SlowActionT' open. The current file is 'SlowActionTest.java' under the package 'LifeGame'. The code contains a single test method:

```
26  * @Test
27  * @DisplayName("[action] action 기능 테스트")
28  void actionTest() {
29
30      //given
31      //when
32      this.slowAction.action();
33
34      //then
35      verify(this.worker).setSpeed(150);
36      verify(this.worker).startThread();
37
38 }
```

The code editor has syntax highlighting for Java and annotations. The status bar at the bottom indicates 'Unable to save plugin settings: The plugin com.intellij... (3 minutes ago)'. The bottom right corner shows the date and time as 12:42 LF UTF-8 4 spaces.

▲ [picture 142] test cases of SlowAction class (2)

The screenshot shows the IntelliJ IDEA interface with the project 'SlowActionT' open. The current file is 'SlowActionTest.java' under the package 'LifeGame'. The 'Run' tool window is open, showing the 'Test Results' section. It displays the following test results:

Test	Time
SlowActionTest	1sec 627 ms
[action] action 기능 테스트	1sec 627 ms
[action] action 기능 테스트	1sec 627 ms

The status bar at the bottom indicates 'Tests passed: 1 (moments ago)'. The bottom right corner shows the date and time as 12:42 LF UTF-8 4 spaces.

▲ [picture 143] SlowActionTest passed

SlowTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.18. [Test] FastActionTest

FastAction class 테스트하는 class이다.

The screenshot shows a Java IDE interface with the title "LifeGame – FastActionTest.java [LifeGame.test]". The code editor displays a test method named "actionTest" in the "FastActionTest.java" file. The code is as follows:

```
26     @Test
27     @DisplayName("[action] action 기능 테스트")
28     void actionTest() {
29
30         //given
31         //when
32         this.fastAction.action();
33
34         //then
35         verify(this.worker).setSpeed(30);
36         verify(this.worker).startThread();
37     }
38 }
```

The IDE's status bar at the bottom indicates "Tests passed: 1 (2 minutes ago)".

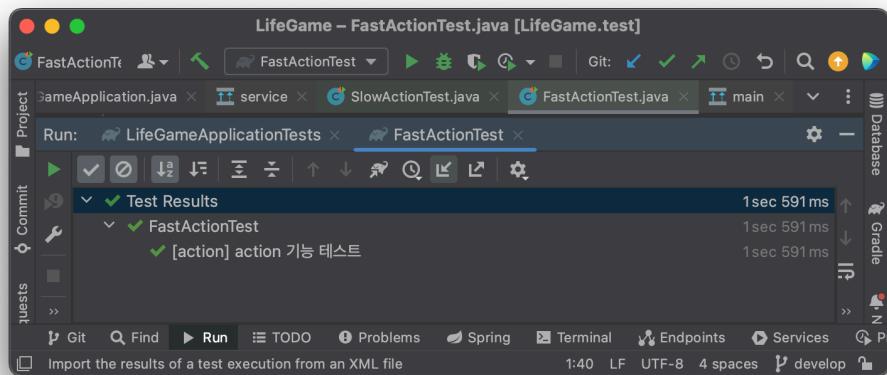
▲ [picture 144] test cases of FastAction class (1)

The screenshot shows a Java IDE interface with the title "LifeGame – FastActionTest.java [LifeGame.test]". The code editor displays the "FastActionTest" class definition. The code is as follows:

```
1 package com.LifeGame.controller.action;
2
3 import ...
4
5
6     @ExtendWith(MockitoExtension.class)
7     class FastActionTest {
8
9         @Mock
10        private Worker worker;
11
12        @InjectMocks
13        private FastAction fastAction;
14
15        @BeforeEach
16        void setUp() {
17    }
18
19
20
21
22
23
24 }
```

The IDE's status bar at the bottom indicates "Tests passed: 1 (a minute ago)".

▲ [picture 145] test cases of FastAction class (2)



▲ [picture 146] FastActionTest passed

FastActionTest의 모든 Test가 통과하는 것을 확인 할 수 있다.

4.7.19. [Test] MediumActionTest

MediumAction class 테스트하는 class이다.

```
1 package com.LifeGame.controller.action;
2
3 import ...
4
5 import AllyHyeseongKim;
6
7 @ExtendWith(MockitoExtension.class)
8 class MediumActionTest {
9
10    2 usages
11    @Mock
12    private Worker worker;
13
14    1 usage
15    @InjectMocks
16    private MediumAction mediumAction;
17
18    import AllyHyeseongKim;
19    @BeforeEach
20    void setUp() {
21    }
22
23
24 }
```

▲ [picture 147] test cases of MediumAction class (1)

Worker를 Mockito로 Mocking하여 MediumAction에 주입하였다.

```
26     @Test
27     @DisplayName("[action] action 기능 테스트")
28     void actionTest() {
29
30         //given
31         //when
32         this.mediumAction.action();
33
34         //then
35         verify(this.worker).setSpeed(70);
36         verify(this.worker).startThread();
37     }
38 }
```

▲ [picture 148] test cases of MediumAction class (2)

actionTest는 MediumAction의 실행될 때 speed가 70으로 잘 설정되고 스레드가 동작하는지 검증한다.

Run: LifeGameApplicationTests > MediumActionTest

Test	Time
Test Results	1sec 556 ms
MediumActionTest	1sec 556 ms
[action] action 기능 테스트	1sec 556 ms

▲ [picture 149] MediumActionTest passed

MediumActionTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.20. [Test] DefaultDrawBehaviorTest

DefaultDrawBehavior class 테스트하는 class이다.

```
LifeGame – DefaultDrawBehaviorTest.java [LifeGame.test]
1 package com.LifeGame.controller.drawBehavior;
2
3 import ...
4
5 * AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class DefaultDrawBehaviorTest {
8
9     1 usage
10    @Mock
11    private Model model;
12
13    1 usage
14    @InjectMocks
15    private DefaultDrawBehavior defaultDrawBehavior;
16
17    * AllyHyeseongKim
18    @BeforeEach
19    void setUp() {
20    }
21
22
23
24 }
```

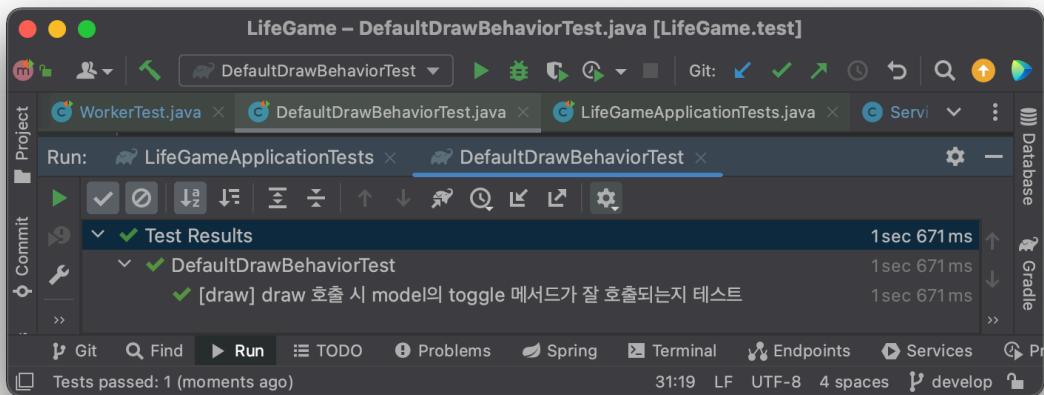
▲ [picture 150] test cases of DefaultDrawBehavior class (1)

Model을 Mocking하여 주입하였다

```
LifeGame – DefaultDrawBehaviorTest.java [LifeGame.test]
1 package com.LifeGame.controller.drawBehavior;
2
3 import ...
4
5 * AllyHyeseongKim
6 @ExtendWith(MockitoExtension.class)
7 class DefaultDrawBehaviorTest {
8
9     @Test
10    @DisplayName("[draw] draw 호출 시 model의 toggle 메서드가 잘 호출되는지 테스트")
11    void drawTest() {
12
13        //given
14        int x = 0;
15        int y = 0;
16
17        //when
18        this.defaultDrawBehavior.draw(x, y);
19
20        //then
21        verify(this.model).toggle(x, y);
22    }
23
24 }
```

▲ [picture 151] test cases of DefaultDrawBehavior class (2)

drawTest는 Model의 올바른 좌표값을 넘겨 toggle을 실행하는지 테스트한다.



