**객체지향프로그래밍 프로젝트 보고서**

**< 게임명: 대학생 키우기 >**

20162528 윤성식

20192766 마민정

20192807 한보혜

**객체지향 구현 수준**

게임 설계 적정성(Use Case 다이어그램)

클래스 설계 적정성(클래스 다이어그램)

객체지향 특징(상속, 캡슐화, 다형성) 구현 정도

**게임 기능 구현**

기능별 구현 적정성(시퀀스 다이어그램)

**게임 완성도**

게임 소개

게임 구성도

게임 사용자 매뉴얼(게임 실행 화면 별 캡쳐 및 설명)

발표자료(시연동영상 1분 이내)

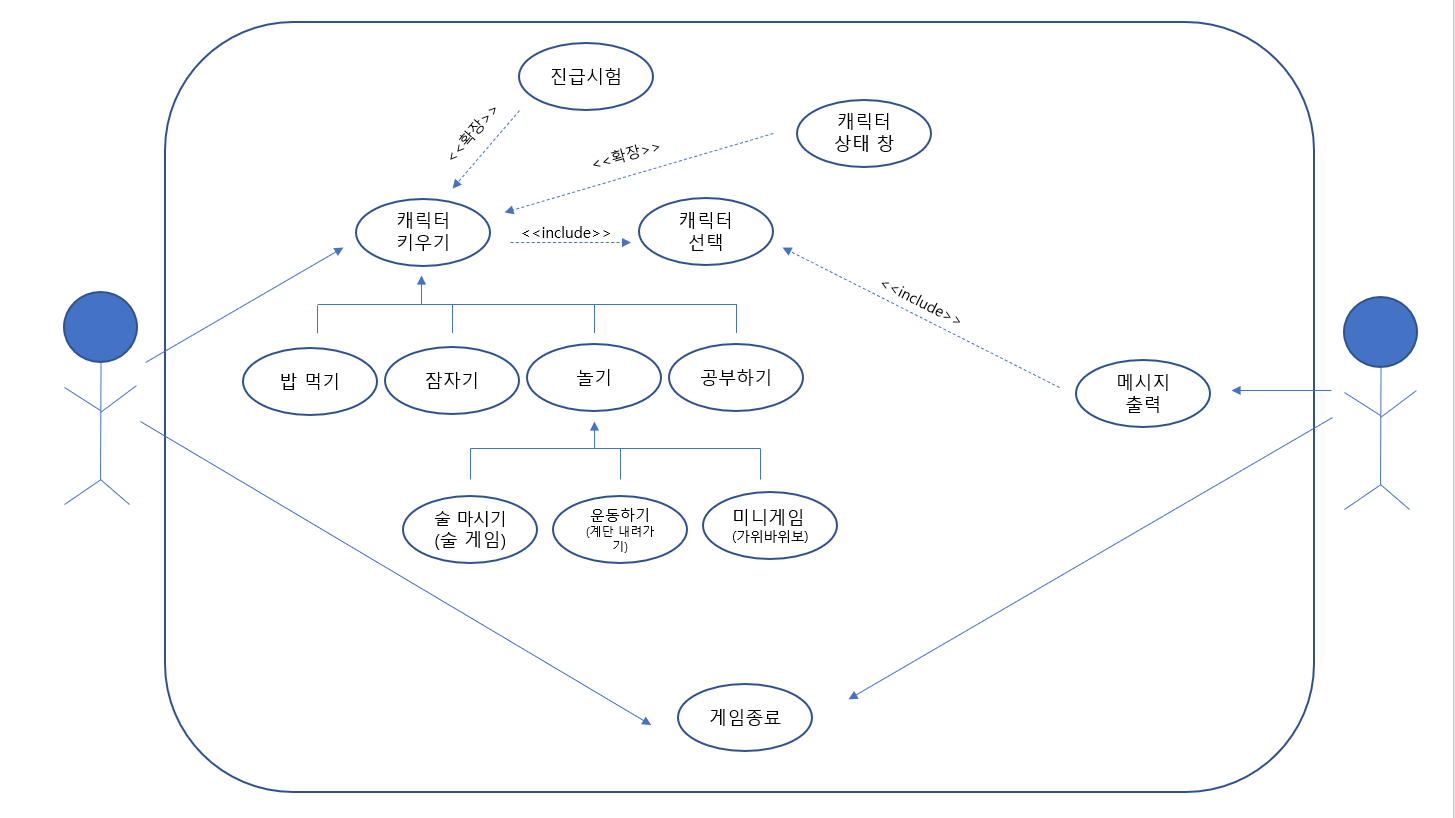
**부록**

소스코드

**객체지향 구현 수준**

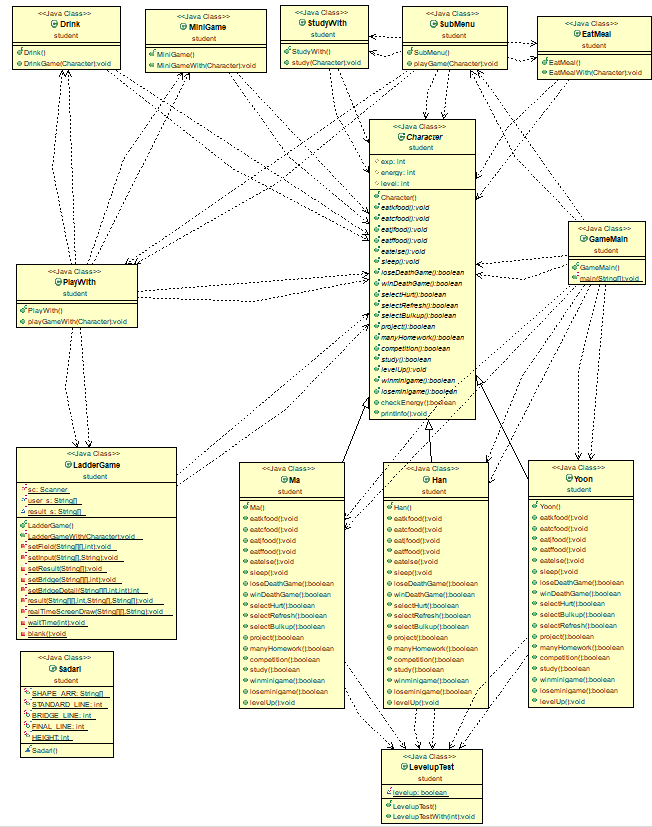
* **게임 설계 적정성(Use Case 다이어그램)**

[ Use case Diagram ]



* **클래스 설계 적정성(클래스 다이어그램)**

[ Class Diagram ]



* **객체지향 특징(상속, 캡슐화, 다형성) 구현 정도**

**게임 기능 구현**

* **기능별 구현 적정성(시퀀스 다이어그램)**

**게임 완성도**

* **게임 소개**

‘대학생 키우기’에는 각각의 특색을 가진 ‘마씨’, ‘윤씨’, ‘한씨’ 3명의 대학생이 존재한다. 게임 시작 시 이들의 특성을 나타내는 메시지가 뜨며 이 중 한 명을 선택하여 경험치(exp)를 쌓아가며 1학년부터 4학년까지 진급을 시켜야 한다. 대학생들은 진짜 사람처럼 체력(energy)이 존재하고 체력이 다 떨어지면 게임 오버가 되기 때문에 사용자는 각 캐릭터의 특성을 잘 파악하여 선택지를 골라야 한다. 각 학년이 끝날 때마다 진급(level) 퀴즈를 보는데, 실패 시에는 유급되어 해당 학년을 다시 진행해야 한다. 최종적으로는 4학년까지 대학생을 진급시킨 뒤 졸업시키는 것이 목표이다. 중간에 에너지가 다 닳거나 졸업했을 경우 게임이 종료된다.



