객체지향 프로젝트 개발 계획서

임재학의 다이어트 프로젝트

신 분	성 명		소 속
팀 원	124897	임재학	전남대학교
	122128	최영수	
	112283	이승우	
	111768	정혜원	

1. 목표 어플리케이션 이름 : “임재학의 다이어트 프로젝트”

2. 개발 동기

가장 흥미로운 주제를 선정 하던 중, 몇 개월 전에 유행하던 ‘살아남아라! 개복치’ 라는 게임을 생각하게 되었다. ‘살아남아라! 개복치’ 라는 게임은 제목 그대로 개복치를 키우는 게임이지만, 개복치 특성 중 하나인 돌연사라는 소재를 갖고 사용자들에게 어이없는 웃음을 선사하며 동시에 중독성을 갖게 하여 몇 주 동안 1위까지 한 게임이다. 이 주제를 가지고 우리 주위의 사람이랑 연관을 시켜보면 정말 재미있겠다고 생각하였다. 그래서 팀원 중 한명인 임재학 학생을 주인공으로 정하여 게임을 개발하기로 하였다.

3. 제공할 기능

1. 초기 화면 : 게임시작, 게임설명, 환경설정, 게임 종료 등 선택

2. 메인 화면 : 게임이 진행되는 배경. 캐릭터의 상태를 볼 수 있는 곳이다. 체중, 아이템 인벤토리, 기분, 스케줄 등을 볼 수 있는 곳이다. 메인 화면을 움직이면 NPC를 만나게 된다.

3. 초기화 : 사용자가 게임을 진행하던 중 새롭게 시작하고 싶을 경우 초기화 버튼을 통해서 새로 시작할 수 있다.

4. 체중 시스템 : 캐릭터의 체중 상태를 볼 수 있다. 칼로리 섭취, 소모를 계산하여 체중이 소수점 둘째 자리까지 보인다. 게임의 목표가 체중 감량을 하는 것이니만큼 체중을 잘 관리하는 것이 게임의 성패를 가른다.

5. 기분 게이지 시스템 : 캐릭터의 감정 상태를 볼 수 있다. NPC의 유혹을 계속 거절할 경우 기분 게이지가 떨어진다. 이 때 증감되는 양은 고정된 것이 아니라 계속 변하여 사용자가 예측할 수

없게 함으로써 흥미를 유발한다. 기분 게이지가 끝까지 떨어지면 캐릭터는 우울증에 걸리게 되고 게임은 실패하게 된다.
시간을 정하여 증감되는 양이 랜덤으로 바뀌게 하고 사용자는 증감되는 양은 구체적으로 모르지만, 기분 이모티콘을 기분 게이지 옆에 두어 증감되는 양이 변한다는 것은 알게 한다. 예를 들어 행복한 이모티콘이 있을 때는 기분 변수가 1.5가 되어 기분이 증가할 때는 (원래 증가하는 양 * 1.5)으로 증가하고 감소할 때는 (원래 증가하는 양 * (2-1.5))로 감소하게 된다.

6. 스케줄 시스템 : 게임 내에서 캐릭터는 매일 스케줄을 받게 된다. 사용자가 선택하는 것이 아닌, 사전에 저장해 놓은 스케줄 중 하나가 임의로 선택되어 사용자에게 보여진다. 사용자는 매일 주어진 스케줄을 모두 수행해야 그 날 하루가 끝나게 된다.

7. 미니게임 : 사용자가 스케줄대로 게임을 진행하거나 특정 NPC를 만날 경우 할 수 있는 미니게임을 제공한다. 사용자의 실력을 요하는 게임과 실력과 무관한 게임 두 종류의 게임이 임의로 나온다. 미니게임을 성공할 경우 게임의 종류에 따라 캐릭터는 체중이 줄어들거나 아이템을 얻게 된다.

8. 아이템 시스템 : 캐릭터가 게임 내에서 모으는 아이템을 나타낸다. 사용자가 현재 자신이 갖고 있는 아이템을 볼 수 있으며 사용하고 싶을 때 자유롭게 쓸 수 있다. 아이템의 종류는 여러 종류가 있으며 감정에 관련된 아이템, 체중에 관련된 아이템 등이 있다. 예를 들면 한 번에 체중 감량하게 해주는 약이라던지, 기분 게이지가 바닥까지 닿아도 회복할 수 있는 우울증 치료제 등이 있다.

4. 요구사항 리스트

4-1) 요구사항 리스트

1. 몰입도를 많이 요하는 게임이 아니라 킬링타임 용으로 쓸 수 있는 게임이었으면 좋겠다.
2. 난이도나 특별한 스킬이 필요한 게임이 아니라 남녀노소 편히 할 수 있는 게임이었으면 좋겠다.
3. 다이어트의 경각심을 일깨워줬으면 좋겠다.
4. 실생활에서 자주 접할 수 있는 상황들을 배치하여 리얼리티가 있었으면 좋겠다.
5. 게임의 주인공을 정할 수 있게 하여 친근감을 높인다.
6. 게임의 배경을 학교 안으로 하여 친근감을 높인다.
7. 사용자로 하여금 지루하지 않게 게임 속의 시간과 스토리의 진행이 빨리 흐르게 한다.
8. 게임이 진행됨에 따라 레벨이 높아지면 캐릭터의 외형도 변하게 하여 사용자로 하게끔 흥미를 잃지 않게 한다.

4-2) 요구사항 기술서

요즘 남녀노소를 불문하고 많은 사람들이 자신을 가꾸기 위하여 많은 신경을 쓴다. 그중 가장 관심이 높은 것은 단연 다이어트라고 할 수 있다. 이게임은 주변에서 흔히 볼 수 있는 다이어트를 하는 사람을 주인공으로 설정하여, 그의 다이어트 과정을 보여준다. 게임의 주인공과 배경을 주변에서 흔히 접할 수 있는 것들로 선정하여 친근감을 높이고, 몰입도를 높인다. 또, 다이어트를 하는 과정을 직접 눈으로 확인하며 대리만족을 느낄 수 있다. 이 게임을 통하여 자신의 평소 식습관과 생활방식을 돌아보고 다이어트에 대한 자극이 되었으면 한다.

게임을 시작하면 캐릭터 이름을 입력한다. '임재학'을 입력하면, 주인공의 이름은 '임재학'으로 입력되고, 주변 배경(공과대학 6호관,7호관 ,연구실, 집, 식당 등)은 기본 배경 으로 설정한다. 게임을 시작할 때, 현제 자신의 몸무게를 입력한다. 게임 진행 중 현재몸무게의 120%가 되었을 시 게임실패. 반대로 다이어트에 성공하여서 현재 몸무게의 80%가 되었을 때 레벨 업을 한다. 게임의 큰 틀은, 게임 시간으로, 하루에 한번 스케줄이 부여된다. 현실시간으로 10분이 게임에서의 하루로 부여된다. 매번 다른 이벤트들(수업, 운동, 약속 등)이 주인공에게 주어지게 되고 이벤트를 진행하며 랜덤으로 NPC가 생성된다. NPC 또한 우리주변에서 흔히 볼 수 있는 인물로 설정한다.