▲ [picture 152] DeafultDrawBehaviorTest passed

DefaultDrawBehaviorTest의 모든 Test가 통과하는 것을 확인 할 수 있다.

4.7.21. [Test] PatternDrawBehaviorTest

PatternDrawBehavior class 테스트하는 class이다.

```

1 package com.LifeGame.controller.drawBehavior;
2
3 import ...;
4
5 import AllyHyeseongKim;
6 import org.junit.jupiter.api.extension.ExtendWith;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9 import org.mockito.MockitoAnnotations;
10 import org.mockito.junit.jupiter.MockitoExtension;
11
12 import static AllyHyeseongKim.*;
13
14 /**
15  * @ExtendWith(MockitoExtension.class)
16 */
17 class PatternDrawBehaviorTest {
18
19     @Mock
20     private Model model;
21
22     @Mock
23     private PatternModel patternModel;
24
25     @BeforeEach
26     void setUp() {
27
28     }
29 }

```

The screenshot shows the IntelliJ IDEA interface with the title bar "LifeGame – PatternDrawBehaviorTest.java [LifeGame.test]". The code editor displays the implementation of the PatternDrawBehaviorTest class. It uses Mockito annotations (@Mock, @InjectMocks) to mock the Model and PatternModel classes. The code editor also shows some code completion suggestions for the @Mock annotation. The status bar at the bottom indicates "Unable to save plugin settings: The plugin com.intellij.... (moments ago)" and the current time as 20:10.

▲ [picture 153] test cases of PatternDrawBehavior class (1)

Model, PatternModel을 Mocking한다.

```
LifeGame – PatternDrawBehaviorTest.java [LifeGame.test]
@DisplayName("[draw] draw 호출 시 model의 toggle 메서드가 잘 호출되는지 테스트")
void drawTest() {
    //given
    int x = 0;
    int y = 0;

    //when
    this.patternDrawBehavior.draw(x, y);

    //then
    verify(this.model).draw(this.patternModel.getBlueprint(), x, y);
}
```

▲ [picture 154] test cases of PatternDrawBehavior class (2)

drawTest는 Model의 draw에 올바른 bluePrint와 x, y 좌표를 설정해 실행하는지 테스트한다.

Test	Time
[draw] draw 호출 시 model의 toggle 메서드가 잘 호출되는지 테스트	1sec 684 ms

▲ [picture 155] PatternDrawBehaviorTest passed

PatternDrawBehaviorTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.22. [Test] ServiceTest

Service class 테스트하는 class이다.

LifeGame – ServiceTest.java [LifeGame.test]

```
package com.LifeGame.service;

import ...

@ExtendWith(MockitoExtension.class)
public class ServiceTest {

    private static String TEST_FILE_NAME = "test.json";

    @Spy
    private Service service;

    void setup() throws IOException {
        File testFile = new File(TEST_FILE_NAME);
        if (testFile.exists()) {
            if (!testFile.delete()) {
                throw new RuntimeException("cannot delete existed test file");
            }
        }

        File newTestFile = new File(TEST_FILE_NAME);
        if (!newTestFile.createNewFile()) {
            throw new RuntimeException("cannot create new test file");
        }
        Mockito.doReturn(newTestFile).when(service).userSelected(any(), any(),
            any(), any());
    }

    static void cleanUp(){
        File testFile = new File(TEST_FILE_NAME);
        if (testFile.exists()) {
            if (!testFile.delete()) {
```

▲ [picture 156] test cases of Service class (1)

UI와 분리하여 테스트 하기 위해 @BeforeEach setUp()에서 mock 테스트 파일을 선언하고 Mockito를 사용하여 userSelected를 처리한다. @AfterAll cleanUp()에서 테스트에서 만든 mock 파일을 삭제한다.

The screenshot shows the IntelliJ IDEA interface with the code editor open to `ServiceTest.java`. The code defines a test method `loadTest()` that creates a random map data, writes it to a file, and then loads it back from the file. It uses `Assertions.assertEquals` to compare the original and loaded map data strings.

```
56     @Test
57     @DisplayName("Load 기능 테스트")
58     void loadTest() throws IOException {
59         // when
60         MapData randomMapData = createRandomMapData();
61         File testFile = new File(TEST_FILE_NAME);
62         Files.writeString(testFile.toPath(), mapToString(randomMapData));
63         MapData loadedData = service.load();
64
65         // then
66         Assertions.assertEquals(
67             mapToString(randomMapData),
68             mapToString(loadedData),
69             message: "loaded map must equal generated map");
70     }
```

▲ [picture 157] test cases of Service class (2)

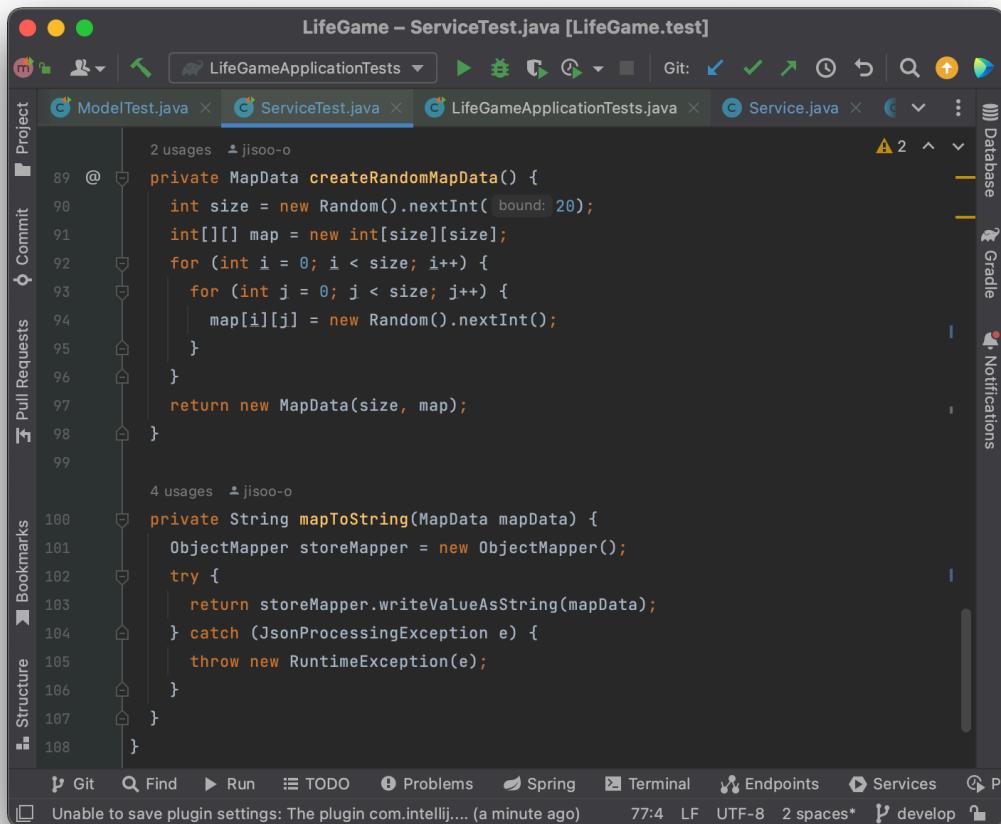
`createRandomMapData()`에서 만든 `randomMapData`에 대한 `store`, `load` 테스트를 진행한다. 테스트 파일의 값과 불러온 값이 일치하는지 `assertEquals`로 확인한다.