* **게임 구성도**

처음에 캐릭터를 생성하면 현재 캐릭터의 경험치, 에너지, 학년을 출력해준다. 1. 밥 먹이기 2. 잠 재우기 3. 놀아 주기 4. 공부하기 5. 종료 중에 원하는 행동을 고르면 캐릭터별 특성에 따라 각기 다르게 경험치와 에너지가 증감된다.

1. 밥 먹이기

밥 먹이기를 선택했을 경우 한식, 중식, 일식, 양식, 간식 중에 원하는 음식을 선택하면 된다. 캐릭터가 생성되기 전 나오는 힌트를 통해 각 캐릭터가 어떤 음식을 먹었을 때 에너지가 많이 오를지 유추할 수 있다.

1. 잠 재우기

잠 재우기를 선택했을 경우 각 캐릭터 별로 에너지가 소폭 증가한다.

1. 놀아 주기

놀아 주기를 선택했을 경우 술 마시기, 운동하기, 미니게임 하기 중에 하나를 선택하면 된다. 술 마시기(Drink)를 선택했을 경우 ‘The Game of Death’가 시작된다. 참여할 인원과 본인의 번호를 지정하면 인원 수만큼 번갈아가며 서로를 지목하게 된다. 마지막에 지목된 사람이 본인이 아니면 게임에서 승리한다. 미니게임하기(miniGame)를 선택했을 경우 ‘가위바위보 게임’이 실행된다. 이 게임도 동일하게 가위바위보 결과에 따라 경험치와 에너지가 증감된다. 운동하기(Exercise)를 선택했을 경우 ‘계단 오르기 게임’이 시작된다. A, B, C 3가지 계단 중 하나를 선택하면 선택한 계단에 따라 근성장, 다침, 개운함 3가지 결과 중 하나가 랜덤으로 결정된다. 그 결과에 따라 경험치와 에너지가 증감된다. 공부하기

공부하기를 선택했을 경우 0에서 5사이의 난수를 발생시킨다. 난수가 0일 경우 프로젝트(project) 수행에 참여하게 되고 난수가 2일 경우는 과제 폭탄(manyHomework)을 맞게 되며 난수가 4일 경우는 공모전(Competition)에 참여하게 된다. 그 외에 난수가 1, 3, 5일 경우는 개인공부를 하게 된다. 공부하기를 수행하는 모든 시간 동안은 지연시간을 주어 생동감을 주었다.

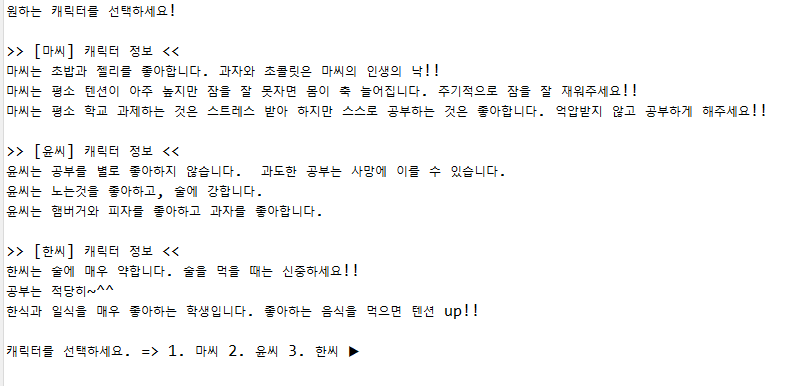
1. 종료

종료를 선택했을 경우 경험치, 에너지와 상관없이 게임이 종료된다.

* **게임 사용자 매뉴얼(게임 실행 화면 별 캡쳐 및 설명)**

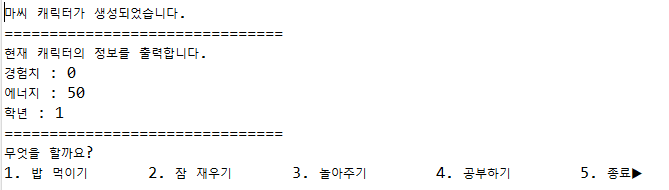
**실행화면1 ) 캐릭터 선택**

처음 게임을 실행하면 각 캐릭터의 정보를 알려주며 캐릭터를 선택하는 창이 뜬다. 마씨, 윤씨, 한씨 각각의 캐릭터별로 어떤 음식을 좋아하는지, 어떤 행동을 해야 경험치와 에너지가 오를 수 있는지 힌트가 출력된다. 그 정보를 보고 원하는 캐릭터를 선택하면 된다.

****

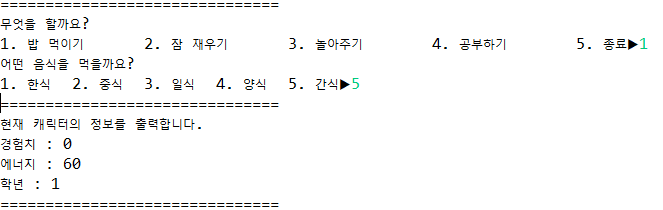
**실행화면2 ) 캐릭터 정보 출력 & 행동 선택**

캐릭터가 생성되면 현재 캐릭터의 정보를 출력해준다. 기본적으로 모든 캐릭터의 경험치는 0, 에너지는 50으로 통일했다. 정보를 출력한 뒤에는 본격적으로 캐릭터를 키우기 위해 ‘밥 먹이기, 잠 재우기, 놀아 주기, 공부하기, 종료’ 중에 하나의 행동을 선택하면 된다.

****

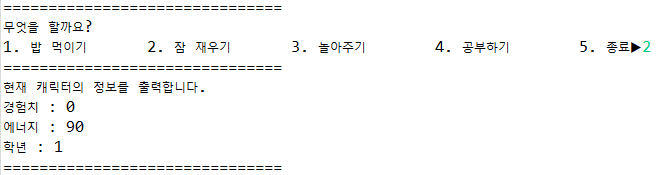
**실행화면3 ) 밥 먹이기**

밥 먹이기를 선택했을 경우 한식, 중식, 일식, 양식, 간식 중에 하나의 메뉴를 선택하게 된다. 캐릭터 힌트를 기반으로 특정 캐릭터가 어떤 음식을 좋아할 지 추측해서 밥을 먹이면 에너지가 소폭 상승한다. 상승한 에너지를 포함한 캐릭터 정보는 각 행동이 끝날 때마다 매번 출력된다.