예를 들어 김경백 교수님, 친구, 연구실원, 여자 친구 등이 등장하게 되며, 그들은 메인 캐릭터의 다이어트 실패를 목표로 유혹을 한다, 주인공은 NPC의 유혹을 받아 들일건지 아닐지를 선택하는데 받아드리게 되면 상황에 따라서 몸무게가 증가하고. 거절할 시에는 체중이 유지됩니다. 하지만 게임의 재미와 균형을 위해 연속된 제안을 거절하면 '기분'이라는 수치가 일정하게 감소되어 많이 떨어졌을 경우 우울증으로 다시 살이 찌게 됩니다. 반대로 운동을 하게 되면 체중을 감량할 수 있다. 단, 체중의 감량치는 음식을 먹었을 때의 증가량보다 현저히 떨어진다. 매 이벤트들로 인한 체중 증가치는, 레벨이 상승함에 따라 줄어들고, 레벨이 상승하면 캐릭터의 외형이 바뀐다.

5. 세부 유즈 케이스

5.1) 초기화면

Main Path	Alternative Path
1. 프로그램을 실행 시킨다. 2. 메인화면이 나타난다. 3. 게임시작, 게임설명, 환경설정, 게임종료 버튼이 나타난다. 4. 4개의 버튼 중 1개의 버튼을 누른다.	3.1 게임시작 맵과 이동 가능한 캐릭터가 나온다. 3.2 게임 설명 미니게임, 캐릭터 조작법, 레벨 시스템 등 전반적인 게임에 대한 설명이 담겨 있다. 3.3 환경설정 게임 초기화 버튼과 사운드 조절 버튼이 나온다. 3.4 게임종료 프로그램이 종료된다.

5.2) 메인화면

Main Path	Alternative Path
1. 맵, 캐릭터, 체중, 기분게이지, 기분을 나타내는 이모티콘, 아이템 인벤토리, 스케줄이 나타난다. 2. 스케줄 창 팝업. 3. 스케줄대로 캐릭터를 움직인다. 4. NPC를 만난다. 5. NPC가 유혹한다.	5.1 유혹에 걸린다. 5.1.1 살이 쪘다는 창과 함께 체중게이지가 올라간다. 5.2 유혹을 거절한다. 5.1.2 체중이 유지되거나 감량 됐다는 창과 동시에 체중게이지가 내려간다. npc의 유혹을 세 번 이상 거절할 경우에는 캐릭터의 '기분'수치가 떨어져서 캐릭터가 우울증에 걸리게 되고 그로 인해 다시 체중이 증가하게 되므로 주의해야 한다. 5.3 게임 도중 ESC 키를 누르면 초기화면으로 돌아간다.

5.3) 아이템 시스템

(기존에는 도감 시스템이어서 도감에 나오는 음식을 모두 찾을 경우 아이템을 주는 방식이었으나 사용자들에게 게임의 이해도를 높이기 위해 도감 시스템을 없애고 아이템 시스템으로 통일하였다.)

Main Path	Alternative Path
1. 캐릭터가 NPC를 만나거나 스케줄대로 움직이다가 미니게임을 실행하게 된다. 2. 미니게임의 결과 성공하면 아이템을 얻게 된다.	1.1 미니게임이 실행 될 때의 경우 사용자가 원하는 미니게임을 선택하는 것이 아니라 게임에서 미니게임 중 하나를 임의로 시작하게 된다. 따라서 사용자는 어떤 게임을 하는지 예측할 수 없게 된다. 2.1 아이템을 얻게 되면 아이템 인벤토리에 새로 얻게 된 아이템이 추가되며 아이템의 종류는 다양하다. 예를 들어 체중을 감량시켜 주는 알약일 수도 있고 우울증 치료제 등 많은 아이템이 있다. 사용자는 인벤토리에서 자신이 보유하고 있는 아이템의 종류를 볼 수 있게 되고 또 사용하기를 원할 때 자유롭게 사용할 수도 있다.

5.4) 운동 시스템

Main Path	Alternative Path
1. 게임 중에 NPC를 만났을 때 NPC가 운동을 권유하거나 스케줄 대로 운동을 갔을 때 미니게임이 시작된다. 2. 운동(미니게임) 창이 뜬다. 3. 게임설명에 따라 게임을 시작한다. 4. 게임이 끝난 후 결과창이 뜬다.	1.1 미니게임을 거절할 수도 있다. 거절할 경우 체중의 변화는 없고 기분 게이지가 떨어진다. 승낙할 경우 게임의 결과에 따라 체중이 변하거나 기분 게이지가 올라간다. 3.1 미니게임에 성공한다. 3.1.1 체중이 감량되거나 아이템을 획득한다. 3.2 미니게임에 실패한다. 3.1.2 체중이 유지된다.

5.5) 초기화

Main Path	Alternative Path
1. 초기화 버튼을 누른다. 2. '확인'버튼과 '취소'버튼 중에서 선택한다. 3. 지금까지 누적된 캐릭터의 정보가 초기화된다. 따라서 사용자는 새로운 캐릭터로 게임을 진행하게 된다.	2.1 취소버튼을 누를시 사용자는 메인 화면으로 돌아가게 된다.