The screenshot shows the IntelliJ IDEA interface with the code editor open to `ServiceTest.java`. The code defines a test method `storeTest()` that creates a random map data, stores it to a file, and then reads it back to verify its contents.

```
72     @Test
73     @DisplayName("Store 기능 테스트")
74     void storeTest() throws IOException {
75         // when
76         MapData randomMapData = createRandomMapData();
77         service.store(randomMapData);
78
79         // then
80         File testFile = new File(TEST_FILE_NAME);
81         assertTrue(testFile.exists(), message: "test file must be generated");
82         List<String> lines = Files.readAllLines(testFile.toPath());
83         Assertions.assertEquals(lines.size(), actual: 1,
84             message: "file must contain 1 line string");
85         Assertions.assertEquals(lines.get(0), mapToString(randomMapData),
86             message: "saved map must equal generated map");
87     }
```

▲ [picture 158] test cases of Service class (3)

storeTest는 Service의 store method가 호출되었을 때 올바른 파일명에 올바른 파일 내용으로 덤프 파일이 생성되었는지 검증하는 테스트이다.

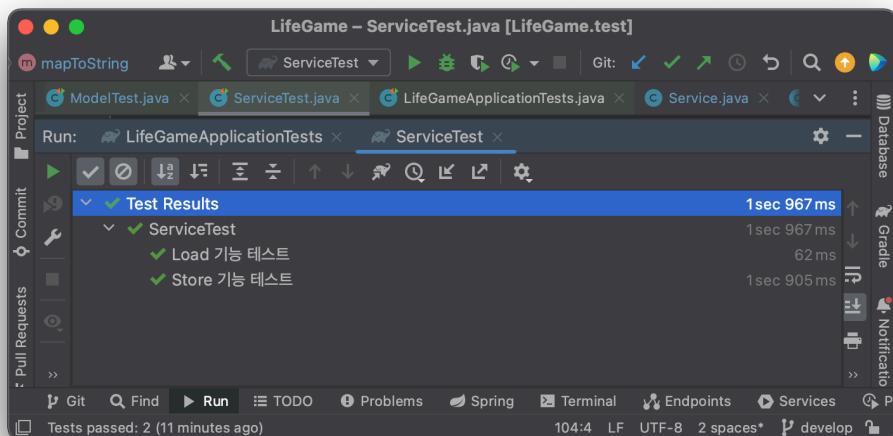


```
private MapData createRandomMapData() {
    int size = new Random().nextInt( bound: 20 );
    int[][] map = new int[size][size];
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            map[i][j] = new Random().nextInt();
        }
    }
    return new MapData(size, map);
}

private String mapToString(MapData mapData) {
    ObjectMapper storeMapper = new ObjectMapper();
    try {
        return storeMapper.writeValueAsString(mapData);
    } catch (JsonProcessingException e) {
        throw new RuntimeException(e);
    }
}
```

▲ [picture 159] test cases of Service class

storeTest와 loadTest에 사용되는 랜덤한 맵 데이터를 생성하는 createRandomMapData 메소드와 storeMapper를 이용하여 객체를 string 타입으로 변환하는 mapToString 메소드이다.



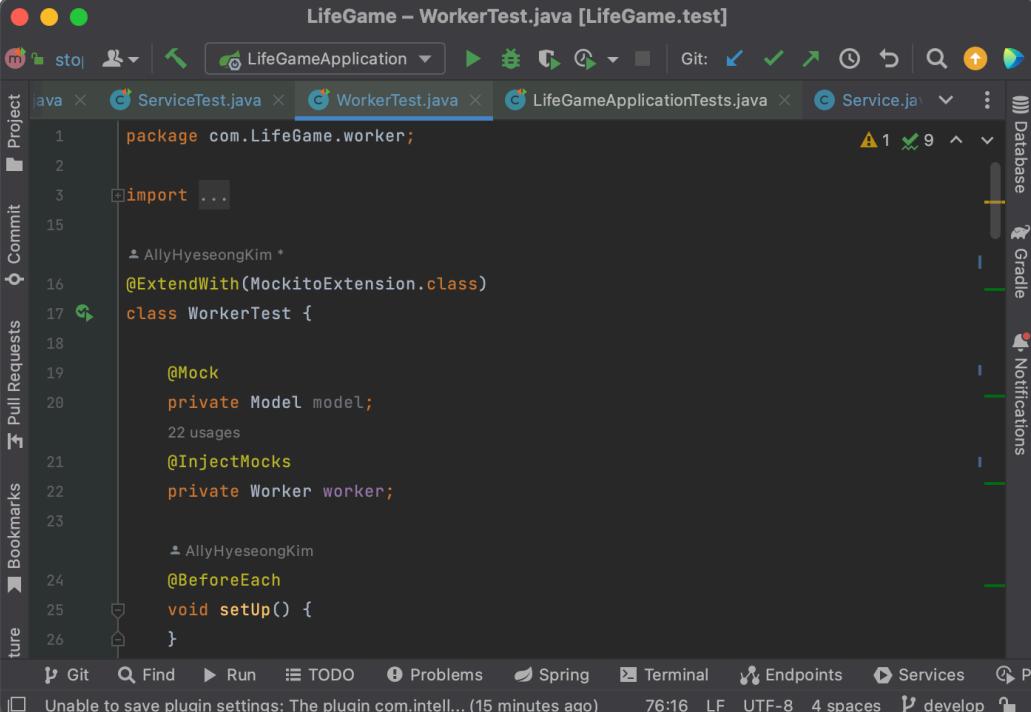
Test Results
ServiceTest
Load 기능 테스트
Store 기능 테스트

▲ [picture 160] ServiceTest passed

ServiceTest의 모든 Test가 통과하는 것을 확인할 수 있다.

4.7.23. [Test] WorkerTest

Worker class 테스트하는 class이다.



The screenshot shows the IntelliJ IDEA interface with the title bar "LifeGame – WorkerTest.java [LifeGame.test]". The code editor displays the following Java test code:

```
package com.LifeGame.worker;

import ...;

@ExtendWith(MockitoExtension.class)
class WorkerTest {

    @Mock
    private Model model;
    22 usages
    @InjectMocks
    private Worker worker;

    @BeforeEach
    void setUp() {
    }
}
```

The code uses Mockito annotations (@Mock, @InjectMocks) and extends the MockitoExtension. The Model class is mocked, and the Worker class is injected. A setup method is defined to initialize the test environment.