****

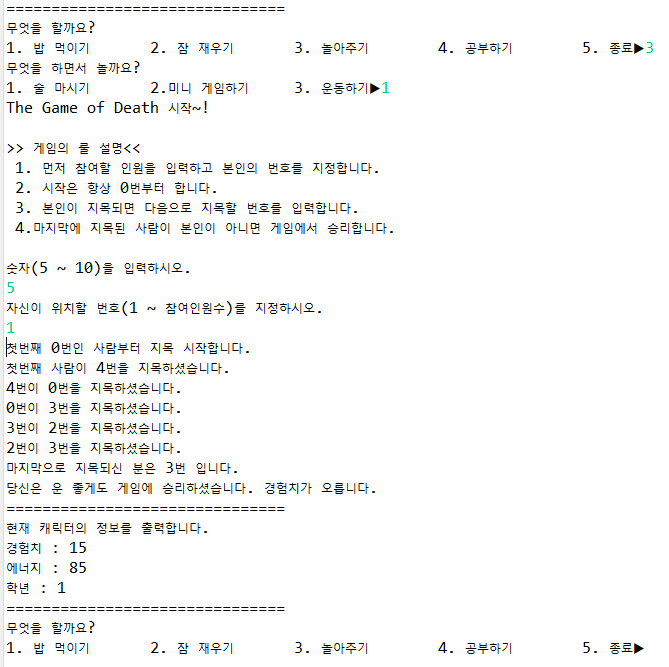
**실행화면4 ) 잠 재우기**

잠 재우기를 선택했을 경우 각 캐릭터 별로 에너지가 소폭 상승한다. 상승한 에너지를 포함한 캐릭터 정보가 출력된다.



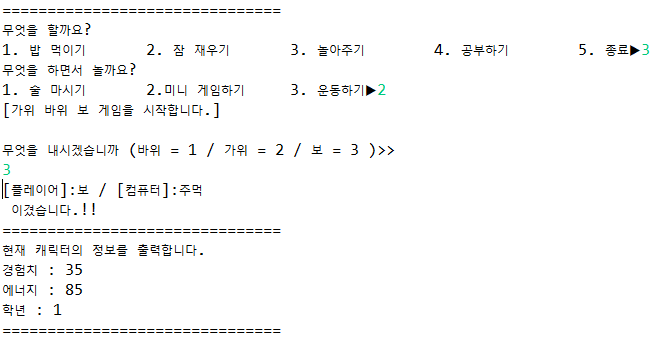
**실행화면5 ) 놀아 주기 -> 술 마시기**

놀아 주기를 선택했을 경우 술 마시기, 미니 게임하기, 운동하기 중 하나를 선택하게 된다. 그 중 술 마시기를 선택했을 경우 ‘The Game of Death’가 시작된다. 게임이 시작하기 전, 게임 룰이 Console창에 표시된다. 참여할 인원과 본인의 번호를 지정하면 인원 수만큼 번갈아가며 서로를 지목하게 된다. 마지막에 지목된 사람이 본인이 아니면 게임에서 승리한다. 게임에서 승리했을 경우 경험치는 오르지만 게임을 했기 때문에 에너지는 소폭 감소한다. 게임에서 패배했을 경우 술을 마셔야 하기 때문에 경험치는 소폭 상승하지만 에너지가 대폭 감소한다. 이 게임 또한 캐릭터별로 특성에 따라 상승, 하강 폭을 다르게 지정해 주었다.

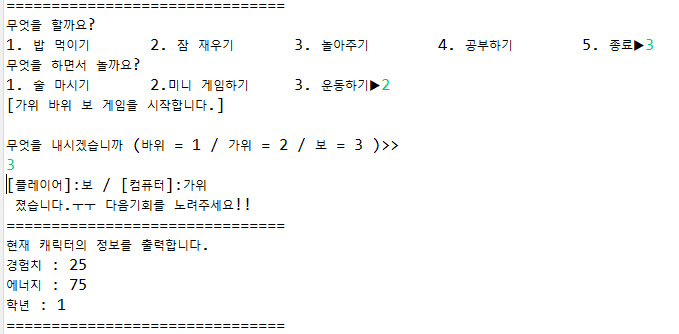


**실행화면6 ) 놀아 주기 -> 미니 게임하기**

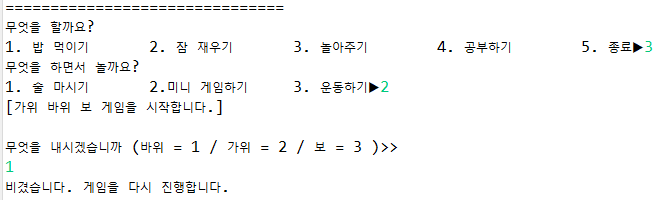
놀아 주기를 선택했을 경우 술 마시기, 미니 게임하기, 운동하기 중 하나를 선택하게 된다. 그 중 미니게임을 선택했을 경우 플레이어는 바위-1, 가위-2, 보-3 중 하나를 선택해서 낸다. 수업시간 때 배운 가위바위보 게임은 플레이어 2명을 모두 직접 입력한 반면 이번 미니 게임은 컴퓨터가 랜덤으로 바위 가위 보 중 하나를 선택하도록 응용해서 만들었다.



게임에서 이겼을 경우는 경험치와 에너지가 모두 증가한다.



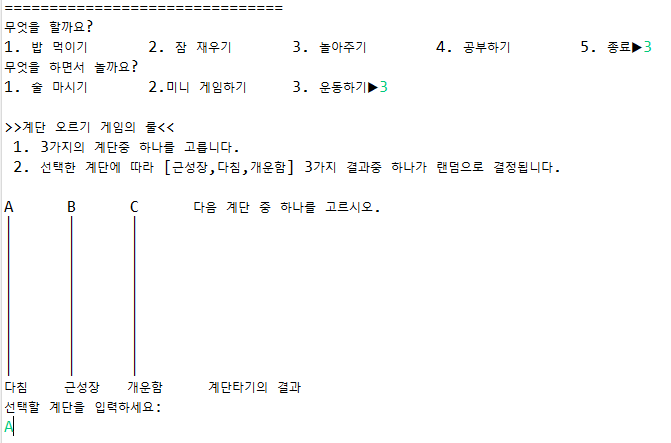
게임에서 졌을 경우 경험치는 증가하지만 에너지가 감소한다.

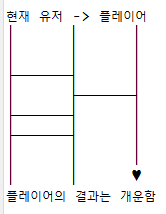
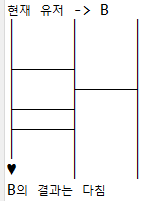
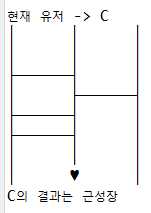


게임에서 비겼을 경우는 게임을 다시 진행한다.

