Java : The First Toy Project

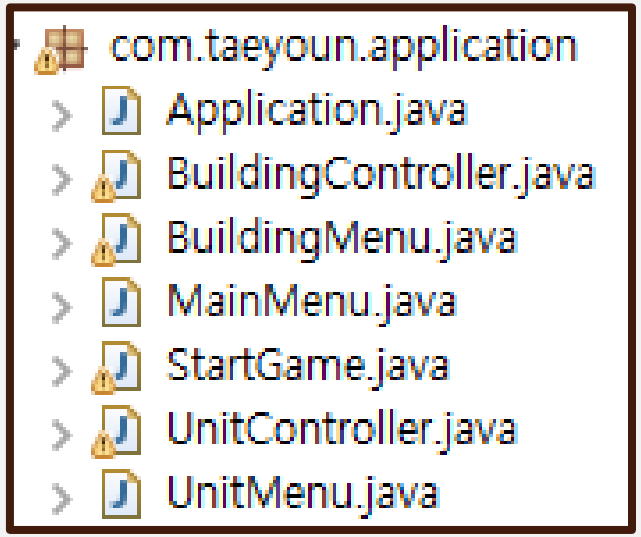
STARCRAFT

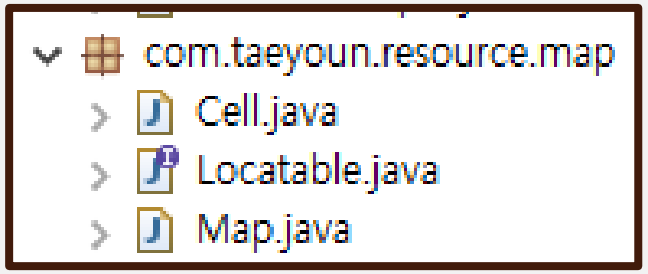
- 작성자:김태연
- 작성일시:2024.01.26
- xodus4194@naver.com

[INDEX]


250
0
11/26
0
00:01
메뉴

1. 시스템





I. StartGame

II. 건물 & 유닛 & 자원 & 명령 관리

III. 맵

1. System

시스템

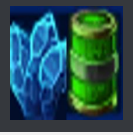



◆ 1. 게임 시작 : new StartGame();

```
public class Application {
    public static void main(String[] args) {
        StartGame game = new StartGame();
        game.showGameStatus();              //게임 끝나면 실행함
    }
}
```

1. System 시스템



◆ 1. 게임 시작 : StartGame 생성자

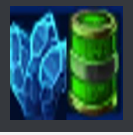
```
public StartGame() {
    this.map = new Map();
    this.human = new Player("김태연");
    this.computer = new Player("Computer");

    CommandCenter commandCenter1 = new CommandCenter(human,0,0);
    Unit scv1 = new Scv(human, 1, 0);
    Unit scv2 = new Scv(human, 2, 0);
    Unit scv3 = new Scv(human, 3, 0);
    Unit scv4 = new Scv(human, 4, 0);
    human.addUnit(scv1);
    human.addUnit(scv2);
    human.addUnit(scv3);
    human.addUnit(scv4);
    CommandCenter commandCenter2 = new CommandCenter(computer,10,10);
    Unit cscv1 = new Scv(computer, 11, 10);
    Unit cscv2 = new Scv(computer, 12, 10);
    Unit cscv3 = new Scv(computer, 13, 10);
    Unit cscv4 = new Scv(computer, 14, 10);
    computer.addUnit(cscv1);
    computer.addUnit(cscv2);
    computer.addUnit(cscv3);
    computer.addUnit(cscv4);
```

I. 플레이어 지정
II. 기본 건물 생성
III. 기본 유닛 생성

1. System

시스템



◆ 1. 게임 시작 : StartGame 생성자

```
System.out.println("---------------------- 게임 시작합니다 -------------------");
System.out.println();
System.out.println();
System.out.println();
human.showGameStatus();
try {
    Thread.sleep(10);
} catch (InterruptedException e) {
    e.printStackTrace();
}

        System.out.println("*****************************************");
        System.out.println("*                                       *");
        System.out.println("*                                       *");
        System.out.println("*                                       *");
        System.out.println("*                                       *");
        System.out.println("*                                       *");
        System.out.println("*                                       *");
        System.out.println("*              Game Start               *");
        System.out.println("*                                       *");
        System.out.println("*****************************************");
        Scanner scanner = new Scanner(System.in);
        MainMenu mainMenu = MainMenu.getInstance();
        mainMenu.showMenu(human, computer, scanner);     //메인메뉴 실행
```

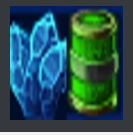
I. showGameStatus(); 실행

II. 플레이어가 보유한 자원, 건물, 유닛에 대한 정보 출력

III. 패널 화면 출력

1. System 시스템



◆ 1. 게임 시작 : StartGame 생성자 출력화면

```
========================================== Current Game Status ==========================================
Player: 김태연
Minerals: 50
Gas: 0
Population: 10
Buildings:
   Unit<1> : TERRAN SCV   체력 : 60/60    공격력 : 5 방어력 : 1     상태 : staying   위치 : (1,0)
   Unit<2> : TERRAN SCV   체력 : 60/60    공격력 : 5 방어력 : 1     상태 : staying   위치 : (4,3)
   Unit<3> : TERRAN SCV   체력 : 60/60    공격력 : 5 방어력 : 1     상태 : staying   위치 : (8,0)
   Unit<4> : TERRAN SCV   체력 : 60/60    공격력 : 5 방어력 : 1     상태 : staying   위치 : (4,0)
Units:
   Building<1> : TERRAN Command Center  상태 : staying

************************************************************************************************************************
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
*                                                Game Start                                                            *
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
*                                                                                                                      *
************************************************************************************************************************
------------------------------------------------- 패널 화면 -------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
```

I. showGameStatus(); 실행

II. 플레이어가 보유한 자원, 건물, 유닛에 대한 정보 출력

III. 패널 화면 출력

1. System 시스템



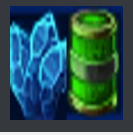

◆ 2. 자원 표시 : showGameStatus()

```
========================================== Current Game Status ==========================================
Player: 김태연
Minerals: 50
Gas: 0
Population: 10
Buildings:
   Unit<1> : TERRAN SCV    체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying    위치 : (1,0)
   Unit<2> : TERRAN SCV    체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying    위치 : (2,3)
   Unit<3> : TERRAN SCV    체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying    위치 : (3,0)
   Unit<4> : TERRAN SCV    체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying    위치 : (4,0)
   Unit<5> : TERRAN Marine         체력 : 40/40     공격력 : 6 방어력 : 1      상태 : staying    위치 : (5,0)
   Unit<6> : TERRAN Marine         체력 : 40/40     공격력 : 6 방어력 : 1      상태 : staying    위치 : (30,20)
Units:
   Building<1> : TERRAN Command Center  상태 : staying

============================================================================================================
```

```java
public void showGameStatus() {
    // 게임 상태를 보여주는 창 만들기
    System.out.println("========================================== Current Game Status ==========================================");
    System.out.println("Player: " + playerName);
    System.out.println("Minerals: " + minerals);
    System.out.println("Gas: " + gas);
    System.out.println("Population: " + population);
    showUnitList();
    showBuildingList();
    System.out.println();
    System.out.println("============================================================================================================");
}
```