▲ [picture 161] test cases of Worker class (1)

Model을 Mocking하여 Worker에 주입하여 테스트한다.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** LifeGame – WorkerTest.java [LifeGame.test]
- Toolbars:** Git, Find, Run, TODO, Problems, Spring, Terminal, Endpoints, Services.
- Code Editor:** The current file is WorkerTest.java, which contains Java test code for the Worker class. The code is annotated with JUnit annotations (@Test, @DisplayName) and assertions (assertTrue, assertEquals). A Korean comment is present in the code.

```
28     @Test
29     @DisplayName("[startThread/alive] 생성되었던 thread를 중단시키고 새로운 thread를 시작되는지 테스트")
30         "잘 시작되는지 테스트"
31     void startAliveThreadTest() throws NoSuchFieldException,
32             IllegalAccessException {
33
34         //given
35         Field field = this.worker.getClass().getDeclaredField("thread");
36         field.setAccessible(true);
37
38         this.worker.setSpeed(100);
39         this.worker.startThread();
40
41         Thread prev = (Thread) field.get(this.worker);
42
43         //when
44         this.worker.startThread();
45
46         //then
47         assertTrue(((Thread) field.get(this.worker)).isAlive());
48         assertEquals(prev, field.get(this.worker));
49     }
```
- Side Panels:** Project, Commit, Pull Requests, Bookmarks, Structure, Database, Gradle, Notifications.
- Bottom Status Bar:** Unable to save plugin settings: The plugin com.intellij... (2 minutes ago), 22:15, LF, UTF-8, 4 spaces, develop.

▲ [picture 162] test cases of Worker class (2)

startAliveThreadTest는 기존에 존재하는 스레드를 중단하고 새로운 스레드를 잘 실행하는지 테스트한다.

현재 worker의 스레드가 alive인지, 예전 스레드와 인스턴스가 다른지를 assert한다.

The screenshot shows the IntelliJ IDEA interface with the project 'LifeGame' open. The current file is 'WorkerTest.java'. The code is a JUnit test for the 'Worker' class, specifically testing the 'startThread' method. The test ensures that when a new thread is created, it has the correct speed setting (100). The code is annotated with JUnit annotations (@Test, @DisplayName) and includes assertions to check if the thread is alive and has the correct speed.

```
51  * @Test
52  * @DisplayName("[startThread/not alive] 생성된 thread가 없을 때 새로운 thread가 " +
53  *             "잘 시작되는지 테스트")
54  void startThreadTest() throws NoSuchFieldException, IllegalAccessException {
55
56      //given
57      Field field = this.worker.getClass().getDeclaredField( name: "thread");
58      field.setAccessible(true);
59
60      this.worker.setSpeed(100);
61
62      //when
63      this.worker.startThread();
64
65      //then
66      assertTrue(((Thread) field.get(this.worker)).isAlive());
67 }
```

▲ [picture 163] test cases of Worker class (3)

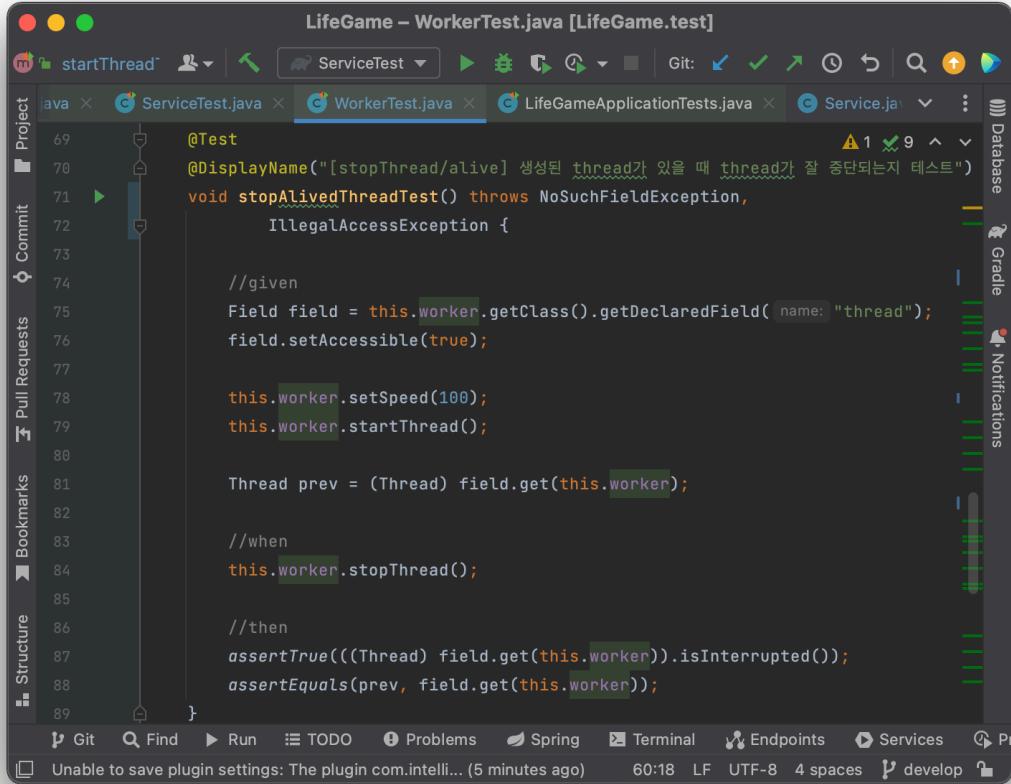
startThreadTest는 기존에 스레드가 없을 때 새로운 스레드가 올바른 speed 설정 값으로 잘 생성되는지 테스트한다.

The screenshot shows the IntelliJ IDEA interface with the project 'LifeGame' open. The current file is 'WorkerTest.java'. The code is a JUnit test for the 'Worker' class, specifically testing the 'stopThread' method. The test ensures that when a thread is stopped, its value is null. The code is annotated with JUnit annotations (@Test, @DisplayName) and includes assertions to check if the field value is null after calling stopThread.

```
91  * @Test
92  * @DisplayName("[stopThread/not alive] 생성된 thread가 없을 때 잘 무시하는지 테스트")
93  void stopThreadTest() throws NoSuchFieldException, IllegalAccessException {
94
95      //given
96      Field field = this.worker.getClass().getDeclaredField( name: "thread");
97      field.setAccessible(true);
98
99      this.worker.setSpeed(100);
100
101     //when
102     this.worker.stopThread();
103
104     //then
105     assertNull(field.get(this.worker));
106 }
107 }
```

▲ [picture 164] test cases of Worker class (4)

stopThreadTest는 생성된 스레드가 없을 때 stopThread가 호출되어도 잘 무시되는지 테스트한다.



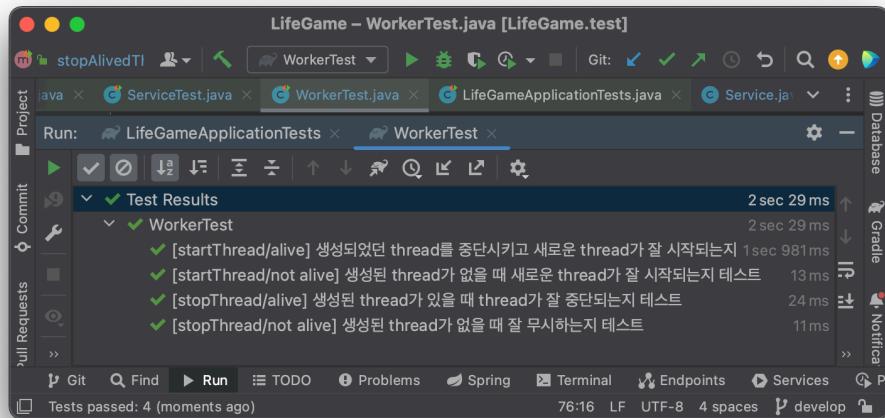
```
LifeGame – WorkerTest.java [LifeGame.test]
startThread* ServiceTest WorkerTest.java LifeGameApplicationTests.java Service.java ...
Project 69
70 @Test
71 @DisplayName("[stopThread/alive] 생성된 thread가 있을 때 thread가 잘 중단되는지 테스트")
72 void stopAliveThreadTest() throws NoSuchFieldException,
73     IllegalAccessException {
74
75     //given
76     Field field = this.worker.getClass().getDeclaredField( name: "thread");
77     field.setAccessible(true);
78
79     this.worker.setSpeed(100);
80     this.worker.startThread();
81
82     Thread prev = (Thread) field.get(this.worker);
83
84     //when
85     this.worker.stopThread();
86
87     //then
88     assertTrue(((Thread) field.get(this.worker)).isInterrupted());
89     assertEquals(prev, field.get(this.worker));
}

```

Unable to save plugin settings: The plugin com.intellij... (5 minutes ago) 60:18 LF UTF-8 4 spaces develop

▲ [picture 165] test cases of Worker class (5)

stopAliveThreadTest는 실행 중인 스레드가 있을 때 stopThread를 호출하면 스레드가 잘 정지하는지 테스트한다.