**실행화면7 ) 놀아 주기 -> 운동하기**

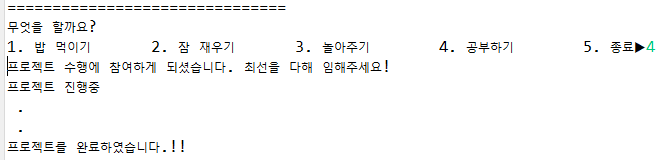
놀아 주기를 선택했을 경우 술 마시기, 미니 게임하기, 운동하기 중 하나를 선택하게 된다. 그 중 운동하기를 선택했을 경우 ‘계단 오르기 게임’이 시작된다. A, B, C 3가지 계단 중 하나를 선택하면 개운함, 다침, 근성장이 랜덤으로 선택된다. 세 가지 모두 경험치는 소폭 상승하지만 다쳤을 경우 에너지는 대폭 감소하고 개운함이나 근성장이 선택됐을 경우에는 에너지가 대폭 상승한다. 이 게임 또한 캐릭터별로 특성에 따라 상승, 하강 폭을 다르게 지정해 주었다.

****

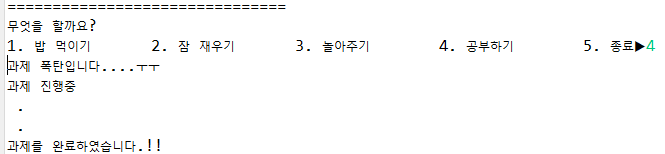
  

**실행화면8 ) 공부하기**

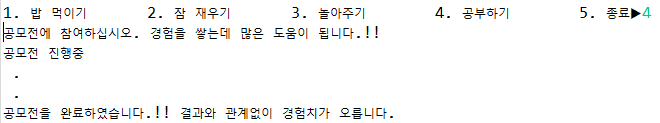
공부하기를 선택했을 경우 0에서 5사이의 난수가 발생한다.

****

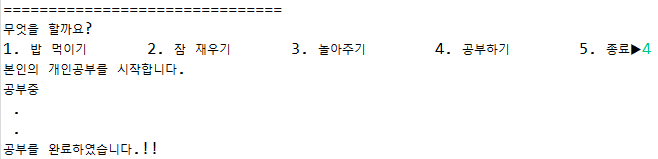
난수가 0일 경우는 프로젝트 수행에 참여하게 된다.

****

난수가 2일 경우는 과제 폭탄을 맞게 된다.

****

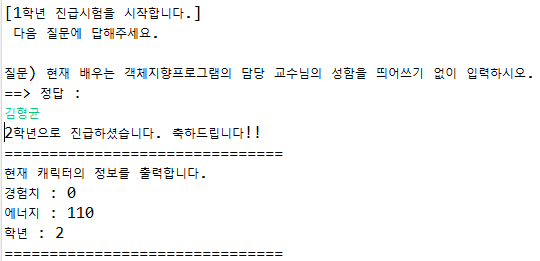
난수가 4일 경우는 공모전에 참여하게 된다.



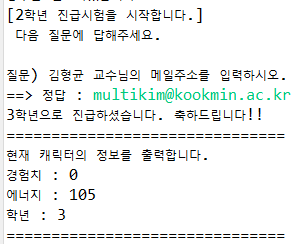
그 외에 난수가 1, 3, 5일 경우에는 개인공부를 하게 된다. 프로젝트가 걸릴 경우 프로젝트 진행중, 과제 폭탄이 걸릴 경우 과제 진행중, 공모전이 걸릴 경우 공모전 진행중, 개인공부가 걸릴 경우 공부 중을 출력하였으며 MiliSecond단위로 시간 지연을 줘서 생동감을 주었다. 이 게임 또한 캐릭터별로 특성에 따라 상승, 하강 폭을 다르게 지정해 주었다.

**실행화면9 ) 진급시험**

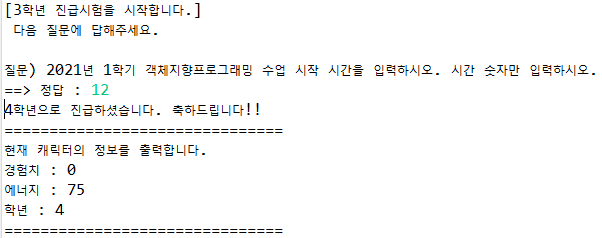
1학년부터 4학년까지 진급시키는 것이 목표이기 때문에 각 학년마다 진급시험을 보게 된다. 진급 시험은 경험치가 50이 넘을 경우는 보는 것으로 설정했다. 진급시험을 통과할 경우 경험치가 다시 0으로 초기화되고 다음 학년으로 넘어가며 통과하지 못할 경우 유급되어 해당 학년을 다시 다니게 된다.

****

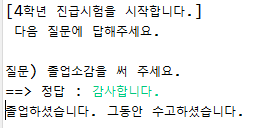
1학년 진급시험은 교수님의 성함을 띄어쓰기 없이 입력하는 것이다.

****

2학년 진급시험은 교수님의 메일주소를 입력하는 것이다.

****

3학년 진급시험은 객체지향프로그래밍 수업의 시작 시간을 입력하는 것이다.

****

4학년 진급시험은 졸업소감을 쓰는 것이다. 어떤 문장이든 상관없이 소감을 쓰면 졸업하는 것으로 설정했다.

**실행화면10 ) 종료**

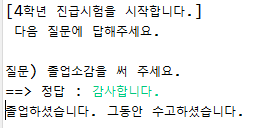
종료 조건은 총 3가지 경우가 있다.

****

게임 실행 창에서 5번 종료를 선택할 경우 종료된다.

****

게임에서 많이 지거나 공부를 많이 시키는 등 에너지가 닳아 0이 될 경우 캐릭터가 사망하게 된다.

****

마지막으로 4학년 진급시험을 마치고 졸업했을 경우 게임이 종료된다.