1. System 시스템



◆ 2. 건물 & 유닛 표시 : showBuildingList() & showUnitList()

```
public void showGameStatus() {
    // 게임 상태를 보여주는 창 만들기
    System.out.println("========================================= Current Game Status =========================================");
    System.out.println("Player: " + playerName);
    System.out.println("Minerals: " + minerals);
    System.out.println("Gas: " + gas);
    System.out.println("Population: " + population);
    showUnitList();
    showBuildingList();
    System.out.println();
    System.out.println("=======================================================================================================");
}
```

```
public void showBuildingList() {
    System.out.println("Buildings:");
    for (Building building : buildingList) {
        System.out.printf("    " + building);
        System.out.printf("\t" +  "상태 : " + building.getState() + "\n");
    }
}
public void showUnitList() {
    System.out.println("Units:");
    for (Unit unit : unitList) {
        System.out.printf("    " + unit + "\n");
    }
}
```

1. System 시스템




◆ 2. 건물 관리 : BuildingController

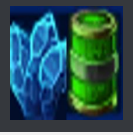
```java
public class BuildingController {

    static int buildingIndex;
    public static void controlBuilding(Player human, Scanner scanner, Class<? extends Building> BuildingType) {
        List<? extends Building> buildingList = human.getListOfType(BuildingType);
        while (true) {
            if (buildingList.size() == 0) {
                System.out.println("건물이 없습니다.");
                break;
            }
            else
                buildingIndex = 1;
            for (Building building : buildingList) {
                System.out.printf("\t입력번호<%d> : %s\n", buildingIndex++, building);
            }
                System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");

            int BuildingNumber = getBuildingNumber(scanner);        // 입력값을 -2,-1,n 셋중에 하나로 반환해주는 메서드

            if (BuildingNumber == -2) {
                break;
            }
            else if (BuildingNumber == -1) {    //for 반복문 사용해보자
                // 건물 전체 선택 시
                System.out.println("다중 건물 명령을 선택했습니다.");

                for (Building Building : buildingList) {            // 해당 종류의 건물 모두에 대해 반복문 실행
                    handleBuildingMenu(Building, scanner);
                }
            }
            else {
                // 개별 건물 선택 시
                System.out.println(buildingList.get(BuildingNumber).getName() + "을(를) 선택했습니다.");
                handleBuildingMenu(buildingList.get(BuildingNumber), scanner);
            }
        }
    }
```

I. 건물 종류 확인
II. 개별 명령
III. 단체 명령

1. System

시스템



◆ 2. 유닛 관리 : UnitController

```java
public class UnitController {

    static int unitIndex;
    public static void controlUnit(Player human, Scanner scanner, Class<? extends Unit> unitType) {
        List<? extends Unit> unitList = human.getListOfType(unitType);
        while (true) {
            if (unitList.size() == 0) {
                System.out.println("유닛이 없습니다.");
                break;
            }
            else
                unitIndex = 1;
                for (Unit unit : unitList) {
                    System.out.printf("\t입력번호<%d> : %s\n", unitIndex++, unit);
                }
                System.out.println(" 사용할 유닛 번호를 고르세요 , 0 : 전체 선택, q : 종료 ");
                int unitNumber = getUnitNumber(scanner);
                if (unitNumber == -2) {
                    break;
                }
                else if (unitNumber == -1) {
                    boolean allScv = unitList.stream().allMatch(unit -> unit instanceof Scv);
                    if (allScv) {
                        MultiCommandScv((List<Scv>) unitList,scanner);
                    } else {
                        System.out.println("다중 명령을 선택했습니다.");
                        MultiCommand((List<Unit>) unitList,scanner);
                    }
                }

                else {
                    // 개별 유닛 선택 시
                    System.out.println(unitList.get(unitNumber).getName() + "을(를) 선택했습니다.");


                    handleUnitMenu(unitList.get(unitNumber), scanner);
            }
        }
    }
```

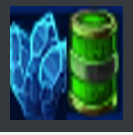
I. 유닛 종류 확인

II. 개별 명령

III. 단체 명령

1. System 시스템



◆ 3. 맵 관리 : class Map & class Cell

```java
public class Map {
    private final static int WIDTH = 100; // 맵의 가로 길이는 100 으로 고정
    private final static int HEIGHT = 100; // 맵의 세로 길이는 100 으로 고정
    private static Cell[][] grid; // 2차원 Cell type 배열 grid 생성

    public Map() { // Map의 생성자 , 파라미터가 없음
        initializeMap();
    }

    private void initializeMap() { // map 에 좌표 번호 지정하기
        grid = new Cell[WIDTH][HEIGHT];
        for (int x = 0; x < WIDTH; x++) {
            for (int y = 0; y < HEIGHT; y++) {
                grid[x][y] = new Cell(x, y);
            }
        }
    }

    public static Cell getCell(int x, int y) {
        if (isValidCoordinate(x, y)) {
            return grid[x][y];
        } else {
            return null;
        }
    }

    private static boolean isValidCoordinate(int x, int y) {
        return x >= 0 && x < WIDTH && y >= 0 && y < HEIGHT;
    }
}
```

```java
public class Cell {
    //좌표칸 만들기 ( 맵의 기본 구성요소 )
    private int x;
    private int y;
    private Unit unit;

    public Cell(int x, int y) {
        super();
        this.x = x;
        this.y = y;
        this.unit = null;
    }

    public boolean isUnit() {
        if (this.unit == null) {
            return false;
        } else {
            return true;
        }
    }
```

I. Cell type 배열

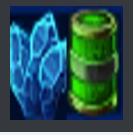
II. 유효값 확인

III. 맵의 구성요소

IV. 유닛 유무 반환

1. System 시스템



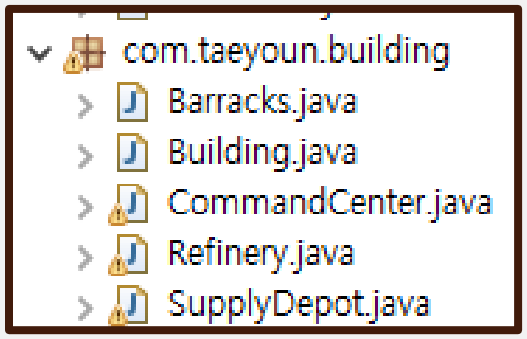
◆ 4. 패널 표시 : MainMenu.showMenu(Player, Player, Scanner);

```
---------------------------------------------------- 패널 화면 ----------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
```

```
while (true) {
    System.out.println("---------------------------------------------------- 패널 화면 ----------------------------
    System.out.println("패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료");
    input = scanner.next();
    try {
        switch (input.toLowerCase()) {
        case "b" :
            buildingMenu.showMenu(human, scanner);
            break;
        case "u" :
            unitMenu.showMenu(human, scanner);
            break;

        case "s" :
            human.showGameStatus();
            if (isGameOver()) {
                endGame();
                return;
            }

            break;
```

```
        case "q":
            System.out.println("정말 게임을 종료하시겠습니까?");
            System.out.println();
            System.out.println();
            System.out.println(" y : 네, n : 아니오");
            Thread.sleep(1500);
            input = scanner.next().toLowerCase();
            if (input.equals("n")) {
            System.out.println("패널 화면으로 돌아갑니다");
            break;
            }
            else if (input.equals("y")) {
            System.out.println("게임을 종료합니다");
            scanner.close();
            return;
            }

        default:
            System.out.println("메인 메뉴 오류1번");
            break;

        } // switch 문 종료
    } catch (Exception e) {
        System.out.println("메인 메뉴 오류2" + e.getMessage());
```