Run: LifeGameApplicationTests WorkerTest

Test Results

Test Case	Time
[startThread/alive]	2 sec 29 ms
[startThread/not alive]	1sec 981ms
[stopThread/alive]	13 ms
[stopThread/not alive]	24 ms
[stopThread/not alive]	11ms

Tests passed: 4 (moments ago) 76:16 LF UTF-8 4 spaces develop

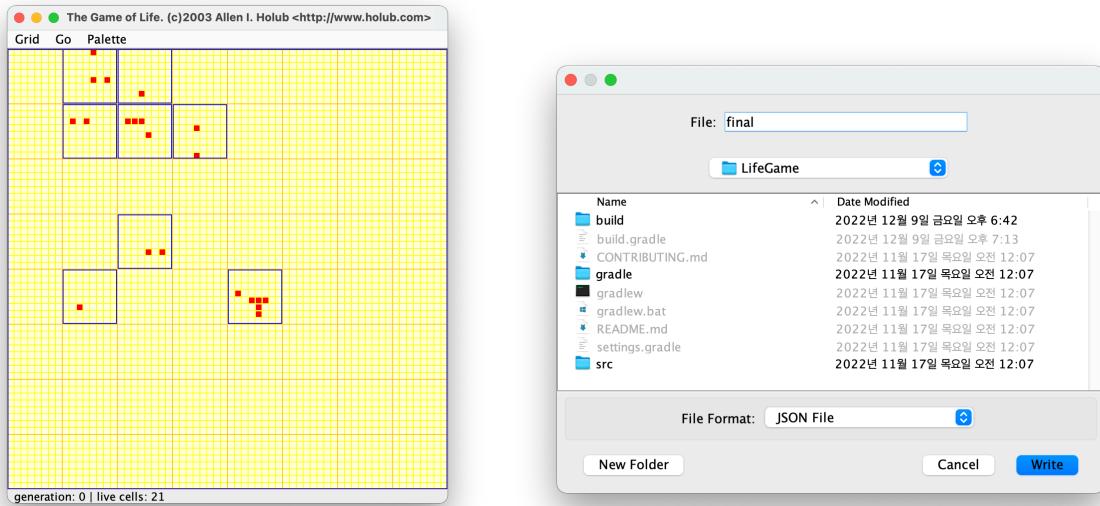
▲ [picture 166] WorkerTest passed

WorkerTest의 모든 Test가 통과하는 것을 확인할 수 있다.

5. 결과

5.1. 기존 Life Game 프로젝트 지원 기능

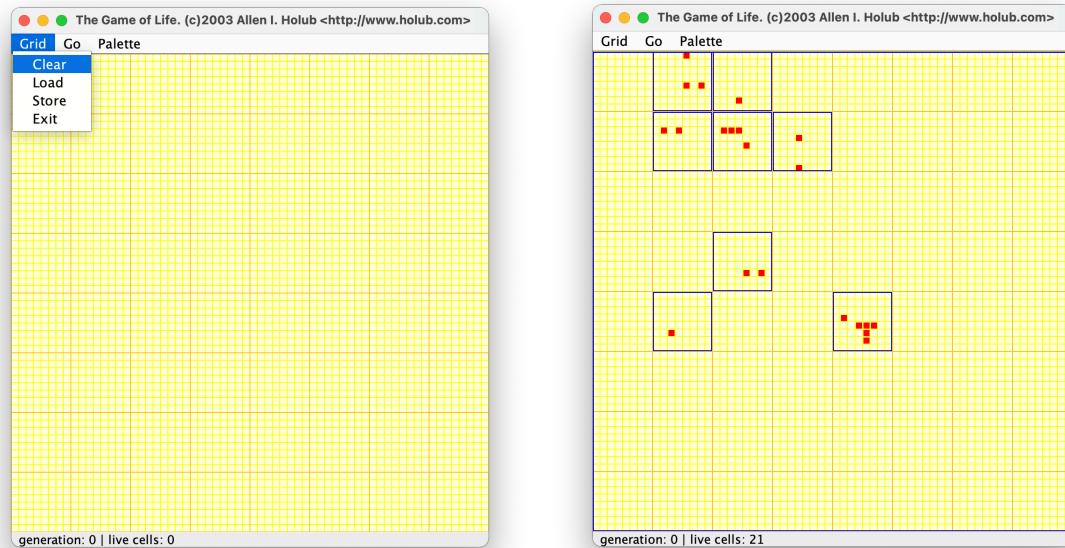
- Toggle , Store



▲ [picture 167] Toggle, Store 동작

빈 cell을 눌러 live 하거나 dead 상태로 toggle 하는 기능과 현재 상태 store 하는 기능이 정상적으로 동작한다.

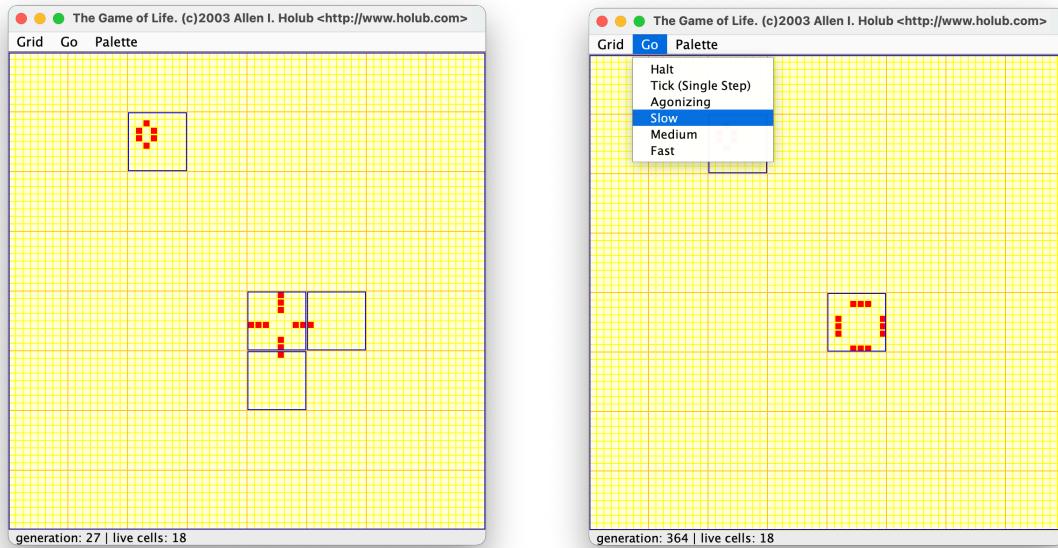
- Clear, Load



▲ [picture 168] Clear, Load 작동

Map을 한번에 지울 수 있는 clear 기능과 저장해둔 상태를 다시 불러오는 load 기능이 정상적으로 동작한다.