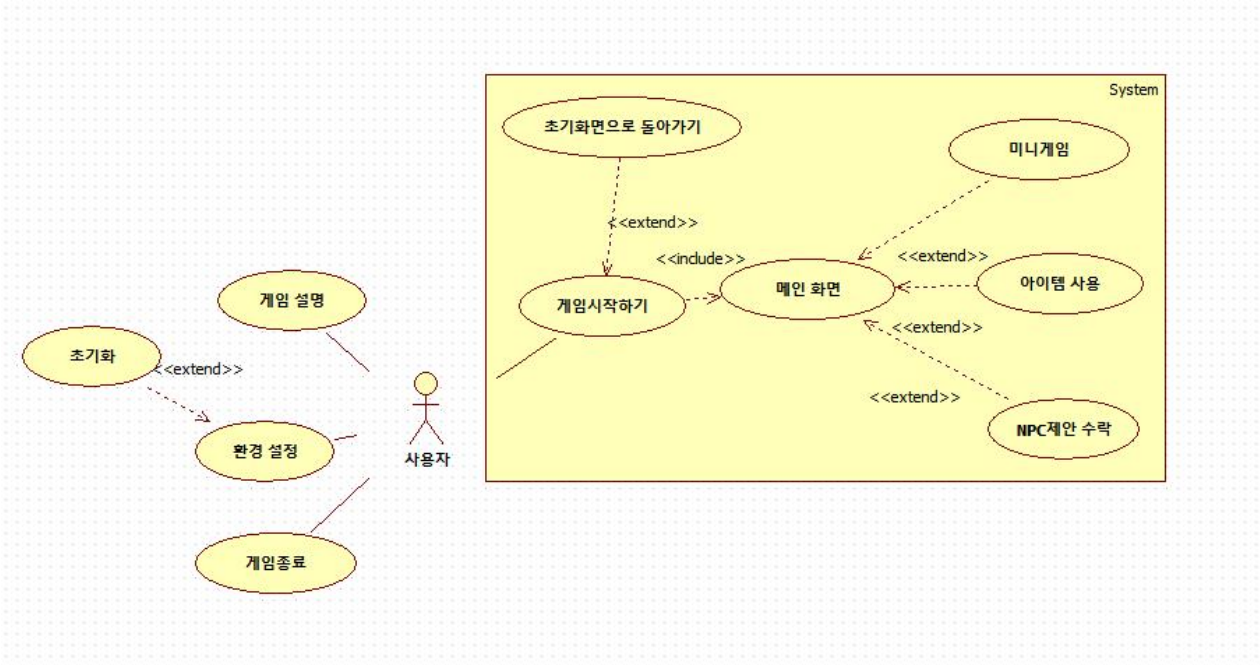
5.6) 체중 게이지 시스템

Main Path	Alternative Path
1. 체중에 따라 게이지 바가 움직인다.	1.1 게이지 바가 내려간다. 1.1.1 목표치에 도달하면 레벨이 올라간다. 1.2 게이지 바가 올라간다. 1.1.2 목표치에 도달하면 레벨이 내려간다.

5.6) 특수키

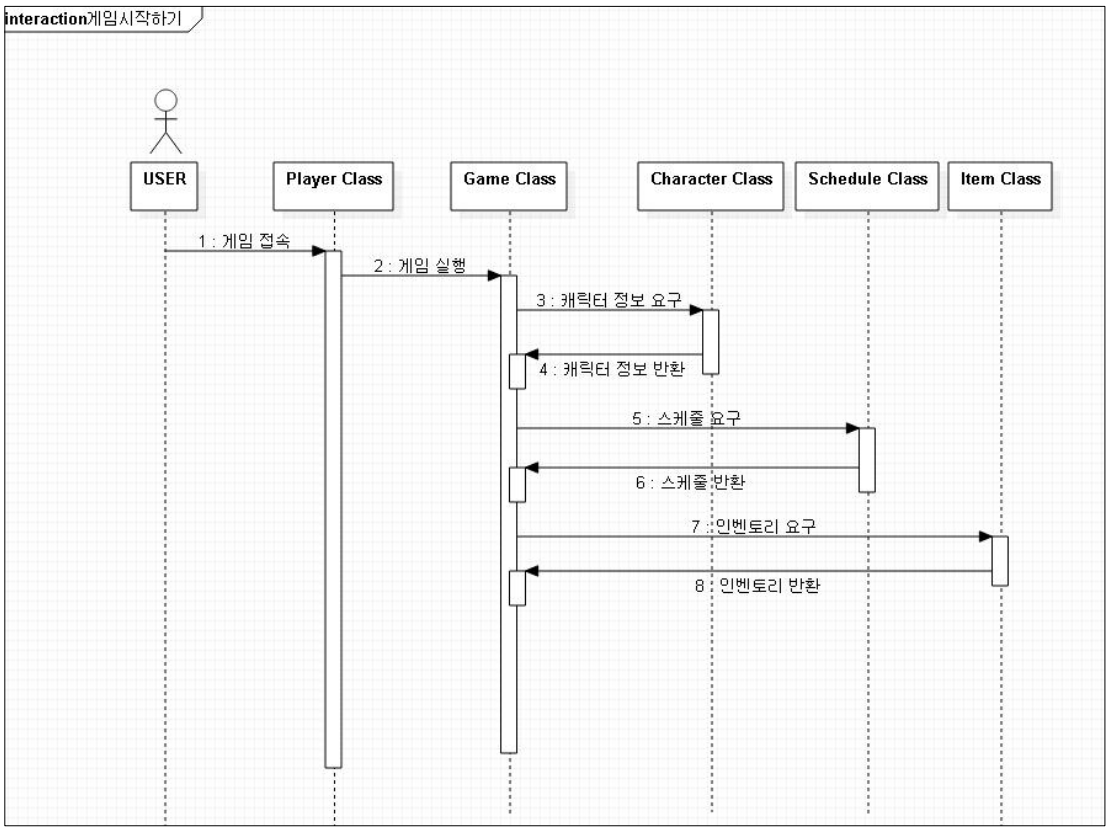
Main Path	Alternative Path
1. 게임 도중, 초기 화면이나 메인 화면에 상관 없이 ESC 키를 누를 경우 게임의 초기 화면으로 돌아간다.	1.1 초기 화면으로 돌아가면 게임 시작 버튼, 리셋 버튼, 게임 설명, 게임 종료 등 버튼이 있는데 사용자가 그 중 원하는 버튼을 누를 수 있다. 게임을 하는 도중 처음부터 시작하고 싶을 때는 ESC 키를 누른 후 초기화면으로 돌아가서 리셋 버튼을 누르면 된다.