2. 건물

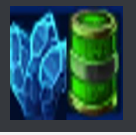


I. 건물 종류 선택
II. 건물 번호 선택
III. 건물 기능 사용

2. Building

시스템



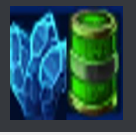
◆ 1. 메뉴 선택 -> 2. 종류 선택 -> 3. 번호 선택 -> 4. 기능 선택 -> 5. 종료

```
---------------------------------------------------- 패널 화면 ----------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
b
건물 창을 보여줍니다
Units:
   Building<1> : TERRAN Command Center  상태 : staying
조작할 건물을 고르세요 (c : Command Center, b : Barracks, s : Supply depot, r : Refinery, q : 종료):
c
        입력번호<1> : Building<1> : TERRAN Command Center
 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료
1
TERRAN Command Center을(를) 선택했습니다.
건물 명령 콘솔입니다. 명령을 선택하세요.
커멘드센터 메뉴입니다. s: SCV 생산, q: 종료
s
SCV 생산 중...
        입력번호<1> : Building<1> : TERRAN Command Center
 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료
SCV GOOD TO GO SIR
내 x좌표는 : (2,0)
unitList에 TERRAN SCV 를(을) 추가 했습니다
q
종료합니다.
```

2. Building

시스템



◆ 1. 메뉴 선택 : BuildingMenu.showMenu(Player, Scanner);

```
---------------------------------------------------- 패널 화면 ----------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
b
건물 창을 보여줍니다
Units:
   Building<1> : TERRAN Command Center  상태 : staying
조작할 건물을 고르세요 (c : Command Center, b : Barracks, s : Supply depot, r : Refinery, q : 종료):
```

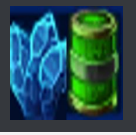
```java
    //static 해제
    public void showMenu(Player human, Scanner scanner) {
        System.out.println("건물 창을 보여줍니다");
        human.showBuildingList();
        System.out.println("조작할 건물을 고르세요 (c : Command Center, b : Barracks, s : Supply depot, r : Refinery, q : 종료):");
        String input = scanner.next().toLowerCase(); // 메뉴판 인쇄랑 input은 세트

        switch (input) {
//      case "c":
//          CommandCenter.controlCommandCenter(human, scanner);
//          break;
//
//      case "b":
//          Barracks.controlBarracks(human, scanner);
//          break;
//
//      case "s":
//          SupplyDepot.controlSupplyDepot ( human, scanner);
//          break;
//
//      case "q":
//          System.out.println("유닛 조작을 종료합니다.");
//          break;
//
//      default:
//          System.out.println("잘못된 입력입니다");
//          break;
```

각 클래스에서 메서드를 끌어오는 형태에서 개선시켰다.

2. Building

시스템



◆ 2-1. 종류 선택 : controlBuilding(Player, Scanner, 건물종류);

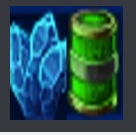
```
        case "c":
            BuildingController.controlBuilding(human, scanner, CommandCenter.class);
            break;

        case "b":
            BuildingController.controlBuilding(human, scanner, Barracks.class);
            break;

        case "s":
            BuildingController.controlBuilding(human, scanner, SupplyDepot.class);
            break;

        case "r":
            BuildingController.controlBuilding(human, scanner, Refinery.class);
            break;

        case "q":
            System.out.println("건물 조작을 종료합니다.");
            break;

        default:
            System.out.println("잘못된 입력입니다");
            break;
        }
    }
```

Building.class 는 해당 건물이 가지는 클래스를 나타낸다.

2. Building

시스템



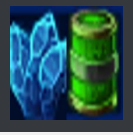
◆ 2-1. 종류 선택 : controlBuilding(Player, Scanner, Building.class);

```
static int buildingIndex;
public static void controlBuilding(Player human, Scanner scanner, Class<? extends Building> BuildingType) {
    List<? extends Building> buildingList = human.getListOfType(BuildingType);
    while (true) {
        if (buildingList.size() == 0) {
            System.out.println("건물이 없습니다.");
            break;
        }
        else
            buildingIndex = 1;
        for (Building building : buildingList) {
            System.out.printf("\t입력번호<%d> : %s\n", buildingIndex++, building);
        }
            System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");

        int BuildingNumber = getBuildingNumber(scanner);        // 입력값을 -2,-1,n 셋중에 하나로 반환해주는 메서드

        if (BuildingNumber == -2) {
            break;
        }
        else if (BuildingNumber == -1) {    //for 반복문 사용해보자
            // 건물 전체 선택 시
            System.out.println("다중 건물 명령을 선택했습니다.");

            for (Building Building : buildingList) {            // 해당 종류의 건물 모두에 대해 반복문 실행
                handleBuildingMenu(Building, scanner);
            }
        }
        else {
            // 개별 건물 선택 시
            System.out.println(buildingList.get(BuildingNumber).getName() + "을(를) 선택했습니다.");
            handleBuildingMenu(buildingList.get(BuildingNumber), scanner);
        }
    }
}
```

2. Building

시스템



◆ 2-2. 종류 선택 : BuildingController

```
public class BuildingController {

    static int buildingIndex;
    public static void controlBuilding(Player human, Scanner scanner, Class<? extends Building> BuildingType) {
        List<? extends Building> buildingList = human.getListOfType(BuildingType);
        while (true) {
            if (buildingList.size() == 0) {
                System.out.println("건물이 없습니다.");
                break;
            }
            else
                buildingIndex = 1;
            for (Building building : buildingList) {
                System.out.printf("\t입력번호<%d> : %s\n", buildingIndex++, building);
            }
                System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");
```

```
            System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");

        int BuildingNumber = getBuildingNumber(scanner);        // 입력값을 -2,-1,n 셋중에 하나로 반환해주는 메서드

        if (BuildingNumber == -2) {
            break;
        }
        else if (BuildingNumber == -1) {    //for 반복문 사용해보자
            // 건물 전체 선택 시
            System.out.println("다중 건물 명령을 선택했습니다.");

            for (Building Building : buildingList) {            // 해당 종류의 건물 모두에 대해 반복문 실행
                handleBuildingMenu(Building, scanner);
            }
        }
        else {
            // 개별 건물 선택 시
            System.out.println(buildingList.get(BuildingNumber).getName() + "을(를) 선택했습니다.");
            handleBuildingMenu(buildingList.get(BuildingNumber), scanner);
        }
    }
}
```

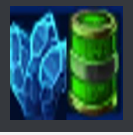
건물의 통제권을 가지는 클래스

If 문을 사용해 입력값에 따라 다른 결과를 받아온다.