- Halt, Tick, Agonizing, Slow, Medium, Fast

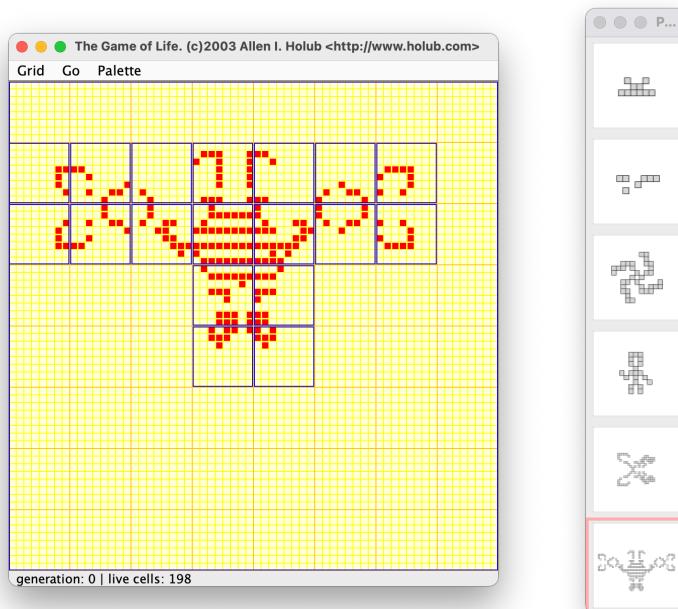


▲ [picture 169] Go menu 작동

진행을 멈추는 halt, 한 generation만 진행하는 tick, 속도 별로 진행하는 slow, medium, fast 기능도 모두 정상적으로 작동한다.

5.2. 확장 기능

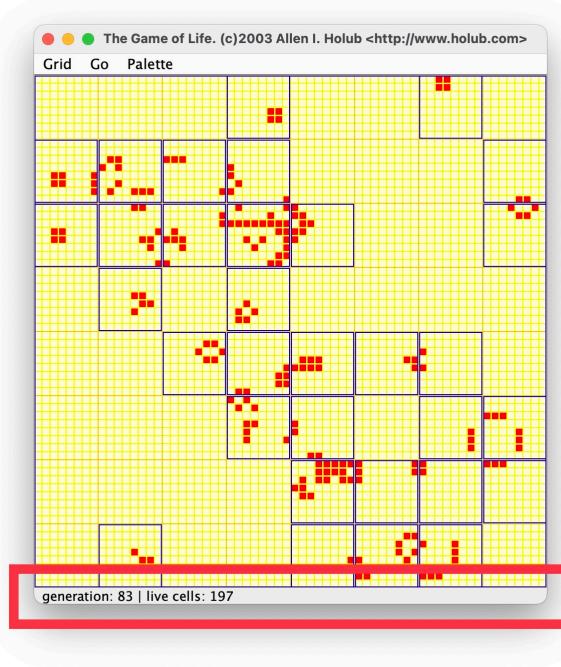
- Palette



▲ [picture 170] Palette 작동

메뉴바의 Palette의 show, hide를 통해 팔레트를 열고 닫는다. 팔레트에 저장되어 있는 흥미로운 패턴을 선택하여 원하는 위치에 붙일 수 있다.

- 현재 generation 수, live cells 개수



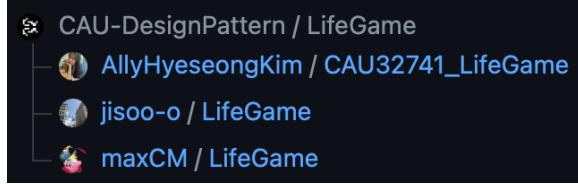
▲ [picture 171] Status bar 작동

Status bar에서 현재 진행 상태인 generation 수와 live cells 수의 변화를 확인할 수 있다.

6. GitHub

6.1. Repository URL

<https://github.com/CAU-DesignPattern/LifeGame>

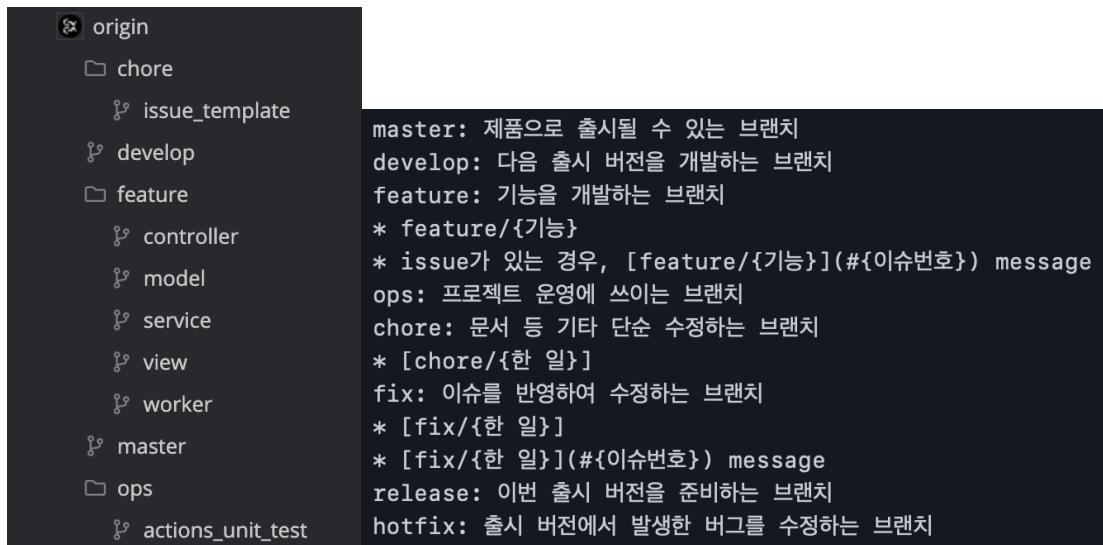


해당 레포에서 각자 개인 레포에 fork를 떠서 작업하였다.

6.2. Commit history

<https://github.com/CAU-DesignPattern/LifeGame/commits/master>

<https://github.com/CAU-DesignPattern/LifeGame/blob/master/CONTRIBUTING.md>



해당하는 각 브랜치에 {브랜치}#{이슈번호} 브랜치로 Pull Request를 날리는 방식으로 작업하였다.