6. 유즈케이스 다이어그램

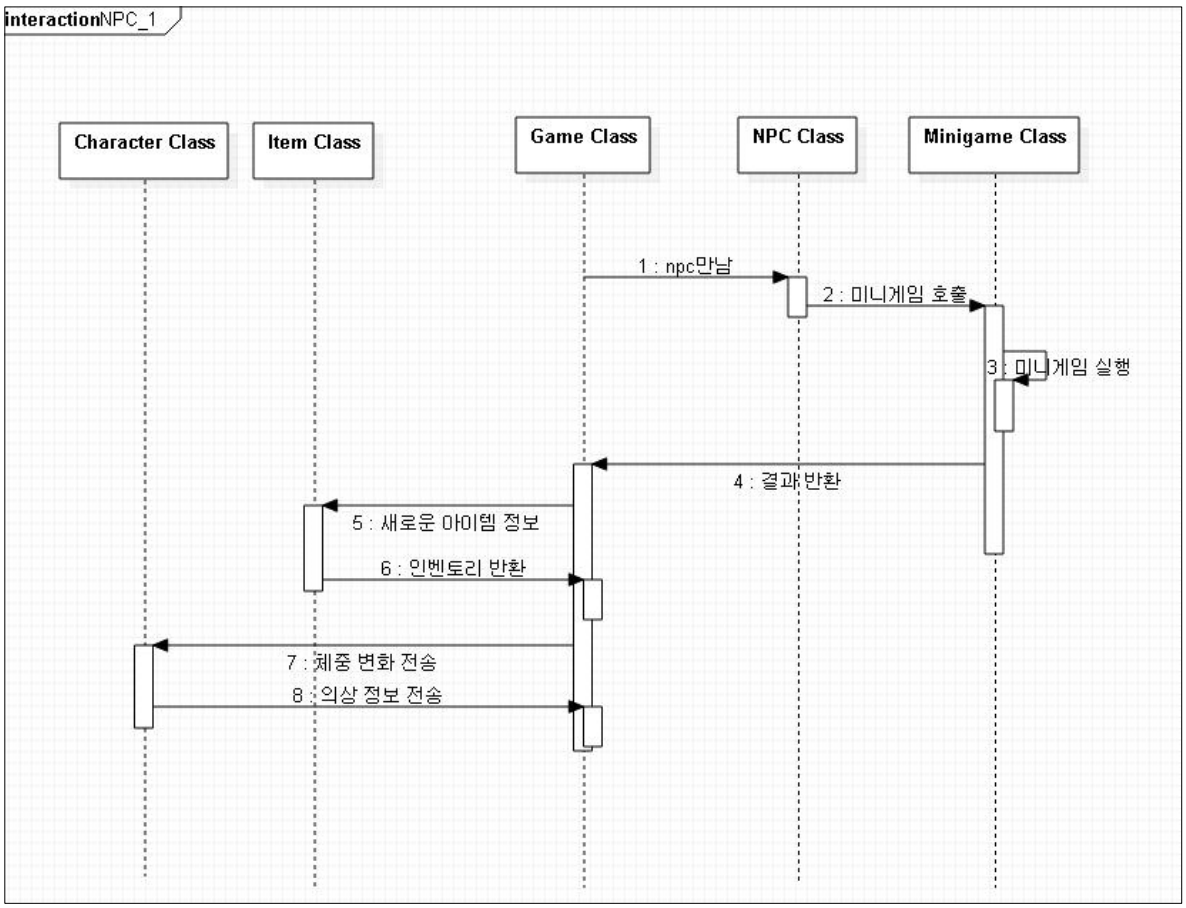


7. 시퀀스 다이어그램

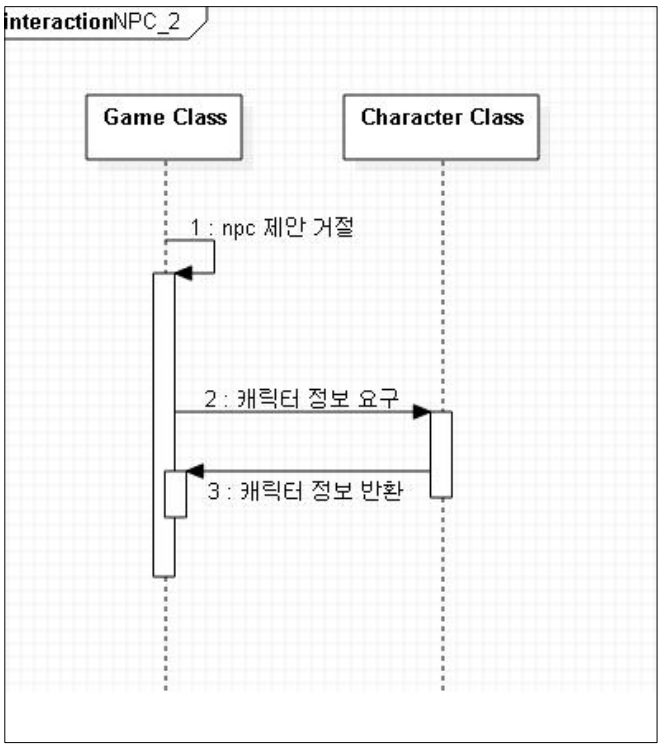
<사용자가 게임을 시작했을 때>



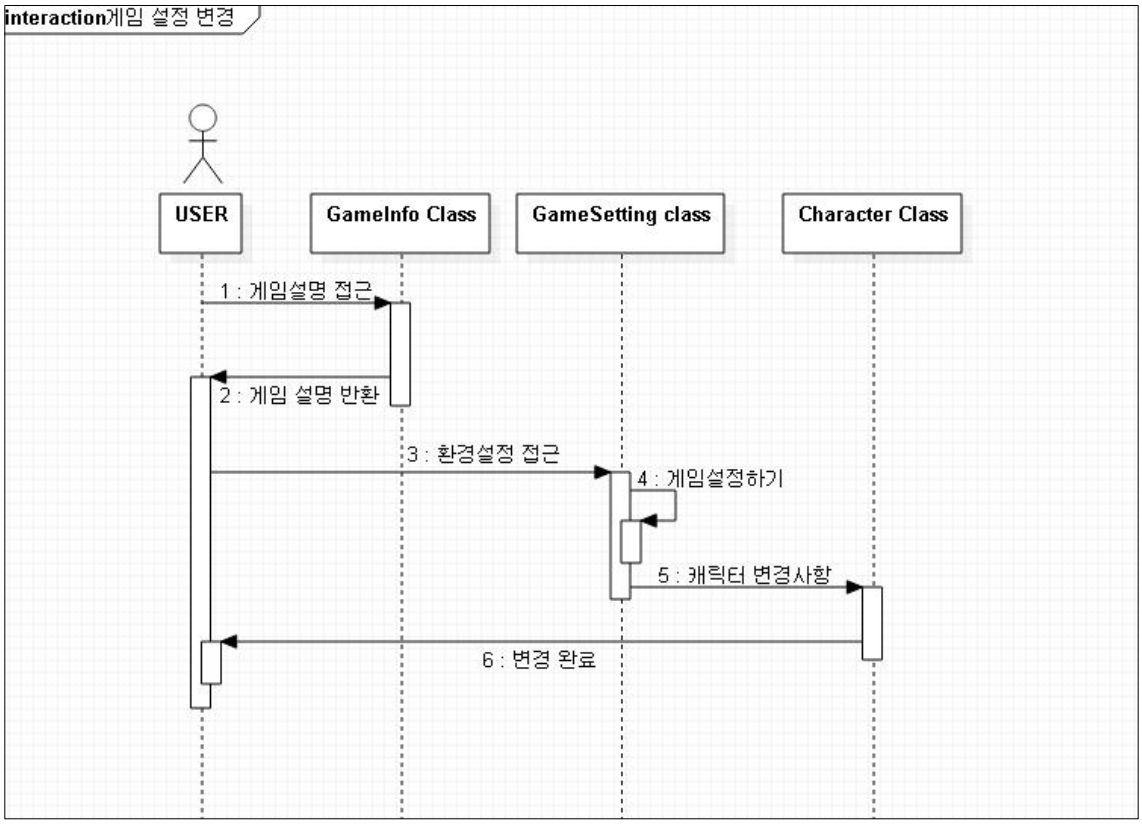
<NPC 제안 수락했을 때>


interactionNPC_1
Character Class
Item Class
Game Class
NPC Class
Minigame Class
1 : npc만남
2 : 미니게임 호출
3 : 미니게임 실행
4 : 결과 반환
5 : 새로운 아이템 정보
6 : 인벤토리 반환
7 : 체중 변화 전송
8 : 의상 정보 전송