2. Building

시스템

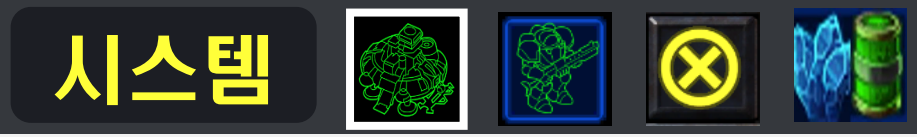


◆ 2-2. 종류 선택 : BuildingController

```
            System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");

        int BuildingNumber = getBuildingNumber(scanner);        // 입력값을 -2,-1,n 셋중에 하나로 반환해주는 메서드

        if (BuildingNumber == -2) {
            break;
        }
        else if (BuildingNumber == -1) {    //for 반복문 사용해보자
            // 건물 전체 선택 시
            System.out.println("다중 건물 명령을 선택했습니다.");

            for (Building Building : buildingList) {            // 해당 종류의 건물 모두에 대해 반복문 실행
                handleBuildingMenu(Building, scanner);
            }
        }
        else {
            // 개별 건물 선택 시
            System.out.println(buildingList.get(BuildingNumber).getName() + "을(를) 선택했습니다.");
            handleBuildingMenu(buildingList.get(BuildingNumber), scanner);
        }
    }
}
```

2. Building



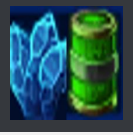
◆ 3. 번호 선택 : getBuildingNumber(Scanner)

```
private static int getBuildingNumber(Scanner scanner) {
    while (true) {
        if (scanner.hasNext()) {
            String input = scanner.next().toLowerCase();

            if (input.equals("q")) {
                System.out.println("종료합니다.");
                return -2; // -2를 반환하여 호출한 곳에서 종료 여부를 확인할 수 있도록 함
            }
            try {
                int parsedInput = Integer.parseInt(input);
                if (parsedInput >= 0) {      //입력값이 0이거나 n일 경우
                    return parsedInput - 1;
                } else {
                    System.out.println("잘못된 건물 번호입니다. 다시 선택하세요.");
                }
            } catch (NumberFormatException e) {
                System.out.println("올바른 숫자를 입력하세요.");
            }
        } else {
            System.out.println("입력값이 없습니다. 다시 입력하세요.");
            scanner.nextLine();
        }
    }
}
```

2. Building

시스템



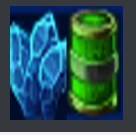
◆ 3-1. 번호 선택 : 배럭

```
건물 창을 보여줍니다
Buildings:
   Building<1> : TERRAN Command Center  상태 : staying
   Building<2> : TERRAN Supply Depot    상태 : staying
   Building<3> : TERRAN Supply Depot    상태 : staying
   Building<4> : TERRAN Refinery        상태 : staying
   Building<5> : TERRAN Barracks        상태 : staying
조작할 건물을 고르세요 (c : Command Center, b : Barracks, s : Supply depot, r : Refinery, q : 종료):
b
        입력번호<1> : Building<5> : TERRAN Barracks
 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료
```

```
    buildingIndex = 1;
for (Building building : buildingList) {
    System.out.printf("\t입력번호<%d> : %s\n", buildingIndex++, building);
}
    System.out.println(" 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료 ");

int BuildingNumber = getBuildingNumber(scanner);        // 입력값을 -2,-1,n 셋중에 하나로 반환해주는 메서드
```

2. Building 시스템



◆ 3-2. 번호 선택 : handleBuildingMenu(Building, scanner)

```
 사용할 건물 번호를 고르세요 , 0 : 전체 선택, 1 : 1번 건물, 2 : 2번 건물 , ... , q : 종료
1
TERRAN Barracks을(를) 선택했습니다.
건물 명령 콘솔입니다. 명령을 선택하세요.
배럭 메뉴입니다. m : 마린 생산, c : 메딕 생산, q: 종료
```

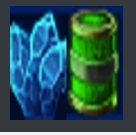
```
else {
    // 개별 건물 선택 시
    System.out.println(buildingList.get(BuildingNumber).getName() + "을(를) 선택했습니다.");
    handleBuildingMenu(buildingList.get(BuildingNumber), scanner);
}
```

```
 private static void handleBuildingMenu(Building building, Scanner scanner) {

     System.out.println("건물 명령 콘솔입니다. 명령을 선택하세요.");
     if(building instanceof CommandCenter) {
         CommandCenter commandCenter = (CommandCenter) building;
         CommandCommandCenter(commandCenter,scanner);
     }
     else if (building instanceof Barracks) {
         Barracks barracks = (Barracks) building;
         CommandBarracks(barracks,scanner);
     }
     else
         Command(building, scanner);

 }
```

2. Building

시스템



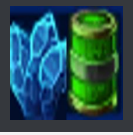
◆ 4. 기능 선택 : Command(Building, Scanner)

```
92
93      private static void Command(Building building, Scanner scanner) {
94          System.out.println("건물 명령 콘솔입니다. m: 내 상태 보여주기 , q: 종료");
95          String input = scanner.next().toLowerCase();
96
97          switch (input) {
98              case "m" :
99                  System.out.println(building.toString());
100             case "q":
101                 System.out.println("되돌아갑니다.");
102                 break;
103
104             default:
105                 System.out.println("에러코드 105");
106                 break;
107         }
108     }
```

2. Building

시스템



◆ 4. 기능 선택 : CommandBarracks(Building, Scanner)

```
130    private static void CommandBarracks(Barracks barracks, Scanner scanner) {
131        System.out.println("배럭 메뉴입니다. m : 마린 생산, c : 메딕 생산, q: 종료");
132        String input = scanner.next().toLowerCase();
133
134        switch (input) {
135        case "m":
136            barracks.produceMarine();
137            break;
138        case "c":
139            barracks.produceMedic();
140            break;
141        case "q":
142            System.out.println("되돌아갑니다.");
143            break;
144
145        default:
146            System.out.println("에러코드 146");
147            break;
148
149        }
150    }
151
152 }
```

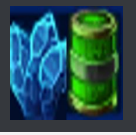
I. Command*

II. 배럭은 CommandBarracks

2. Building

시스템



◆ 4-1. 기능 선택 : produceMarine();

```
public void produceMarine() {
    if ( barracksCount > 0 ) {
        if (getProduceHelper().validCheck(player, Data.MARINE_M, Data.MARINE_G, Data.MARINE_P)) {
        System.out.println("Marine 생산 중...");
            Timer timer = new Timer();
            timer.schedule(new TimerTask() {
                @Override
                public void run() {
                    Unit marine = new Marine(player,x,y);
                    player.addUnit(marine);
                timer.cancel();
                }
            }, Data.MARINE_TIME * 100);
        }
    }
}
```

validCheck -> Timer -> new Marine

2. Building 시스템

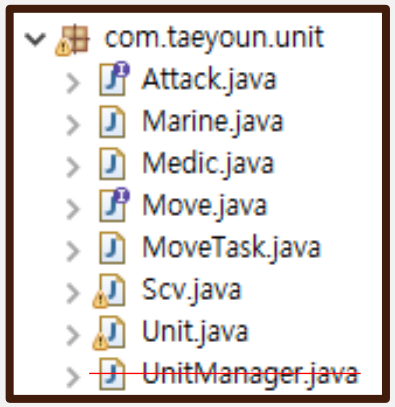
◆ 4-2. 생산 보조 : ProduceHelper.validCheck(Player, int, int, int)

```
    public boolean validCheck(Player player, int CostM, int CostG, int CostP) {
        if (!isPopulation(player, CostP)) {
            System.out.println("서플라이 디팟을 더 지으세요");
            return false;
        }
        if (!isMineral(player, CostM)) {
            System.out.println("미네랄이 부족합니다");
            return false;
        }
        if (!isGas(player, CostG)) {
            System.out.println("가스가 부족합니다");
            return false;
        } else {
            // System.out.println("valid check 성공"); //이 메서드가 호출되는지 확인용
            player.setMinerals(player.getMinerals() - CostM);
            player.setGas(player.getGas() - CostG);
            player.setPopulation(player.getPopulation() - CostP);

            return true;
        }
    }
}
```


```
public class ProduceHelper {

    // 싱글톤

    private static ProduceHelper produceHelper;
    public static ProduceHelper getInstance() {
        if (produceHelper == null) {
            produceHelper = new ProduceHelper();
        }
        return produceHelper;
    }

    public static void freeInstance() {
        produceHelper = null;
    }

    // 싱글톤 생성 끝
```