* **발표자료(시연 동영상 1분 이내)**

https://drive.google.com/file/d/1rALVJBOpVn9a6gGqMRh1w4M747EDVIo4/view?usp=sharing

**부록**

* **소스코드**

[ GameMain ]

**package** student;

**import** java.util.Scanner;

**public** **class** GameMain {

**public** **static** **void** main(String[] args) {

Character character = **null**;

SubMenu sm = **new** SubMenu();

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("원하는 캐릭터를 선택하세요!\n");

System.***out***.println(">> [마씨] 캐릭터 정보 <<");

System.***out***.println("마씨는 초밥과 젤리를 좋아합니다. 과자와 초콜릿은 마씨의 인생의 낙!!");

System.***out***.println("마씨는 평소 텐션이 아주 높지만 잠을 잘 못자면 몸이 축 늘어집니다. 주기적으로 잠을 잘 재워주세요!!");

System.***out***.println("마씨는 평소 학교 과제하는 것은 스트레스 받아 하지만 스스로 공부하는 것은 좋아합니다. 억압받지 않고 공부하게 해주세요!!\n");

System.***out***.println(">> [윤씨] 캐릭터 정보 <<");

System.***out***.println("윤씨는 공부를 별로 좋아하지 않습니다. 과도한 공부는 사망에 이를 수 있습니다. \n윤씨는 노는것을 좋아하고, 술에 강합니다.");

System.***out***.println("윤씨는 햄버거와 피자를 좋아하고 과자를 좋아합니다.\n");

System.***out***.println(">> [한씨] 캐릭터 정보 <<");

System.***out***.println("한씨는 술에 매우 약합니다. 술을 먹을 때는 신중하세요!!\n공부는 적당히~^^");

System.***out***.println("한식과 일식을 매우 좋아하는 학생입니다. 좋아하는 음식을 먹으면 텐션 up!!\n");

System.***out***.print("캐릭터를 선택하세요. => 1. 마씨 2. 윤씨 3. 한씨 ▶");

**int** c = Integer.*parseInt*(scanner.nextLine());

**if** (c == 1) {

character = **new** Ma();

sm.playGame(character);

} **else** **if** (c == 2) {

character = **new** Yoon();

sm.playGame(character);

} **else** **if** (c == 3) {

character = **new** Han();

sm.playGame(character);

} **else** {

System.***out***.println("잘못 입력하셨습니다!");

}

}// main()

} // class

[ Character ]

**package** student;

**import** java.util.concurrent.TimeUnit;

**public** **abstract** **class** Character {

// 멤버 변수 선언

**protected** **int** exp;

**protected** **int** energy;

**protected** **int** level;

// 추상 메소드

**public** **abstract** **void** eatkfood(); // 한식먹기

**public** **abstract** **void** eatcfood(); // 중식먹기

**public** **abstract** **void** eatjfood(); // 일식먹기

**public** **abstract** **void** eatffood(); // 양식먹기

**public** **abstract** **void** eatelse(); // 간식먹기

**public** **abstract** **void** sleep(); // 잠자기

**public** **abstract** **boolean** loseDeathGame(); // 술게임 졌을 때

**public** **abstract** **boolean** winDeathGame(); // 술게임 이겼을 때

**public** **abstract** **boolean** selectHurt();

**public** **abstract** **boolean** selectRefresh();

**public** **abstract** **boolean** selectBulkup();

**public** **abstract** **boolean** project(); // 프로젝트 당첨

**public** **abstract** **boolean** manyHomework(); // 과제폭탄

**public** **abstract** **boolean** competition(); // 공모전 나가기

**public** **abstract** **boolean** study(); // 공부하기

**public** **abstract** **void** levelUp(); // 진급

**public** **abstract** **boolean** winminigame();

**public** **abstract** **boolean** loseminigame();

// 에너지 체크

**public** **boolean** checkEnergy() {

**if** (energy > 0) {

**return** **true**;

} **else** {

**return** **false**;

}

}

// 현재 캐릭터의 상태를 출력

**public** **void** printInfo() {

**try** {

TimeUnit.***MILLISECONDS***.sleep(500);

System.***out***.println("===============================");

System.***out***.println("현재 캐릭터의 정보를 출력합니다.");

System.***out***.println("경험치 : " + exp);

System.***out***.println("에너지 : " + energy);

System.***out***.println("학년 : " + level);

System.***out***.println("===============================");

TimeUnit.***MILLISECONDS***.sleep(500);

} **catch** (Exception e) {

System.***out***.println(e);

}

}

}

[ SubMenu ]

**package** student;

**import** java.util.Scanner;

**public** **class** SubMenu {

**public** **void** playGame(Character character) {

Scanner scanner = **new** Scanner(System.***in***);

PlayWith ply = **new** PlayWith();

StudyWith stw = **new** StudyWith();

EatMeal etm = **new** EatMeal();

**while** (**true**) {

**if** (character.level > 4) {

**break**;

}

System.***out***.println("무엇을 할까요?");

System.***out***.print("1. 밥 먹이기\t2. 잠 재우기\t3. 놀아주기\t4. 공부하기\t5. 종료▶");

**int** menu = Integer.*parseInt*(scanner.nextLine());

**if** (menu == 1) {

etm.EatMealWith(character);

**if** (!character.checkEnergy()) {

System.***out***.println("에너지가 부족해서 캐릭터가 사망했습니다!");

**break**;

}

} **else** **if** (menu == 2) {

character.sleep();

} **else** **if** (menu == 3) {

ply.playGameWith(character);

**if** (!character.checkEnergy()) {

System.***out***.println("에너지가 부족해서 캐릭터가 사망했습니다!");

**break**;

}

character.levelUp();

} **else** **if** (menu == 4) {

stw.study(character);

**if** (!character.checkEnergy()) {

System.***out***.println("에너지가 부족해서 캐릭터가 사망했습니다!");

**break**;

}

character.levelUp();

} **else** **if** (menu == 5) {

System.***out***.println("게임을 종료합니다.!! ");

**break**;

} **else** {

System.***out***.println("잘못 입력하셨습니다!");

**continue**;

}

**if** (character.level <= 4)

character.printInfo();

}

}

}

[ Ma ]

**package** student;

**public** **class** Ma **extends** Character {

**public** Ma() {

exp = 0;

energy = 50;

level = 1;

System.***out***.println("마씨 캐릭터가 생성되었습니다.");

printInfo();

}

**public** **void** eatkfood() {

energy += 5;

}

**public** **void** eatcfood() {

energy += 5;

}

**public** **void** eatjfood() {

energy += 10;

}

**public** **void** eatffood() {

energy += 5;

}

**public** **void** eatelse() {

energy += 10;

}

**public** **void** sleep() {

energy += 30;

}

**public** **boolean** loseDeathGame() {

energy -= 20;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** winDeathGame() {

energy -= 5;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectHurt() {

energy -= 20;

exp += 5;

**return** checkEnergy();

}

**public** **boolean** selectRefresh() {

energy += 25;

exp += 5;

**return** checkEnergy();

}

**public** **boolean** selectBulkup() {

energy += 35;

exp += 5;

**return** checkEnergy();

}

**public** **boolean** project() {

energy -= 20;

exp += 55;

**return** checkEnergy();

}

**public** **boolean** manyHomework() {

energy -= 15;

exp += 50;

**return** checkEnergy();

}

**public** **boolean** competition() {

energy -= 25;

exp += 55;

**return** checkEnergy();

}

**public** **boolean** study() {

energy -= 5;

exp += 50;

**return** checkEnergy();

}

**public** **boolean** winminigame() {

energy += 10;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** loseminigame() {

energy -= 10;

exp += 10;

**return** checkEnergy();

}

**public** **void** levelUp() {

**if** (exp >= 40) {

LevelupTest test = **new** LevelupTest();

test.LevelupTestWith(level);

**if** (test.*levelup* == **true**) {

level++;

exp = 0;

} **else** {

exp -= 30;