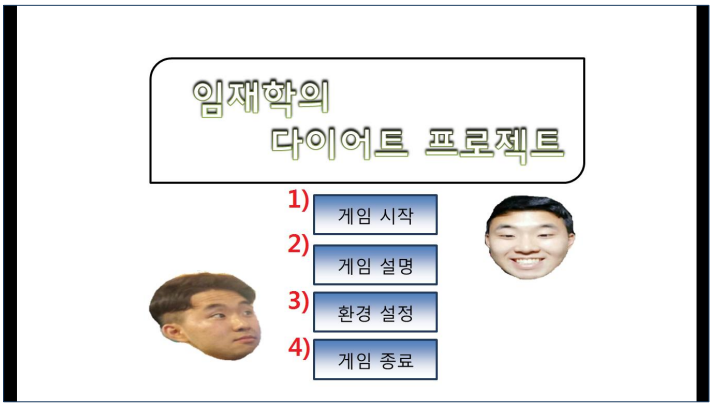
<NPC제안 거절했을 때>


interactionNPC_2
Game Class
Character Class
1 : npc 제안 거절
2 : 캐릭터 정보 요구
3 : 캐릭터 정보 반환

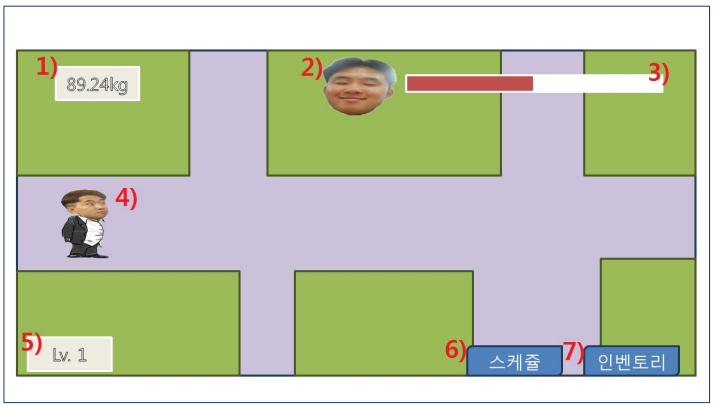
<게임 설정 변경>

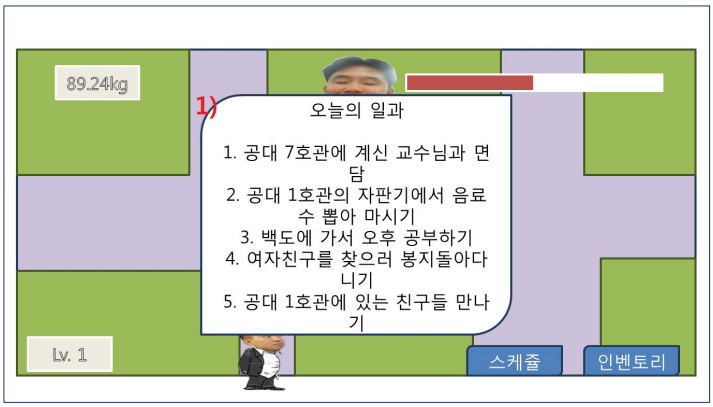

interaction게임 설정 변경
USER
GameInfo Class
GameSetting class
Character Class
1 : 게임설명 접근
2 : 게임 설명 반환
3 : 환경설정 접근
4 : 게임설정하기
5 : 캐릭터 변경사항
6 : 변경 완료

8. 화면 정의서

화면 ID	diet_001	화면 명	어플리케이션 초기 화면
화면 layout			설명
			1. 화면 설명 - 어플리케이션을 실행하면 나타나는 초기화면이다. 2. 주요화면 설명 1) 게임 시작- 메인 화면으로 이어진다. 맵과 이동 가능한 캐릭터가 나온다. 2) 게임 설명- 미니게임, 캐릭터 조작법, 레벨 시스템 등 전반적인 게임에 대한 설명이 담겨 있다. 3) 환경 설정- 게임 초기화 버튼과 사운드 조절 버튼이 나온다. 4) 게임 종료- 프로그램이 종료된다.

화면 ID	diet_002	화면 명	메인 화면

화면 layout	설명
	1. 화면 설명 - 게임을 실행하면 나오는 메인화면이다. 2. 주요화면 설명 1) 체중 수치- 게임 성공의 주요소인 체중이 나타난다. 2) 표정- 기분의 상태를 이미지로 보인다. 3) 기분 게이지- 기분 증감 상황을 게이지를 통해 나타낸다. 4) 캐릭터- 사용자가 움직일 수 있는 캐릭터이다. 5) 레벨- 사용자의 레벨을 보여준다. 6) 스케줄- '오늘의 일과' 창을 뜨게 한다. 7) 인벤토리- '인벤토리' 창을 뜨게 한다.

화면 ID	diet_003	화면 명	메인 화면 – 오늘의 일과
화면 layout		설명	
		1. 화면 설명 - 메인 화면의 '스케줄' 버튼을 누르면 나타나는 창이다. 2. 주요화면 설명 1) 스케줄 창- 사용자가 완수해야 하는 리스트가 담긴 '오늘의 일과' 창이다.	
화면 ID	diet_004	화면 명	메인 화면 – 인벤토리
화면 layout		설명	
		1. 화면 설명 - 메인 화면의 '인벤토리' 버튼을 누르면 나타나는 창이다. 2. 주요화면 설명 1) 인벤토리 창- 사용자가 보유하고 있는 아이템을 나타내는 '인벤토리' 창이다. 사용할 수 있다. 2,3) 아이템- 해당 아이템을 클릭하면 아이템에 대한 설명이 나온 후 사용자의 선택에 따라 캐릭터에 적용된다.	