workspace repository branch master

Viewing 13/13 Show All

LOCAL 4/4

- develop
- feature
- model
- service
- master**

REMOTE 9/9

- origin
- chore
- issue_template
- develop
- feature

PULL REQUESTS 0

Search pull requests

My Pull Requests 0

ISSUES

TEAMS

TAGS 0/0

SUBMODULES 0

GITHUB ACTIONS 1

BRANCH / TAG GRAPH COMMIT MESSAGE

Merge pull request #47 from CAU-DesignPattern/develop

Merge pull request #46 from CAU-DesignPattern/feature/view

[feature/view] Delete Observable

Merge pull request #45 from CAU-DesignPattern/develop

(develop) Modify model to count live cells

Merge pull request #44 from CAU-DesignPattern/feature/model

Merge branch 'develop' into feature/model

[feature/model] Create StatusLabel

Merge pull request #43 from maxCM/model#36

Merge branch 'feature/model' into model#36

Merge pull request #42 from CAU-DesignPattern/feature/service

Merge pull request #41 from CAU-DesignPattern/feature/view

Merge pull request #39 from AllyHyeseongKim/view#37

Merge pull request #38 from jisoo-o/service#3

[feature/service](#3) Modify Service

[feature/view](#37) Modify the unit test to remove @Spy

[feature/service](#3) Delete the debug point

[feature/service](#3) Remove declared library

[feature/model](#37) Create the unit test for views and controllers

[feature/service](#3) Create actionCatchTest

[feature/service](#3) Create service test

[feature/view](#37) Create the unit test for models

[feature/view](#37) Create the unit test for controllers

[feature/view](#37) Create the unit test for DrawBehavior

[feature/view](#37) Create the pattern image

commit: 9aedad

Merge pull request #47 from CAU-DesignPattern/develop

Merge 'develop' into master

AllyHyeseongKim parent: 652e96,8dc749
authored 12/10/2022 @ 1:51 AM

GitHub committed 12/10/2022 @ 1:51 AM

1 modified

Path Tree View all files

src/main/java/com/LifeGam... /MenuBar.java

workspace repository branch master

Viewing 13/13 Show All

LOCAL 4/4

- develop
- feature
- model
- service
- master**

REMOTE 9/9

- origin
- chore
- issue_template
- develop
- feature

PULL REQUESTS 0

Search pull requests

My Pull Requests 0

ISSUES

TEAMS

TAGS 0/0

SUBMODULES 0

GITHUB ACTIONS 1

BRANCH / TAG GRAPH COMMIT MESSAGE

[feature/view](#37) Create the palette view

[feature/service](#3) Create load, store function

[feature/service](#3) Add Read Fail Alert

[feature/model](#36) next state & toggle live cell 개선

[feature/model](#36) next state&I generation 1일에트되도록 개선

Merge pull request #35 from CAU-DesignPattern/feature/worker

Merge pull request #34 from AllyHyeseongKim/worker#10

[feature/worker](#10) Create the unit test for actions: Agonizing, Fast, Halt, Medium, Slow

[feature/worker](#10) Create the multi-threading actions: Halt, Agonizing, Slow, Medium

[feature/worker](#10) Create the multi-threading worker

Merge pull request #33 from CAU-DesignPattern/feature/model

Merge pull request #32 from maxCM/model#29

[feature/model] Modify the display name of test

[feature/model] mapchanged 호출 확인

[feature/model] remove map parameter(tick action), change test

commit: 2d8e1c

[feature/view](#37) Create the pattern image

src/main/resources/patterns

AllyHyeseongKim parent: c4b827
authored 12/8/2022 @ 2:17 AM

6 added

Path Tree View all files

src/main/resources/patterns/1.png

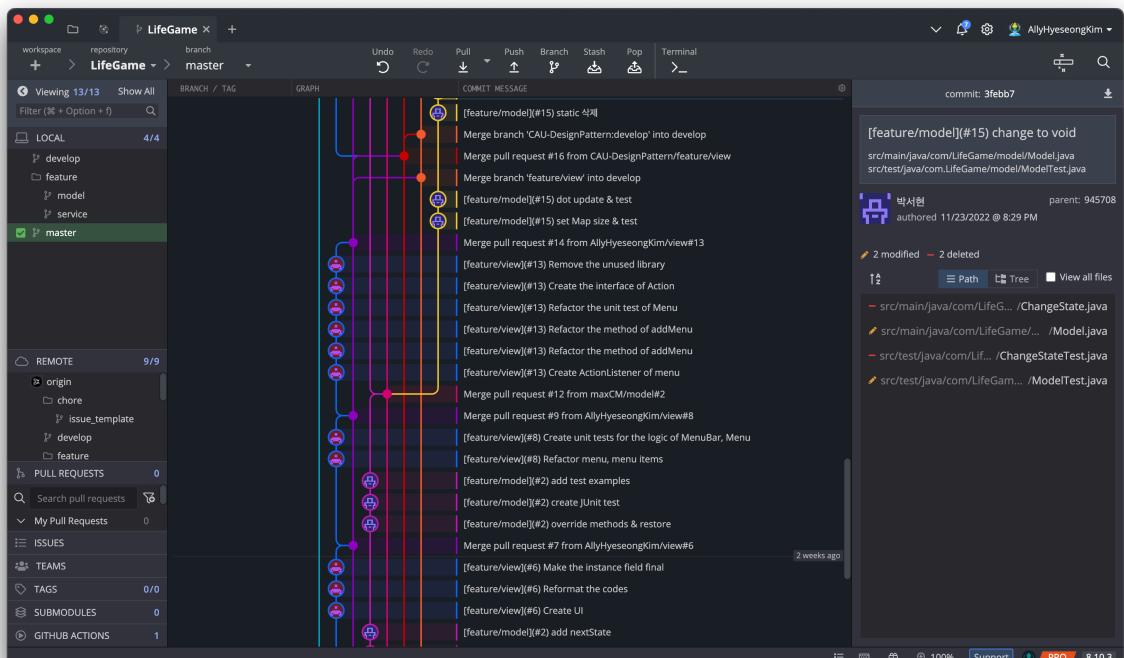
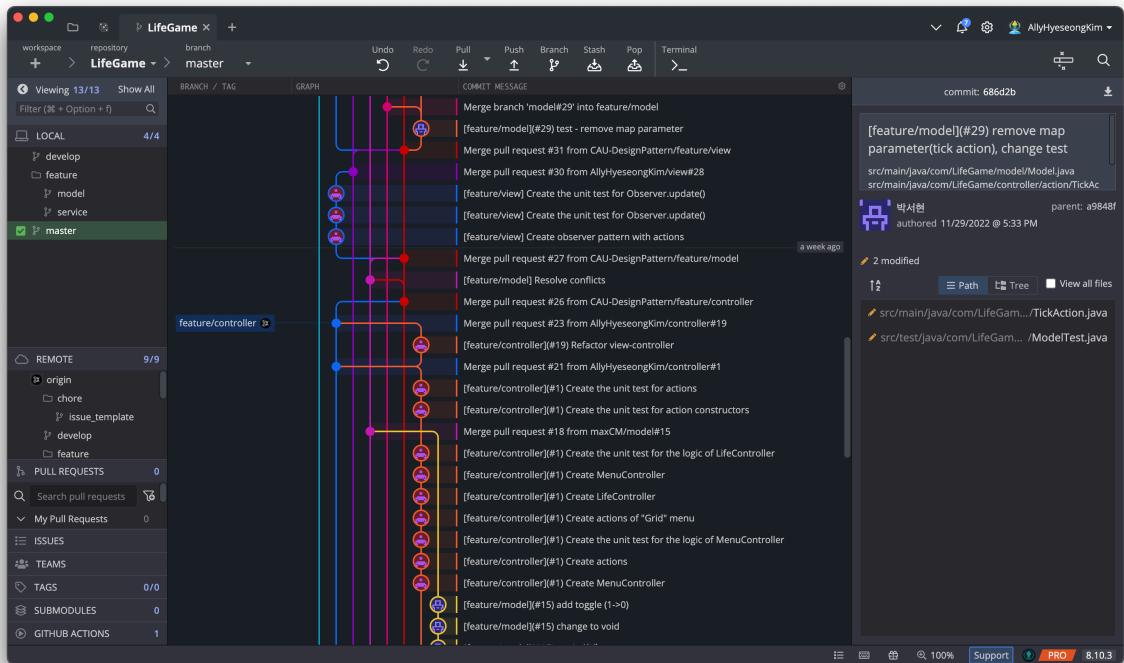
src/main/resources/patterns/2.png