}

}

}

}

[ Han ]

**package** student;

**public** **class** Han **extends** Character {

**public** Han() {

exp = 0;

energy = 50;

level = 1;

System.***out***.println("한씨 캐릭터가 생성되었습니다.");

printInfo();

}

**public** **void** eatkfood() {

energy += 10;

}

**public** **void** eatcfood() {

energy += 5;

}

**public** **void** eatjfood() {

energy += 10;

}

**public** **void** eatffood() {

energy += 5;

}

**public** **void** eatelse() {

energy += 5;

}

**public** **void** sleep() {

energy += 30;

}

**public** **boolean** loseDeathGame() {

energy -= 30;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** winDeathGame() {

energy -= 10;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectHurt() {

energy -= 30;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectRefresh() {

energy += 15;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectBulkup() {

energy += 25;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** project() {

energy -= 20;

exp += 20;

**return** checkEnergy();

}

**public** **boolean** manyHomework() {

energy -= 15;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** competition() {

energy -= 25;

exp += 30;

**return** checkEnergy();

}

**public** **boolean** study() {

energy -= 15;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** winminigame() {

energy += 10;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** loseminigame() {

energy -= 10;

exp += 10;

**return** checkEnergy();

}

**public** **void** levelUp() {

**if** (exp >= 40) {

LevelupTest test = **new** LevelupTest();

test.LevelupTestWith(level);

**if** (test.*levelup* == **true**) {

level++;

exp = 0;

} **else** {

exp -= 30;

}

}

}

}

[ Yoon ]

**package** student;

**public** **class** Yoon **extends** Character {

**public** Yoon() {

exp = 20;

energy = 30;

level = 1;

System.***out***.println("윤씨 캐릭터가 생성되었습니다.");

printInfo();

}

**public** **void** eatkfood() {

energy += 5;

}

**public** **void** eatcfood() {

energy += 5;

}

**public** **void** eatjfood() {

energy += 5;

}

**public** **void** eatffood() {

energy += 10;

} // 양식 좋아!!

**public** **void** eatelse() {

energy += 10;

} // 간식 좋아!!

**public** **void** sleep() {

energy += 30;

}

**public** **boolean** loseDeathGame() {

energy -= 10;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** winDeathGame() {

energy -= 5;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectHurt() {

energy -= 10;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectBulkup() {

energy += 15;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** selectRefresh() {

energy += 10;

exp += 15;

**return** checkEnergy();

}

**public** **boolean** project() {

energy -= 25;

exp += 30;

**return** checkEnergy();

} // 프로젝트는 힘들어요.

**public** **boolean** manyHomework() {

energy -= 25;

exp += 20;

**return** checkEnergy();

} // 숙제도 힘들어요.

**public** **boolean** competition() {

energy -= 35;

exp += 30;

**return** checkEnergy();

} // 공모전도 힘들어요.

**public** **boolean** study() {

energy -= 20;

exp += 10;

**return** checkEnergy();

} // 그냥 공부도 힘들어요.

**public** **boolean** winminigame() {

energy += 10;

exp += 10;

**return** checkEnergy();

}

**public** **boolean** loseminigame() {

energy -= 10;

exp += 10;

**return** checkEnergy();

}

**public** **void** levelUp() {

**if** (exp >= 40) {

LevelupTest test = **new** LevelupTest();

test.LevelupTestWith(level);

**if** (test.*levelup* == **true**) {

level++;

exp = 0;

} **else** {

exp -= 30;

}

}

}

}

[ EatMeal ]

**package** student;

**import** java.util.Scanner;

**public** **class** EatMeal {

**public** **void** EatMealWith(Character character) {

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("어떤 음식을 먹을까요?");

System.***out***.print("1. 한식\t2. 중식\t3. 일식\t4. 양식\t5. 간식▶");

**int** eat = Integer.*parseInt*(in.nextLine());

**if** (eat == 1)

character.eatkfood();

**else** **if** (eat == 2)

character.eatcfood();

**else** **if** (eat == 3)

character.eatjfood();

**else** **if** (eat == 4)

character.eatffood();

**else**

character.eatelse();

}

}

[ PlayWith ]

**package** student;

**import** java.util.Scanner;

**public** **class** PlayWith {

**public** **void** playGameWith(Character character) {

Scanner scanner = **new** Scanner(System.***in***);

Drink drink = **new** Drink();

MiniGame minigame = **new** MiniGame();

LadderGame laddergame = **new** LadderGame();

System.***out***.println("무엇을 하면서 놀까요?");

System.***out***.print("1. 술 마시기\t2.미니 게임하기\t3. 운동하기▶");

**int** play = Integer.*parseInt*(scanner.nextLine());

**if** (play == 1) {

drink.DrinkGame(character);

} **else** **if** (play == 2) {

minigame.MiniGameWith(character);

} **else** **if** (play == 3) {

laddergame.*LadderGameWith*(character);

}

}

}

[ Drink ]

package student;

import java.util.Random;

import java.util.Scanner;

import java.util.concurrent.TimeUnit;

public class Drink {

public void DrinkGame(Character character) {

Scanner in = new Scanner(System.in);

Random random = new Random();

System.out.println("The Game of Death 시작~!\n");

System.out.println(">> 게임의 룰 설명<< ");

System.out.println(" 1. 먼저 참여할 인원을 입력하고 본인의 번호를 지정합니다.\n 2. 시작은 항상 0번부터 합니다.\n "

+ "3. 본인이 지목되면 다음으로 지목할 번호를 입력합니다.\n 4.마지막에 지목된 사람이 본인이 아니면 게임에서 승리합니다.\n ");

System.out.println("숫자(5 ~ 10)을 입력하시오.");

int N = in.nextInt();

System.out.println("자신이 위치할 번호(1 ~ 참여인원수)를 지정하시오.");

int ME = in.nextInt();

int bound = N, count = 1, index = 0;

int[] people = new int[N + 1];

// 첫 번째 지목 시작

System.out.println("첫번째 0번인 사람부터 지목 시작합니다.");

int first = random.nextInt(bound);

try {

if (first == ME) {

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("========================================");

System.out.println("0번이 당신을 지목하였습니다. 다음으로 지목할 번호를 입력하세요.");

int num = in.nextInt();

System.out.println("========================================");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(ME + "번이 " + num + "번을 지목하였습니다.");

people[index] = num;

} else {

System.out.println("첫번째 사람이 " + first + "번을 지목하셨습니다.");

people[index] = first;

}

} catch (Exception e) {

System.out.println(e);

}

// 첫번째 지목 이후 while문으로 N만큼 반복

while (true) {

if (count == N) {

System.out.println("마지막으로 지목되신 분은 " + people[index] + "번 입니다.");

if (people[index] == ME) {

System.out.println("당신은 죽음에 당첨되셨습니다. 술이들어간다 쭉~쭉~쭉쭉!! \n체력이 급격하게 떨어집니다.");

character.loseDeathGame();