화면 ID	diet_005	화면 명	메인 화면 - NPC
화면 layout			설명
			1. 화면 설명 - 게임을 진행하며 캐릭터를 움직이다가 NPC와 랜덤으로 만나게 됐을 때 나타나는 화면이다. 2. 주요화면 설명 1) 메시지 창- NPC를 만나게 되면 각 NPC에 해당하는 문구가 나오게 된다. 2) 선택- 사용자는 NPC의 제안을 수락하거나 거절을 선택하게 되고, 이로 인해 사용자의 기분이 영향을 받게 된다.

화면 ID	diet_006	화면 명	미니게임1 – 주사위 게임

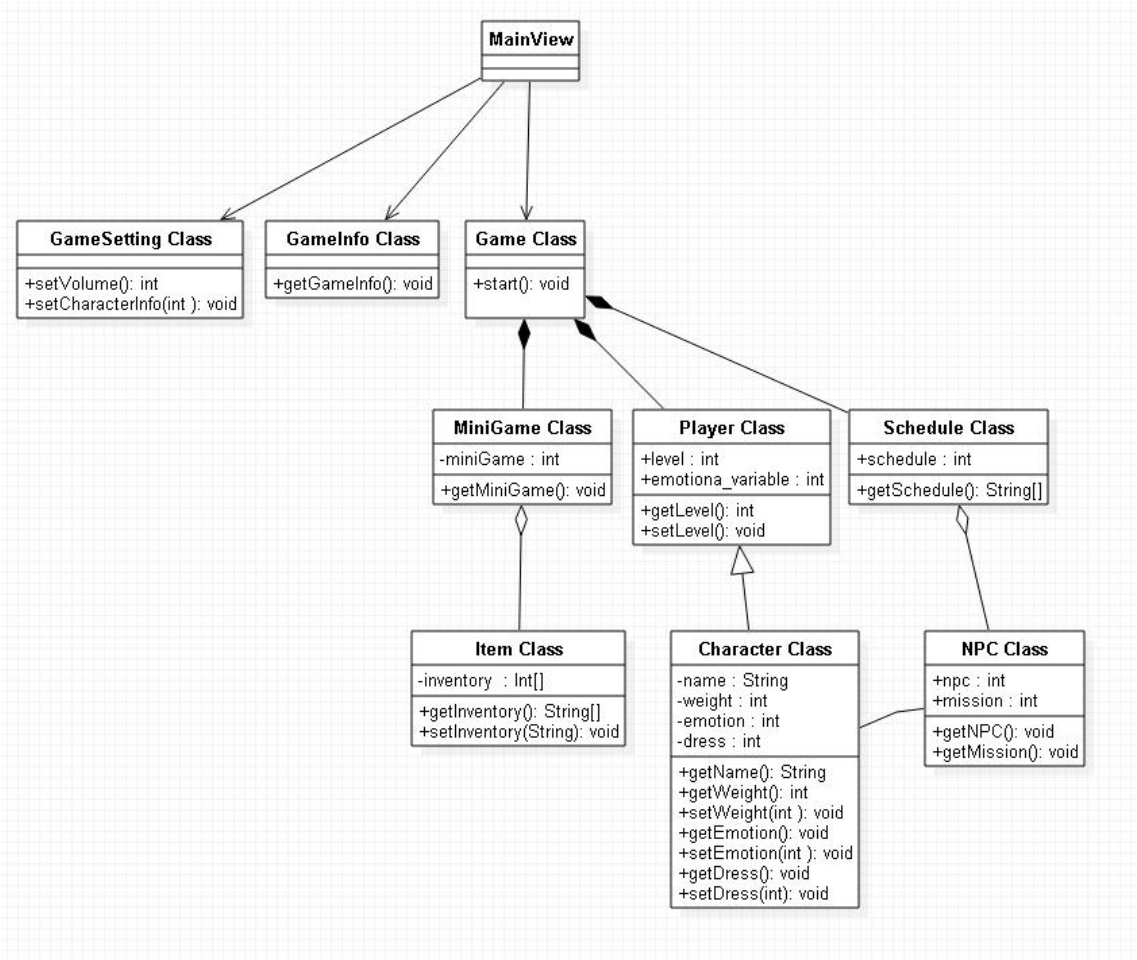
화면 layout	설명
	1. 화면 설명 - 게임 진행 중 나오는 임의의 미니게임이다. 2. 주요화면 설명 1) 제목- 미니게임의 종류가 나타난다. 2) 미션- 미니게임 한 종류 내에서도 여러가지 미션이 존재한다. 랜덤으로 선택되는 미션이 나타난다. 3) 게임 화면- 화면을 터치하면 게임이 시작되고 결과가 나타난다.

화면 ID	diet_007	화면 명	미니게임2 – 사다리타기

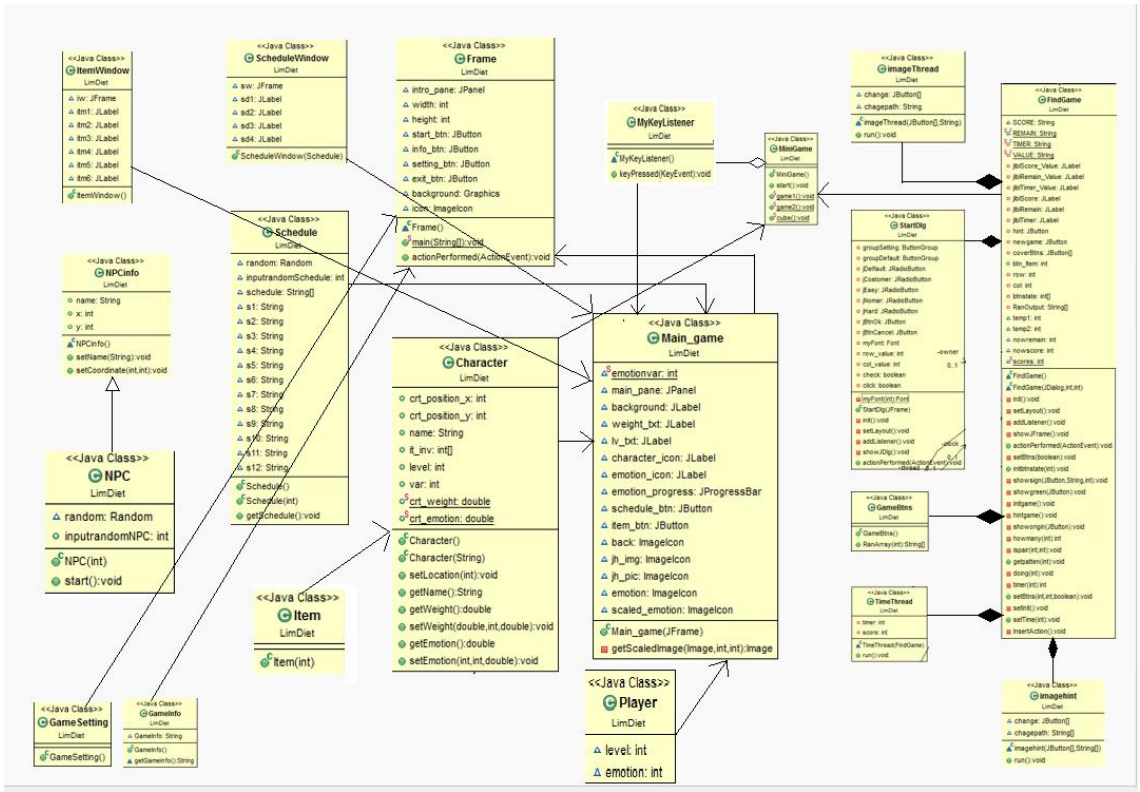
화면 layout	설명
	1. 화면 설명 - 게임 진행 중 나오는 임의의 미니게임이다. 2. 주요화면 설명 1) 제목- 미니게임의 종류가 나타난다. 2) 게임 화면- 화면에 나와있는 각 숫자를 터치하면 게임이 시작되고 결과가 나타난다.