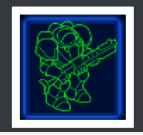
싱글톤 방식 채용

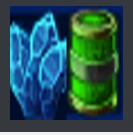
3. 유닛



I. 유닛 종류 선택
II. 유닛 번호 선택
III. 유닛 기능 사용

3. Unit 시스템



◆ 1. 메뉴 선택 -> 2. 종류 선택 -> 3. 번호 선택 -> 4. 기능 선택 -> 5. 종료

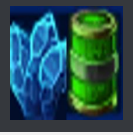
```
---------------------------------------------------- 패널 화면 ----------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
u
유닛 창을 보여줍니다
        Unit<1> : TERRAN SCV      체력 : 60/60    공격력 : 5 방어력 : 1      상태 : staying    위치 : (1,0)
        Unit<2> : TERRAN SCV      체력 : 60/60    공격력 : 5 방어력 : 1      상태 : staying    위치 : (2,3)
        Unit<3> : TERRAN SCV      체력 : 60/60    공격력 : 5 방어력 : 1      상태 : staying    위치 : (3,0)
        Unit<4> : TERRAN SCV      체력 : 60/60    공격력 : 5 방어력 : 1      상태 : staying    위치 : (4,0)
        Unit<5> : TERRAN Marine   체력 : 40/40    공격력 : 6 방어력 : 1      상태 : staying    위치 : (5,0)
        Unit<6> : TERRAN Marine   체력 : 40/40    공격력 : 6 방어력 : 1      상태 : staying    위치 : (30,20)
조작할 유닛을 고르세요 (s : SCV, m : Marine, c : Medic, q : 종료):
m
        입력번호<1> : Unit<5> : TERRAN Marine      체력 : 40/40    공격력 : 6 방어력 : 1      상태 : staying    위치 : (5,0)
        입력번호<2> : Unit<6> : TERRAN Marine      체력 : 40/40    공격력 : 6 방어력 : 1      상태 : staying    위치 : (30,20)
 사용할 유닛 번호를 고르세요 , 0 : 전체 선택, q : 종료
```

```
0
다중 명령을 선택했습니다.
다중 유닛 명령 콘솔입니다. m : 전체 이동, s : 전체 정지, a : 전체 공격, q: 종료
```

```
1
TERRAN Marine을(를) 선택했습니다.
유닛 명령 콘솔입니다. 명령을 선택하세요.
유닛 명령 콘솔입니다. m : 이동, s : 정지, a : 공격, q: 종료
```

3. Unit 시스템



◆ 1. 메뉴 선택 : UnitMenu.showMenu(Player, Scanner);

```
---------------------------------------------------- 패널 화면 ----------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
u
유닛 창을 보여줍니다
        Unit<1> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (1,0)
        Unit<2> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (2,3)
        Unit<3> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상
        Unit<4> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상
        Unit<5> : TERRAN Marine   체력 : 40/40     공격력 : 6 방어력 : 1      상
        Unit<6> : TERRAN Marine   체력 : 40/40     공격력 : 6 방어력 : 1      상
        Unit<7> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상
        Unit<8> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상
        Unit<9> : TERRAN SCV      체력 : 60/60     공격력 : 5 방어력 : 1      상
        Unit<10> : TERRAN SCV     체력 : 60/60     공격력 : 5 방어력 : 1      상
조작할 유닛을 고르세요 (s : SCV, m : Marine, c : Medic, q : 종료):
```

```java
public class UnitMenu {
        private static UnitMenu instance;

        private UnitMenu() {
        }

        public static UnitMenu getInstance() {
            if (instance == null) {
                instance = new UnitMenu();
            }
            return instance;
        }
```

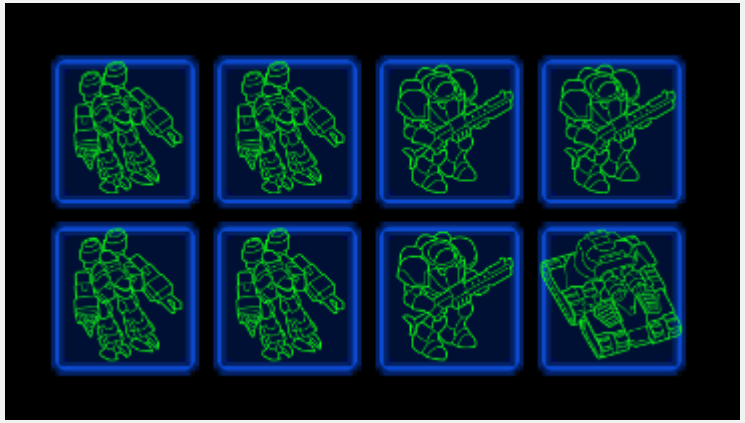
```java
    public void showMenu(Player human, Scanner scanner) {
        System.out.println("유닛 창을 보여줍니다");
        human.showUnitList();
        System.out.println("조작할 유닛을 고르세요 (s : SCV, m : Marine, c : Medic, q : 종료):");
        String input = scanner.next().toLowerCase(); // 메뉴판 인쇄랑 input은 세트

        switch (input) {
        case "s":
            UnitController.controlUnit(human, scanner, Scv.class);
            break;

        case "m":
            UnitController.controlUnit(human, scanner, Marine.class);
            break;

        case "c":
            UnitController.controlUnit(human, scanner, Medic.class);
            break;

        case "q":
            System.out.println("유닛 조작을 종료합니다.");
            break;

        default:
            System.out.println("잘못된 입력입니다");
            break;
        }
    }

}
```

3. Unit 시스템




◆ 2. 종류 선택 : UnitMenu.showMenu(Player, Scanner);

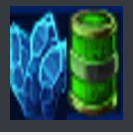
```
    public void showMenu(Player human, Scanner scanner) {
        System.out.println("유닛 창을 보여줍니다");
        human.showUnitList();
        System.out.println("조작할 유닛을 고르세요 (s : SCV, m : Marine, c : Medic, q : 종료):");
        String input = scanner.next().toLowerCase(); // 메뉴판 인쇄랑 input은 세트

        switch (input) {
        case "s":
            UnitController.controlUnit(human, scanner, Scv.class);
            break;

        case "m":
            UnitController.controlUnit(human, scanner, Marine.class);
            break;

        case "c":
            UnitController.controlUnit(human, scanner, Medic.class);
            break;

        case "q":
            System.out.println("유닛 조작을 종료합니다.");
            break;

        default:
            System.out.println("잘못된 입력입니다");
            break;
        }
    }

}
```