} else {

System.out.println("당신은 운 좋게도 게임에 승리하셨습니다. 경험치가 오릅니다.");

character.winDeathGame();

}

break;

}

try {

if (count == 1) {

int next = random.nextInt(bound);

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(people[index] + "번이 " + next + "번을 지목하셨습니다.");

index++;

people[index] = next;

count++;

} else {

int next = random.nextInt(bound);

if (next == ME & count != N) {

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("========================================");

System.out.println(people[index] + "번이 당신을 지목하였습니다. 다음으로 지목할 번호를 입력하세요.");

int num = in.nextInt();

System.out.println("========================================");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(ME + "번이 " + num + "번을 지목하였습니다.");

count++;

index++;

people[index] = num;

} else if (count != N) {

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(people[index] + "번이 " + next + "번을 지목하셨습니다.");

count++;

index++;

people[index] = next;

}

}

} catch (Exception e) {

System.out.println(e);

}

}

}

}

[ MiniGame ]

package student;

import java.util.Random;

import java.util.Scanner;

public class MiniGame {

public void MiniGameWith(Character character) {

Random random = new Random();

Scanner in = new Scanner(System.in);

int rock = 1, sissor = 2, paper = 3;

System.out.println("[가위 바위 보 게임을 시작합니다.]\n");

System.out.println("무엇을 내시겠습니까 (바위 = 1 / 가위 = 2 / 보 = 3 )>> ");

while (true) {

int ME = in.nextInt();

int COM = random.nextInt(3);

if (ME == COM)

System.out.println("비겼습니다. 게임을 다시 진행합니다.");

else if (ME == 1) {

if (COM == 2) {

System.out.println("[플레이어]:바위 / [컴퓨터]:가위 \n 이겼습니다.!!");

character.winminigame();

break;

} else if (COM == 3) {

System.out.println("[플레이어]:바위 / [컴퓨터]:보 \n 졌습니다.ㅜㅜ 다음기회를 노려주세요!!");

character.loseminigame();

break;

}

} else if (ME == 2) {

if (COM == 1) {

System.out.println("[플레이어]:가위 / [컴퓨터]:바위 \n 졌습니다.ㅜㅜ 다음기회를 노려주세요!!");

character.loseminigame();

break;

} else if (COM == 3) {

System.out.println("[플레이어]:가위 / [컴퓨터]:보 \n 이겼습니다.!!");

character.winminigame();

break;

}

} else if (ME == 3) {

if (COM == 1) {

System.out.println("[플레이어]:보 / [컴퓨터]:주먹 \n 이겼습니다.!!");

character.winminigame();

break;

} else if (COM == 2) {

System.out.println("[플레이어]:보 / [컴퓨터]:가위 \n 졌습니다.ㅜㅜ 다음기회를 노려주세요!!");

character.loseminigame();

break;

}

}

}

}

}

[ LadderGame ]

package student;

import static student.Sadari.FINAL\_LINE;

import static student.Sadari.HEIGHT;

import static student.Sadari.SHAPE\_ARR;

import static student.Sadari.STANDARD\_LINE;

import java.util.Scanner;

public class LadderGame {

private static Scanner sc = new Scanner(System.in);

static String[] user\_s = new String[3];

static String[] result\_s = new String[3];

public static void LadderGameWith(Character character) {

int playNum = 0;

System.out.println("\n>>계단 오르기 게임의 룰<< ");

System.out.println(" 1. 3가지의 계단중 하나를 고릅니다.\n 2. 선택한 계단에 따라 [근성장,다침,개운함] 3가지 결과중 하나가 랜덤으로 결정됩니다.\n");

while (true) {

try {

playNum = 3;

System.out.println("A B C 다음 계단 중 하나를 고르시오.");

} catch (Exception e) {

System.out.println(e.getMessage());

continue;

}

String[][] sadariField = new String[HEIGHT][playNum \* FINAL\_LINE];

String[] user = new String[playNum];

String[] result = new String[playNum];

setField(sadariField, playNum);

setInput(user, "플레이어");

setResult(result);

setBridge(sadariField, playNum);

result(sadariField, playNum, user, result);

for (int i = 0; i < 3; i++) {

if (user\_s[i].equals("플레이어")) {

if (result\_s[i].equals("다침"))

character.selectHurt();

else if (result\_s[i].equals("개운함"))

character.selectRefresh();

else if (result\_s[i].equals("근성장"))

character.selectBulkup();

}

}

break;

}

/// 나온 결과 가지고 if 문으로 추가하기

}//////////// 메인 메서드

private static void setField(String[][] arr, int num) {

for (int q = 0; q < HEIGHT; q++) {

for (int w = 0; w < FINAL\_LINE \* num; w++) {

if (w % 4 == STANDARD\_LINE - 1) {

arr[q][w] = SHAPE\_ARR[0];

System.out.print(SHAPE\_ARR[0]);

} else {

arr[q][w] = " ";

System.out.print(" ");

}

}

System.out.println();

}

System.out.println("다침 근성장 개운함 계단타기의 결과");

}// 사다리를 설정하는 메서드

private static void setInput(String[] userArr, String msg) {

String select;

System.out.println("선택할 계단을 입력하세요: ");

select = sc.next();

if (select.equals("A")) {

System.out.println("당신이 선택한 계단은: " + select);

userArr[0] = "플레이어";

userArr[1] = "B";

userArr[2] = "C";

} else if (select.equals("B")) {

System.out.println("당신이 선택한 계단은: " + select);

userArr[0] = "A";

userArr[1] = "플레이어";

userArr[2] = "C";

} else if (select.equals("C")) {

System.out.println("당신이 선택한 계단은: " + select);

userArr[0] = "A";

userArr[1] = "B";

userArr[2] = "플레이어";

}

}

private static void setResult(String[] userArr) {

userArr[0] = "다침";

userArr[1] = "근성장";

userArr[2] = "개운함";

}

// 결과값을 저장하는 메소드를 하나 만들어야 함

private static void setBridge(String[][] arr, int num) {

int lineNumber = arr[0].length / FINAL\_LINE;

int[] checkLine = new int[lineNumber];

for (int q = 1; q < arr.length - 1; q++) {

for (int w = 0; w < lineNumber - 1; w++) {

if ((int) (Math.random() \* 2) == 0)

checkLine[w] += checkLine[w] < 3 ? setBridgeDetail(arr, q, w \* FINAL\_LINE) : 0;

}

if (q == arr.length - 2)

for (int w = 0; w < lineNumber - 1; w++) {

if (checkLine[w] == 0) {

q = 1;

break;

}

}

}

} // 안고쳐도 되고 - 사다리에 연결다리 설정

private static int setBridgeDetail(String[][] sadari, int q, int w) {

if (sadari[q][w].equals(SHAPE\_ARR[0]) && sadari[q][w + FINAL\_LINE].equals(SHAPE\_ARR[0])) {

sadari[q][w] = SHAPE\_ARR[3];

for (int e = 1; e < FINAL\_LINE + 1; e++) {

if (e < FINAL\_LINE) {

sadari[q][w + e] = SHAPE\_ARR[1];