9. 클래스 다이어그램

1) 초기 단계 – 설계

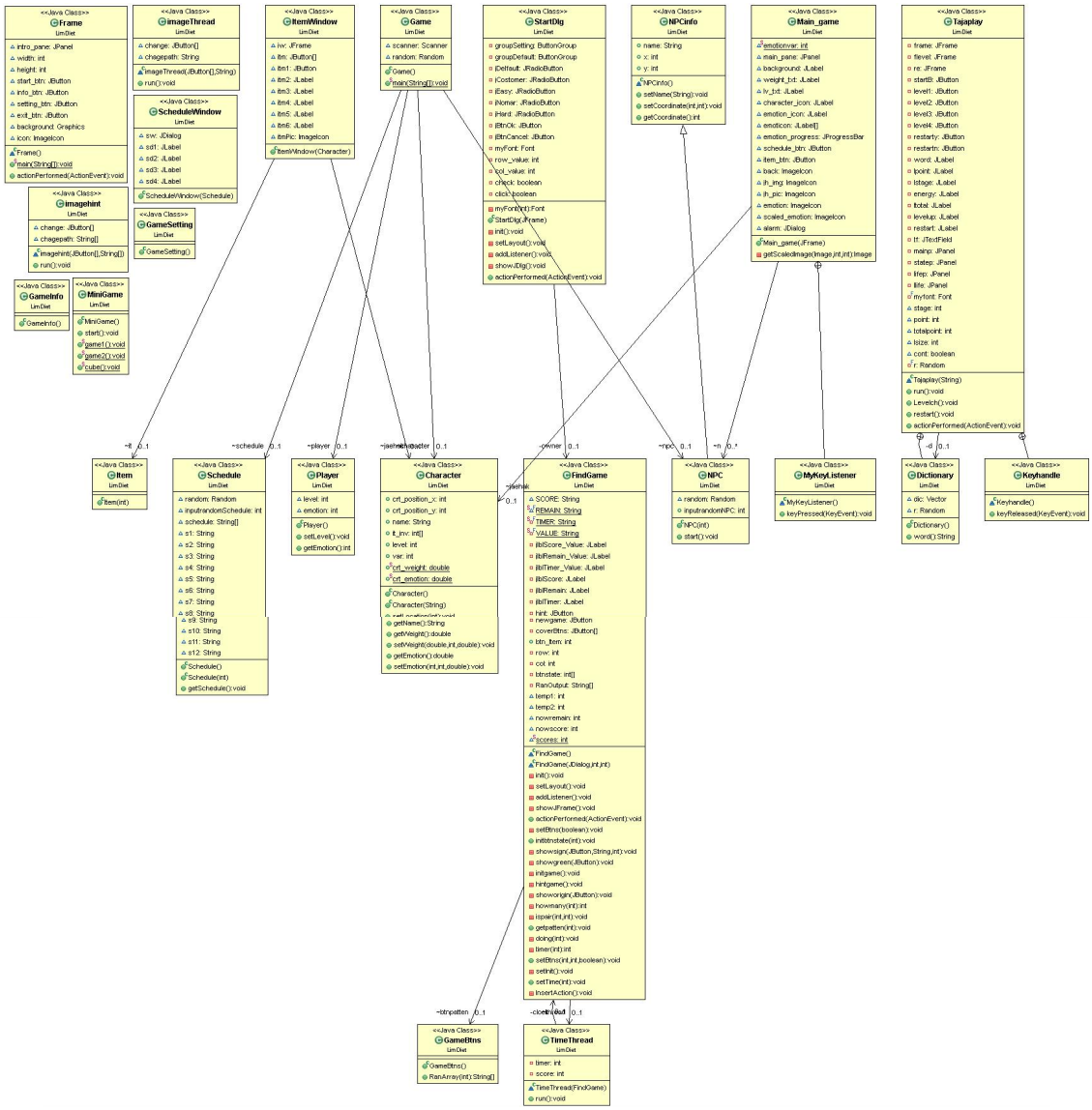


2) 중간 과정 클래스 다이어그램

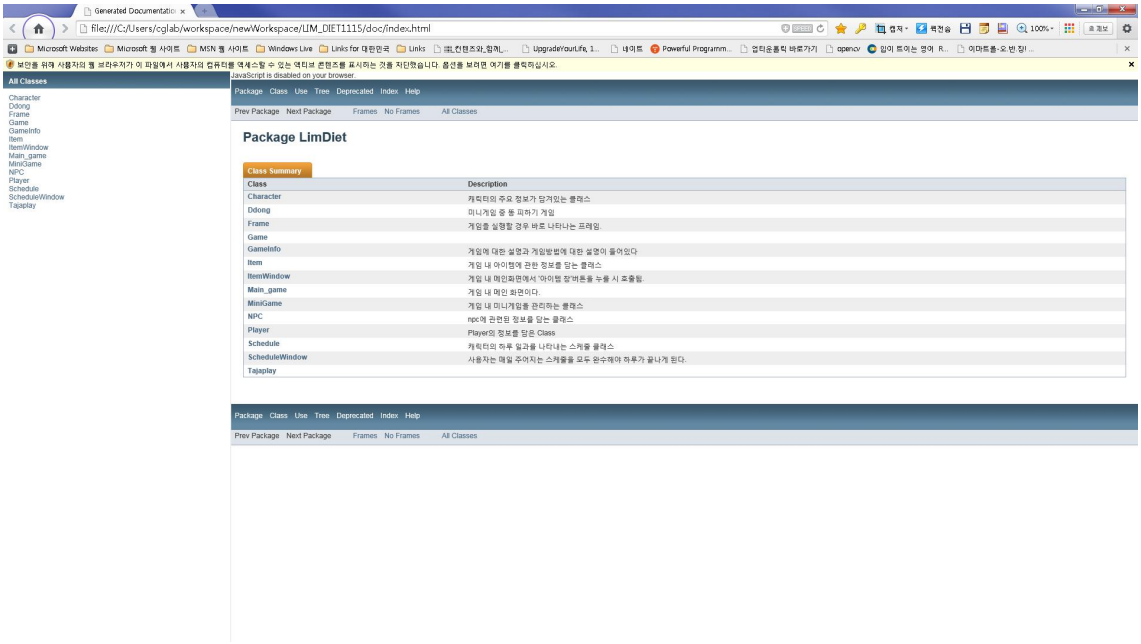


3) 최종 클래스 다이어그램

(이클립스 자동 생성 기능 사용)