3. Unit

시스템



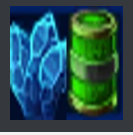
◆ 3. 번호 선택 : UnitController.controlUnit(Player, Scanner, 유닛 종류)

```
s
        입력번호<1> : Unit<1> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (1,0)
        입력번호<2> : Unit<2> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (2,3)
        입력번호<3> : Unit<3> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (3,0)
        입력번호<4> : Unit<4> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : gathering 위치 : (4,0)
        입력번호<5> : Unit<7> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying   위치 : (2,0)
        입력번호<6> : Unit<8> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying   위치 : (6,0)
        입력번호<7> : Unit<9> : TERRAN SCV   체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying   위치 : (6,0)
        입력번호<8> : Unit<10> : TERRAN SCV          체력 : 60/60     공격력 : 5 방어력 : 1      상태 : staying   위치 : (7,0)
사용할 유닛 번호를 고르세요 , 0 : 전체 선택, q : 종료
```

3. Unit

시스템

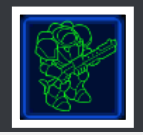


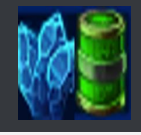



◆ 3. 번호 선택 : UnitController

```java
public class UnitController {

    static int unitIndex;
    public static void controlUnit(Player human, Scanner scanner, Class<? extends Unit> unitType) {
        List<? extends Unit> unitList = human.getListOfType(unitType);
        while (true) {
            if (unitList.size() == 0) {
                System.out.println("유닛이 없습니다.");
                break;
            }
            else
                unitIndex = 1;
                for (Unit unit : unitList) {
                    System.out.printf("\t입력번호<%d> : %s\n", unitIndex++, unit);
                }
                System.out.println(" 사용할 유닛 번호를 고르세요 , 0 : 전체 선택, q : 종료 ");
                int unitNumber = getUnitNumber(scanner);
                if (unitNumber == -2) {
                    break;
                }
                else if (unitNumber == -1) {
                    boolean allScv = unitList.stream().allMatch(unit -> unit instanceof Scv);
                    if (allScv) {
                        MultiCommandScv((List<Scv>) unitList,scanner);
                    } else {
                        System.out.println("다중 명령을 선택했습니다.");
                        MultiCommand((List<Unit>) unitList,scanner);
                    }
                }

                else {
                    // 개별 유닛 선택 시
                    System.out.println(unitList.get(unitNumber).getName() + "을(를) 선택했습니다.");


                    handleUnitMenu(unitList.get(unitNumber), scanner);
            }
        }
    }
```

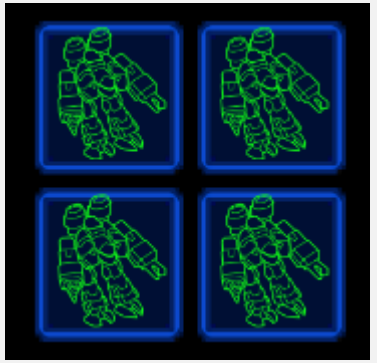
3. Unit 시스템



◆ 3. 번호 선택 : multiCommand

```
조작할 유닛을 고르세요 (s : SCV, m : Marine, c : Medic, q : 종료):
s
        입력번호<1> : Unit<1> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : staying    위치 : (1,0)
        입력번호<2> : Unit<2> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : staying    위치 : (2,3)
        입력번호<3> : Unit<3> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : staying    위치 : (3,0)
        입력번호<4> : Unit<4> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : staying    위치 : (4,0)
 사용할 유닛 번호를 고르세요 , 0 : 전체 선택, q : 종료
0
SCV 전용 멀티 커맨드 콘솔입니다. m : 전체 이동, s : 전체 정지, a : 전체 공격, g : 전체 자원 채굴 q: 종료
g
자원을 채굴합니다. m: 미네랄 캐기, g: 가스 캐기, q: 종료
m
미네랄을 캐러갑니다!!!
58
멀티 커맨드 : 미네랄 캐기 성공
미네랄을 캐러갑니다!!!
66
멀티 커맨드 : 미네랄 캐기 성공
미네랄을 캐러갑니다!!!
74
멀티 커맨드 : 미네랄 캐기 성공
미네랄을 캐러갑니다!!!
82
멀티 커맨드 : 미네랄 캐기 성공
```



3. Unit

시스템

◆ 3. 번호 선택 : Command

I. 일반 유닛 명령

```
private static void Command(Unit unit, Scanner scanner) {
    System.out.println("유닛 명령 콘솔입니다. m : 이동, s : 정지, a : 공격, q: 종료");
    String input = scanner.next().toLowerCase();

    switch (input) {
        case "m":
            int x;
            int y;
            System.out.println("유닛을 이동합니다. 목표 x좌표를 입력하세요");
            x = scanner.nextInt();
            System.out.println("유닛을 이동합니다. 목표 y좌표를 입력하세요");
            y = scanner.nextInt();
            unit.move(x, y);
            break;

        case "s":
            unit.stop();
            break;

        case "a" :
            unit.attack();
            break;

        case "q":
            System.out.println("메인 메뉴로 돌아갑니다.");
            break;

        default:
            System.out.println("에러코드 116");
            break;
    }
}
```

3. Unit

시스템

◆ 3. 번호 선택 : CommandScv

I. SCV 명령

```
1
TERRAN SCV을(를) 선택했습니다.
유닛 명령 콘솔입니다. 명령을 선택하세요.
SCV 전용 유닛 명령 콘솔입니다. m : 이동, s : 정지, a : 공격, b : 건물 짓기 , g : 자원 채굴 q: 종료
```

```
private static void CommandScv(Scv scv, Scanner scanner) {
    System.out.println("SCV 전용 유닛 명령 콘솔입니다. m : 이동, s : 정지, a : 공격, b : 건물 짓기 , g : 자원 채굴 q: 종료");
    String input = scanner.next().toLowerCase();
```

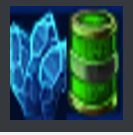
4. 명령

I. 이동
II. 공격
III. 건설
IV. 채굴

4. 명령 : 이동

시스템



◆ 1. 이동 : Move()

```
다중 명령을 선택했습니다.
다중 유닛 명령 콘솔입니다. m : 전체 이동, s : 전체 정지, a : 전체 공격, q: 종료
m
유닛을 이동합니다. 목표 x좌표를 입력하세요
15
유닛을 이동합니다. 목표 y좌표를 입력하세요
20
유닛을 멈춥니다.
이동 명령을 시작합니다!!!
유닛을 멈춥니다.
이동 명령을 시작합니다!!!
```

```
이 유닛의 x좌표는 : 15
이 유닛의 y좌표는 : 19
x거리는0입니다
y거리는1입니다
TERRAN Marine이 y축 이동했습니다
현재 위치는 : (15, 20)
이동이 종료됩니다.
현재 위치는 : (15, 20)
유닛을 멈춥니다.
```

각각의 유닛이 서로 다른 스케쥴을 가져야 하기 때문에 굉장히 어려웠다.