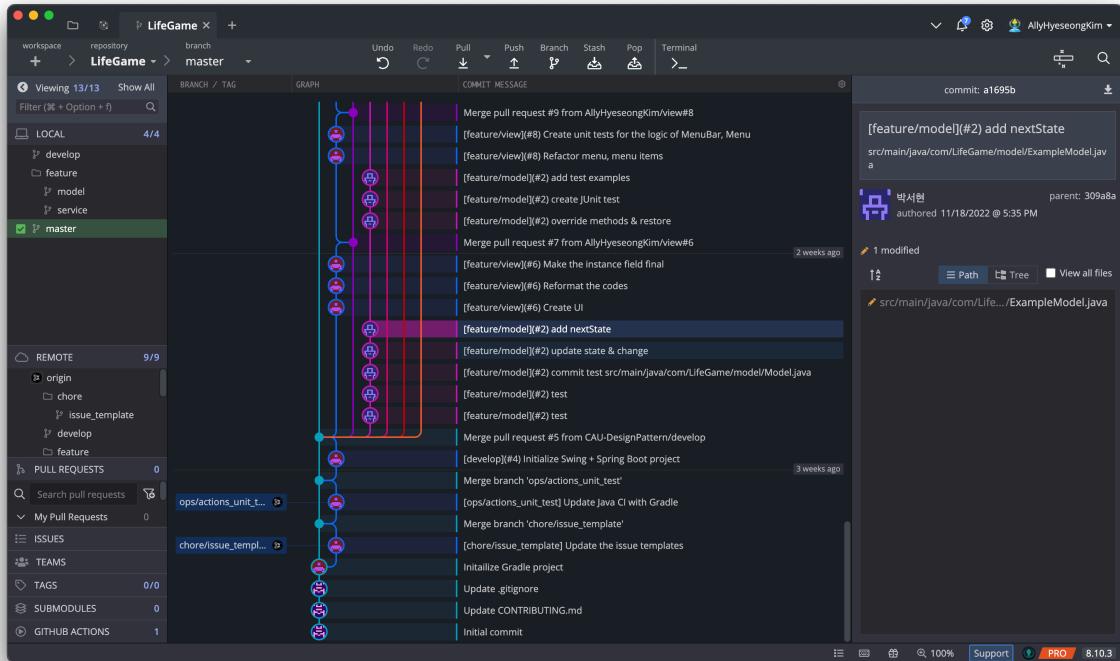
src/main/resources/patterns/3.png

src/main/resources/patterns/4.png

src/main/resources/patterns/5.png

src/main/resources/patterns/6.png





▲ [picture 172] Commit history

6.3. Issues (역 할분배)

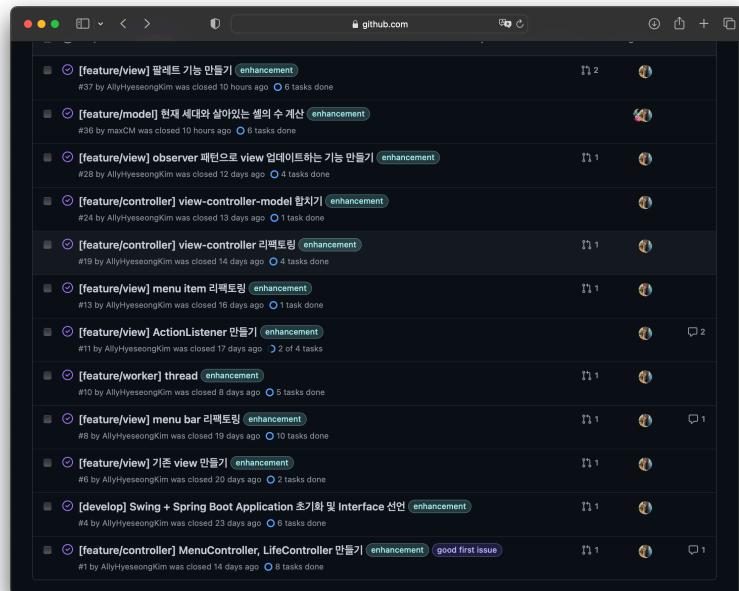
https://github.com/CAU-DesignPattern/LifeGame/tree/master/.github/ISSUE_TEMPLATE

<https://github.com/CAU-DesignPattern/LifeGame/issues?q=is:issue+is:closed>

Issue template을 만들어서 모든 작업은 issue로 등록하여 issue 단위로 각 역할을 분배하여 작업하였다.

6.3.1. 김혜성 작업 issue

<https://github.com/CAU-DesignPattern/LifeGame/issues?q=is:issue+is:closed+assignee:AllyHyeseongKim>

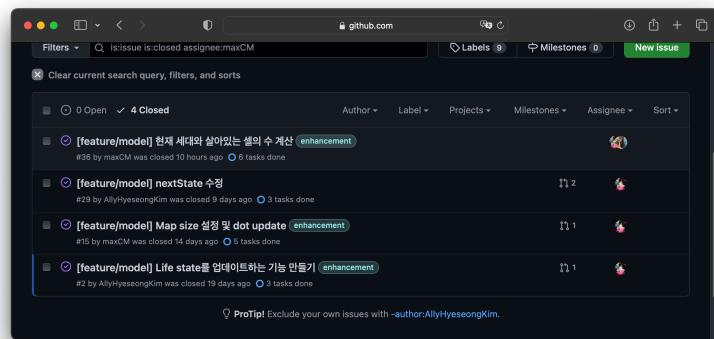


▲ [picture 173] 김혜성 작업 issue

- 기존 기능 view, controller 개발
- 기존 기능 worker (thread를 사용하여 state 업데이트) 개발
- Observer pattern 적용하여 view 업데이트
- Spring 적용
- 확장 기능 Palette(model, view, controller) 개발
- 확장 기능 현재 세대, 셀 수 계산 개발

6.3.2. 박서현 작업 issue

<https://github.com/CAU-DesignPattern/LifeGame/issues?q=is:issue+is:closed+assignee:maxCM>

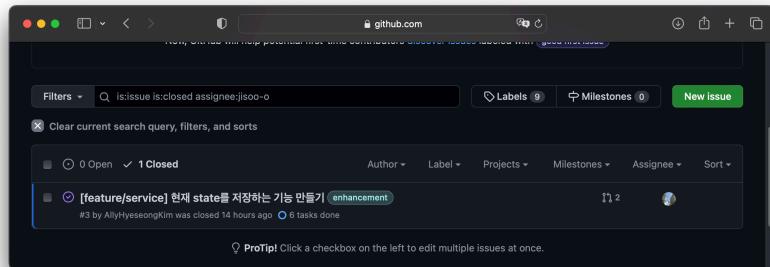


▲ [picture 174] 박서현 작업 issue

- 기존 기능 model 개발
- 확장 기능 현재 세대, 셀 수 계산 개발

6.3.3. 박지수 작업 issue

<https://github.com/CAU-DesignPattern/LifeGame/issues?q=is:issue+is:closed+assignee:jisoo-o>



▲ [picture 175] 박지수 작업 issue

- 기존 기능 service 개발

6.4. Actions

<https://github.com/CAU-DesignPattern/LifeGame/blob/master/.github/workflows/gradle.yml>

<https://github.com/CAU-DesignPattern/LifeGame/actions>

Java gradle unit test CI를 만들어서 test를 통과하지 않으면 Pull Request를 merge하지 않도록 하였다.

The screenshot shows a code editor window with a dark theme. The file being edited is `gradle.yml`, which contains a GitHub Actions workflow for a Java CI project. The code is as follows:

```
name: Java CI with Gradle

on:
  push:
    branches: [ "*" ]
  pull_request:
    branches: [ "*" ]

jobs:
  unit_test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'corretto'

      - name: Build with Gradle
        uses: gradle/gradle-build-action@67421db6bd0bf253fb4bd25b31ebb98943c375e1
        with:
          arguments: build

      - name: Grant execute permission for gradlew
        run: chmod +x gradlew

      - name: Test with Gradle
        run: ./gradlew --info test
```

▲ [picture 176] Java CI with Gradle