} else {

sadari[q][w + e] = SHAPE\_ARR[2];

}

}

return 1;

}

return 0;

} // 안고쳐도 되고 - 사다리에 연결다리 디테일 설정

private static void result(String[][] arr, int sadariNum, String[] user, String[] result) {

realTimeScreenDraw(arr, "");

String tmp = "";

int num = 0;

for (int q = 0; q < arr[0].length; q += FINAL\_LINE) {

int x = 0, y = q;

for (int w = 0; w < arr.length; w++) {

if (arr[x][y].equals(SHAPE\_ARR[3]))

y += FINAL\_LINE;

else if (arr[x][y].equals(SHAPE\_ARR[2]))

y -= FINAL\_LINE;

// "♥"이동 로직

tmp = arr[x][y];

arr[x][y] = "♥";

realTimeScreenDraw(arr, user[q / FINAL\_LINE]);

waitTime(250);

arr[x][y] = tmp;

x++;

}

System.out.println(user[q / FINAL\_LINE] + "의 결과는 " + result[y / FINAL\_LINE]);

user\_s[num] = user[q / FINAL\_LINE];

result\_s[num] = result[y / FINAL\_LINE];

user[q / FINAL\_LINE] += "의 결과 -> " + result[y / FINAL\_LINE];

waitTime(1500);

num++;

}

blank();

System.out.println("────── 결과창 ──────");

for (String temp : user) {

System.out.printf("%s %n", temp);

}

} /// 얘를 좀 고쳐야 할 것.

private static void realTimeScreenDraw(String[][] arr, String user) {

blank();

System.out.println("현재 유저 -> " + user);

for (int q = 0; q < arr.length; q++) {

for (int w = 0; w < arr[q].length; w++) {

System.out.print(arr[q][w]);

}

System.out.println();

}

} // 실시간으로

private static void waitTime(int waitNum) {

try {

Thread.sleep(waitNum);

} catch (InterruptedException e) {

e.printStackTrace();

}

} // 안바꿔도 되고

private static void blank() {

System.out.printf("%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n%n");

}

}// 바꾸면 안되고

class Sadari {

public static final String[] SHAPE\_ARR = { "│", "──", "┤", "├" };

public static final int STANDARD\_LINE = 1;

public static final int BRIDGE\_LINE = 3;

public static final int FINAL\_LINE = STANDARD\_LINE + BRIDGE\_LINE;

public static final int HEIGHT = 8;

}// 바꾸면 안되고

[ StudyWith ]

package student;

import java.util.Random;

import java.util.concurrent.TimeUnit;

public class StudyWith {

public void study(Character character) {

Random random = new Random();

int iValue = random.nextInt(5); // 0 ~ 5 사이의 난수 발생

if (iValue == 0) {

try {

System.out.println("프로젝트 수행에 참여하게 되셨습니다. 최선을 다해 임해주세요!");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("프로젝트 진행중");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("프로젝트를 완료하였습니다.!!");

character.project();

} catch (Exception e) {

System.out.println(e);

}

} else if (iValue == 2) {

try {

System.out.println("과제 폭탄입니다....ㅜㅜ");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("과제 진행중");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("과제를 완료하였습니다.!!");

character.manyHomework();

} catch (Exception e) {

System.out.println(e);

}

} else if (iValue == 4) {

try {

System.out.println("공모전에 참여하십시오. 경험을 쌓는데 많은 도움이 됩니다.!! ");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("공모전 진행중");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("공모전을 완료하였습니다.!! 결과와 관계없이 경험치가 오릅니다.");

TimeUnit.MILLISECONDS.sleep(500);

character.competition();

} catch (Exception e) {

System.out.println(e);

}

} else {

try {

System.out.println("본인의 개인공부를 시작합니다.");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("공부중");

TimeUnit.MICROSECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println(" .");

TimeUnit.MILLISECONDS.sleep(500);

System.out.println("공부를 완료하였습니다.!!");

character.study();

} catch (Exception e) {

System.out.println(e);

}

}

}

}

[ LevelupTest ]

**package** student;

**import** java.util.Scanner;

**public** **class** LevelupTest {

**static** **boolean** *levelup* = **false**;

**public** **void** LevelupTestWith(**int** level) {

Scanner in = **new** Scanner(System.***in***);

**switch** (level) {

**case** 1:

System.***out***.println("[1학년 진급시험을 시작합니다.]\n 다음 질문에 답해주세요.");

System.***out***.println();

System.***out***.println("질문) 현재 배우는 객체지향프로그램의 담당 교수님의 성함을 띄어쓰기 없이 입력하시오.");

System.***out***.println("==> 정답 : ");

**while** (**true**) {

String answer = in.next();

**if** (answer.equalsIgnoreCase("김형균")) {

*levelup* = **true**;

System.***out***.println("2학년으로 진급하셨습니다. 축하드립니다!! ");

**break**;

} **else** {

System.***out***.println("틀리셨습니다. 1학년으로 유급됩니다.");

*levelup* = **false**;

**break**;

}

}

**break**;

**case** 2:

System.***out***.println("[2학년 진급시험을 시작합니다.]\n 다음 질문에 답해주세요.");

System.***out***.println();

System.***out***.println("질문) 김형균 교수님의 메일주소를 입력하시오.");

System.***out***.print("==> 정답 : ");

**while** (**true**) {

String answer = in.next();

**if** (answer.equalsIgnoreCase("multikim@kookmin.ac.kr")) {

*levelup* = **true**;

System.***out***.println("3학년으로 진급하셨습니다. 축하드립니다!! ");

**break**;

} **else** {

System.***out***.println("틀리셨습니다. 2학년으로 유급됩니다.");

*levelup* = **false**;

**break**;

}

}

**break**;

**case** 3:

System.***out***.println("[3학년 진급시험을 시작합니다.]\n 다음 질문에 답해주세요.");

System.***out***.println();

System.***out***.println("질문) 2021년 1학기 객체지향프로그래밍 수업 시작 시간을 입력하시오. 시간 숫자만 입력하시오.");

System.***out***.print("==> 정답 : ");

**while** (**true**) {

**int** answer = in.nextInt();

**if** (answer == 12) {

*levelup* = **true**;

System.***out***.println("4학년으로 진급하셨습니다. 축하드립니다!! ");

**break**;

} **else** {

System.***out***.println("틀리셨습니다. 3학년으로 유급됩니다.");

*levelup* = **false**;

**break**;

}

}

**break**;

**case** 4:

System.***out***.println("[4학년 진급시험을 시작합니다.]\n 다음 질문에 답해주세요.");

System.***out***.println();

System.***out***.println("질문) 졸업소감을 써 주세요.");

System.***out***.print("==> 정답 : ");

String answer = in.next();

*levelup* = **true**;

System.***out***.println("졸업하셨습니다. 그동안 수고하셨습니다.");

**break**;

}

}

}