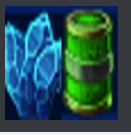
이동중...

```
이 유닛의 x좌표는 : 5
이 유닛의 y좌표는 : 0
x거리는10입니다
y거리는20입니다
TERRAN Marine이 y축 이동했습니다
현재 위치는 : (5, 1)
이 유닛의 x좌표는 : 30
이 유닛의 y좌표는 : 20
x거리는15입니다
y거리는0입니다
TERRAN Marine이 x축 이동했습니다
현재 위치는 : (29, 20)
이 유닛의 x좌표는 : 5
이 유닛의 y좌표는 : 1
x거리는10입니다
y거리는19입니다
TERRAN Marine이 y축 이동했습니다
현재 위치는 : (5, 2)
이 유닛의 x좌표는 : 29
이 유닛의 y좌표는 : 20
x거리는14입니다
y거리는0입니다
TERRAN Marine이 x축 이동했습니다
현재 위치는 : (28, 20)
이 유닛의 x좌표는 : 5
이 유닛의 y좌표는 : 2
x거리는10입니다
y거리는18입니다
TERRAN Marine이 y축 이동했습니다
```

4. 명령 : 이동

시스템



◆ 1. 이동 : Move(); 거리계산

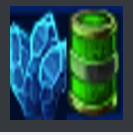
```
@Override
public void move(int x, int y) {
    this.stop();
    this.executorService = Executors.newScheduledThreadPool(1);
    this.delay();
    this.setState(State.moving);
    System.out.println("이동 명령을 시작합니다!!!");

    this.executorService.scheduleAtFixedRate(() -> {

        System.out.println("이 유닛의 x좌표는 : " + getX()); // 내 x좌표
        System.out.println("이 유닛의 y좌표는 : " + getY()); // 내 y좌표

        int distanceX = Math.abs(x - getX()); // x축 거리
        int distanceY = Math.abs(y - getY()); // y축 거리

        System.out.println("x거리는" + distanceX + "입니다");
        System.out.println("y거리는" + distanceY + "입니다");
```

ExecutorService.scheduleAtFixedRate(runnable,int,int,timeUnit)를 사용했다.

4. 명령 : 이동

시스템



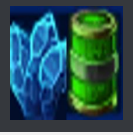
◆ 1. 이동 : Move(); 상황 판단 후 행동

```
        if (distanceX == 0 && distanceY == 0) {
            System.out.println("이동이 종료됩니다.");
            System.out.println("현재 위치는 : (" + getX() + ", " + getY() + ")");
            this.stop();

        } else if (Math.abs(distanceX) >= Math.abs(distanceY)) {
            this.setX(getX() + Integer.compare(x, this.getX())); // 현재 위치에서 목표 지점 방향으로 1 칸 이동
            Map.getCell(getX(), getY()).setUnit(null);
            Map.getCell(getX() + Integer.compare(x, getX()), getY()).setUnit(this);
            System.out.println(getName() + "이 x축 이동했습니다");
            System.out.println("현재 위치는 : (" + getX() + ", " + getY() + ")");
        }

        else {
            setY(getY() + Integer.compare(y, getY())); // 현재 위치에서 목표 지점 방향으로 1 칸 이동
            Map.getCell(getX(), getY()).setUnit(null);
            Map.getCell(getX(), getY() + Integer.compare(y, getY())).setUnit(this);
            System.out.println(getName() + "이 y축 이동했습니다");
            System.out.println("현재 위치는 : (" + getX() + ", " + getY() + ")");
        }

        if (getX() == x && getY() == y) {
            // 목표 지점에 도달했을 때 작업 종료
            System.out.println("이동이 종료됩니다.");
            System.out.println("현재 위치는 : (" + getX() + ", " + getY() + ")");
            this.stop();
        }
    }, 1, 1, TimeUnit.SECONDS);
}
```

4. 명령 : 공격

시스템



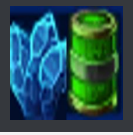
◆ 2. 공격 : search(); 타겟 검색

```
public Unit search() {
    System.out.println("타겟 서칭~");
    Unit target = null;
    Unit maybeTarget = null;
    //범위검사
    for(int i = getX()-this.range; i <= getX() + this.range; i++) {
        for(int j = getY()-this.range; j <= getY() + this.range; j++) {
            if (Map.getCell(i,j)!=null) {
                maybeTarget = Map.getCell(i, j).getUnit();
                if (maybeTarget != null && maybeTarget.getPlayer() != this.getPlayer())
                {
                    target = maybeTarget;
                    break;
                }
            }
        }
        if(target != null)
            {break;}
    }
    return target;
}
```

4. 명령 : 공격

시스템



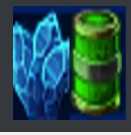
◆ 2. 공격 : attack(); 공격

```java
@Override
public void attack() {
    stop();
    Unit target = search();
    if (target != null) {
        System.out.println("공격 명령을 시작합니다!!!");
        setState(State.fighting);
        executorService = Executors.newScheduledThreadPool(1);
        executorService.scheduleAtFixedRate(() -> {
            if (target.getCurrentHp() > 0) {
                target.setCurrentHp(target.getCurrentHp() - (this.getAtk() - target.getDef()));
                System.out.println(this.getPlayer().getPlayerName() + "의 유닛 " + this.getName() + "이 "
                        + target.getPlayer().getPlayerName() + " 의 유닛" + target.getName() + "을 공격합니다.");
                System.out.println("피격 유닛 체력 : " + target.getCurrentHp());
            } else {
                target.delete();
                stop();
            }
        }, 1, 1, TimeUnit.SECONDS);
    } else
        System.out.println("타겟이 없습니다.");
}
```

4. 명령 : 건설

시스템



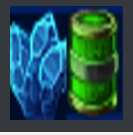
◆ 3. 건설 : scv.produceCommandCenter(int, int);

```
case "b":
    System.out.println("설치할 건물을 고르세요.");
    System.out.println(" 건물 리스트 -> c : Command Center, b : Barracks, s : Supply depot, r : Refinery q : 종료");
    input = scanner.next().toLowerCase();
    switch (input) {
    case "c":
        scv.produceCommandCenter(scv.getX(), scv.getY());
        break;
    case "b":
        scv.produceBarracks(scv.getX(), scv.getY());
        break;
    case "s":
        scv.produceSupplyDepot(scv.getX(), scv.getY());
        break;
    case "r":
        scv.produceRefinery(scv.getX(), scv.getY());
        break;
    }

    break;
```

4. 명령 : 건설

시스템



◆ 3. 건설 : scv.produceCommandCenter(int, int);

```
public void produceCommandCenter(int x, int y) {
    if (getState() != State.staying) {
        System.out.println("다른 일을 하는 중입니다.");
    } else if (produceHelper.validCheck(player, Data.COMMANDCENTER_M, Data.COMMANDCENTER_G, 0))
        // 자원 있음. 건설 시작.
        {
        setState(State.building);
    System.out.println("건설을 시작합니다.");
    executorService = Executors.newScheduledThreadPool(1);
    executorService.schedule(() -> {
        Building CommandCenter = new CommandCenter(player, x, y);
        setState(State.staying);
        stop();
        System.out.println(CommandCenter.getName() + " 이 완성되었습니다.");
    }, Data.COMMANDCENTER_TIME, TimeUnit.SECONDS);

        }
}
```

4. 명령 : 채굴

시스템




◆ 4. 채굴 : scv.gatherMineral();

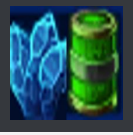
```
1
TERRAN SCV을(를) 선택했습니다.
유닛 명령 콘솔입니다. 명령을 선택하세요.
SCV 전용 유닛 명령 콘솔입니다. m : 이동, s : 정지, a : 공격, b : 건물 짓기 , g : 자원 채굴 q: 종료
```

```
        case "g":
            System.out.println("자원을 채굴합니다. m: 미네랄 캐기, g: 가스 캐기, q: 종료");
            input = scanner.next().toLowerCase();

            switch (input) {
            case "m":
                scv.gatherMineral();
                System.out.println("단독 명령 : 미네랄을 캐러갑니다");
                break;
            case "g":
                scv.gatherGas();
                System.out.println("단독 명령 : 가스를 캐러갑니다");
                break;
            case "q":
                System.out.println("유닛 명령 콘솔로 되돌아갑니다.");
                break;
            default:
                System.out.println("잘못된 입력입니다.");
                break;
            }
            break;
```

4. 명령 : 채굴

시스템

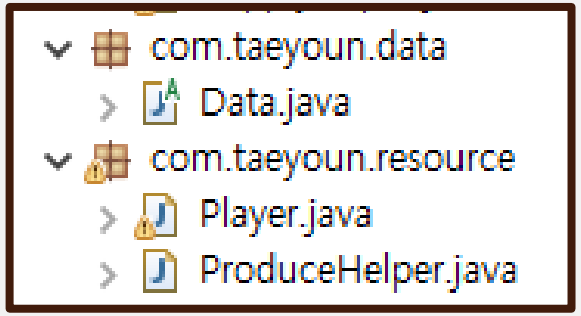


◆ 4. 채굴 : scv.gatherMineral();

```
//이건 gatherMineral 이 SCV에만 있는 특수 기능이라서 SCV에 stop을 오버라이딩 시켜놔야 되는것같다.
public void gatherMineral() {
    if (getState() == State.staying) {
        executorService = Executors.newScheduledThreadPool(1);

        setState(State.gathering);
        System.out.println("미네랄을 캐러갑니다!!!");
        executorService.scheduleAtFixedRate(() ->
            {
                player.setMinerals(player.getMinerals() + 8);
/*보여주기용*/      System.out.println(player.getMinerals());
            }
            , 0, 5, TimeUnit.SECONDS);
    } else
        System.out.println("바빠서 명령을 수행할 수 없습니다.");
}
```

4. 자원



I. 데이터 관리

II. 플레이어 자원 관리

III. 생산 가능 여부 판단

5. 자원




◆ 1. 데이터 관리 : public static final

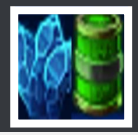
```
//MARINE
    public static final String MARINE_NAME = "TERRAN Marine";
    public static final String MARINE_STATEMENT1 = "한 판 붙어볼까, 애송이?";
    public static final String MARINE_STATEMENT2 = "전투준비완료. 명령만 내리십쇼(마린)";
    public static final String MARINE_STATEMENT3 = "전투중.(마린)";
    public static final String MARINE_STATEMENT4 = "사망했습니다.(마린)";
    public static final int MARINE_M = 50;
    public static final int MARINE_G = 0;
    public static final int MARINE_P = 1;
    public static final int MARINE_TIME = 24;
    public static final int MARINE_HPMAX = 40;
    public static final int MARINE_ATK = 6;
    public static final int MARINE_DEF = 1;
    public static final int MARINE_ATKRANGE = 5;
```

```
//COMMANDCENTER

    public static final String COMMANDCENTER_NAME = "TERRAN Command Center";
    public static final String COMMANDCENTER_STATEMENT = "COMMANDCENTER입니다.";
    public static final int COMMANDCENTER_HP = 2500;
    public static final int COMMANDCENTER_M = 400;
    public static final int COMMANDCENTER_G = 0;
    public static final int COMMANDCENTER_TIME = 75;
```

5. 자원

시스템



◆ 2. 플레이어 자원 관리

```
------------------------------------------------------ 패널 화면 ------------------------------------------------------
패널 화면입니다. 기능을 선택하세요 ( b : 건물 선택하기, u : 유닛 선택하기, s : 자원 표시하기, q : 종료
s
╞========================================== Current Game Status ==========================================
Player: 김태연
Minerals: 338
Gas: 0
Population: 10
Units:
   Unit<1> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : gathering 위치 : (1,0)
   Unit<2> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : gathering 위치 : (2,3)
   Unit<3> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : gathering 위치 : (3,0)
   Unit<4> :TERRAN SCV  체력 : 60/60      공격력 : 5 방어력 : 1      상태 : gathering 위치 : (4,0)
   Unit<5> :TERRAN Marine       체력 : 40/40      공격력 : 6 방어력 : 1      상태 : staying   위치 : (5,0)
   Unit<6> :TERRAN Marine       체력 : 40/40      공격력 : 6 방어력 : 1      상태 : staying   위치 : (30,20)
```

5. 자원

시스템




◆ 플레이어 유닛 관리 : showUnitList()

```
public void showUnitList() {
    System.out.println("Buildings:");
    for (Unit unit : unitList) {
        System.out.printf("    " + unit + "\n");
    }
}
```

◆ 2. 플레이어 유닛 추가 : addUnit(Unit)

```
public void addUnit(Unit unit) {
    getUnitList().add(unit);
    System.out.println("unitList에 " + unit.getName() + " 를(을) 추가 했습니다");
}
```

5. 자원



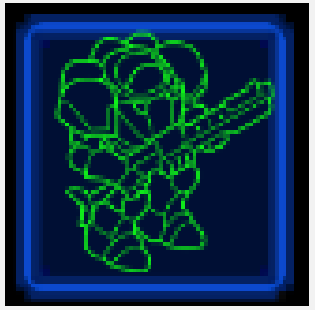

◆ 3. 생산 가능 여부 확인 : class ProduceHelper

```java
public class ProduceHelper {

    // 싱글톤

    private static ProduceHelper produceHelper;
    public static ProduceHelper getInstance() {
        if (produceHelper == null) {
            produceHelper = new ProduceHelper();
        }
        return produceHelper;
    }

    public static void freeInstance() {
        produceHelper = null;
    }

    // 싱글톤 생성 끝
```

```java
    public boolean validCheck(Player player, int CostM, int CostG, int CostP) {
        if (!isPopulation(player, CostP)) {
            System.out.println("서플라이 디팟을 더 지으세요");
            return false;
        }
        if (!isMineral(player, CostM)) {
            System.out.println("미네랄이 부족합니다");
            return false;
        }
        if (!isGas(player, CostG)) {
            System.out.println("가스가 부족합니다");
            return false;
        } else {
            // System.out.println("valid check 성공"); //이 메서드가 호출되는지 확인용
            player.setMinerals(player.getMinerals() - CostM);
            player.setGas(player.getGas() - CostG);
            player.setPopulation(player.getPopulation() - CostP);

            return true;
        }
    }
}
```

SingleTon 사용

[Q&A]

Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ
시스템				
시스템	건물	유닛	명령	자원


STARCRAFT
BROOD WAR
감사합니다.
12세이용가
15세이용가
폭력성
Copyright © 1998-2020 Blizzard Entertainment. All rights reserved.
BLIZZARD ENTERTAINMENT