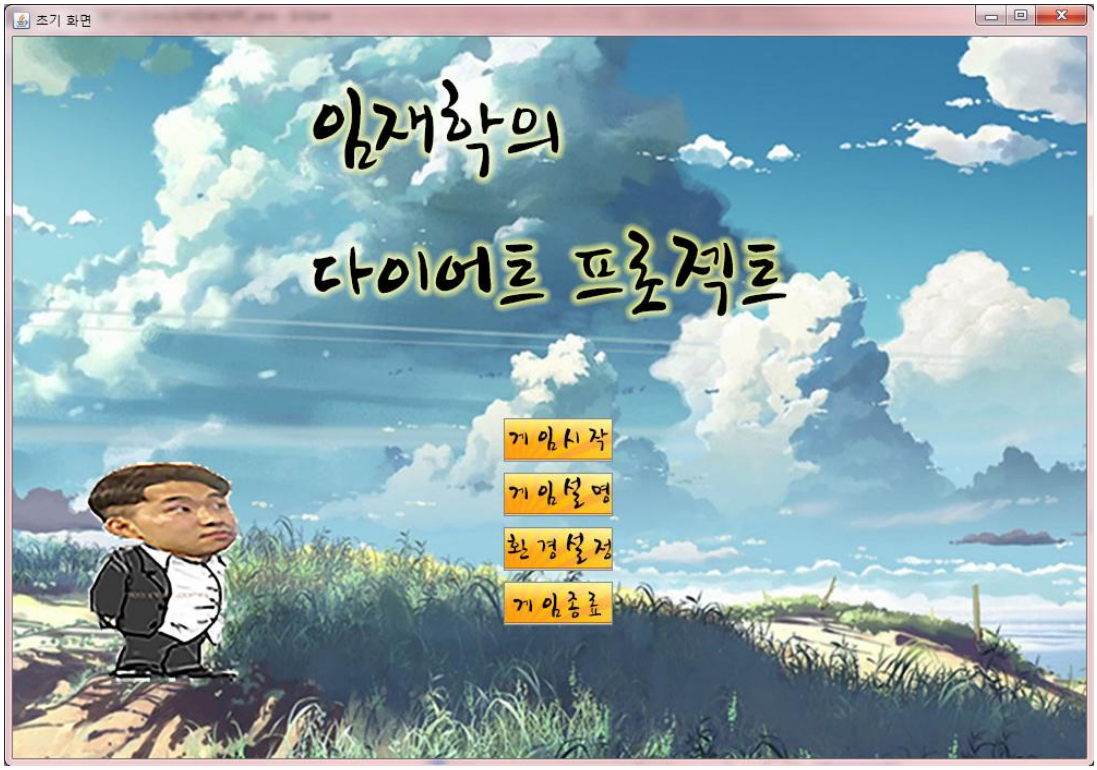


10. API 매뉴얼 (Javadoc 사용)초기페이지 스크린샷

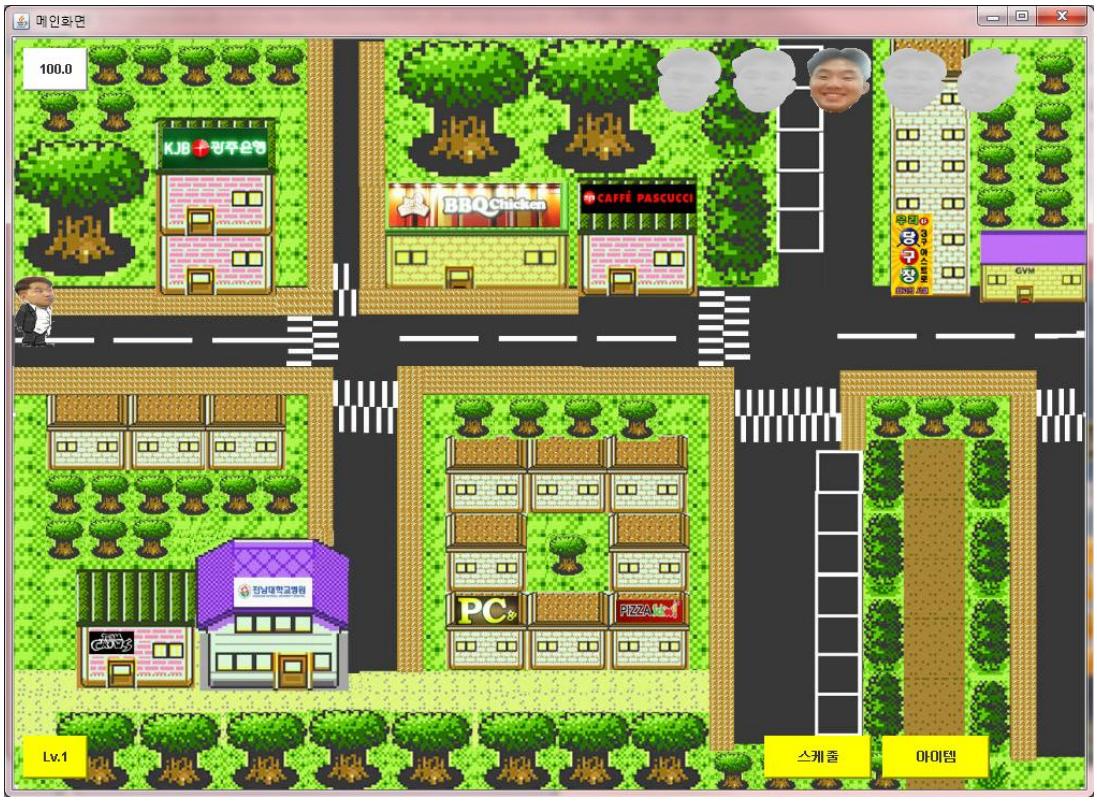


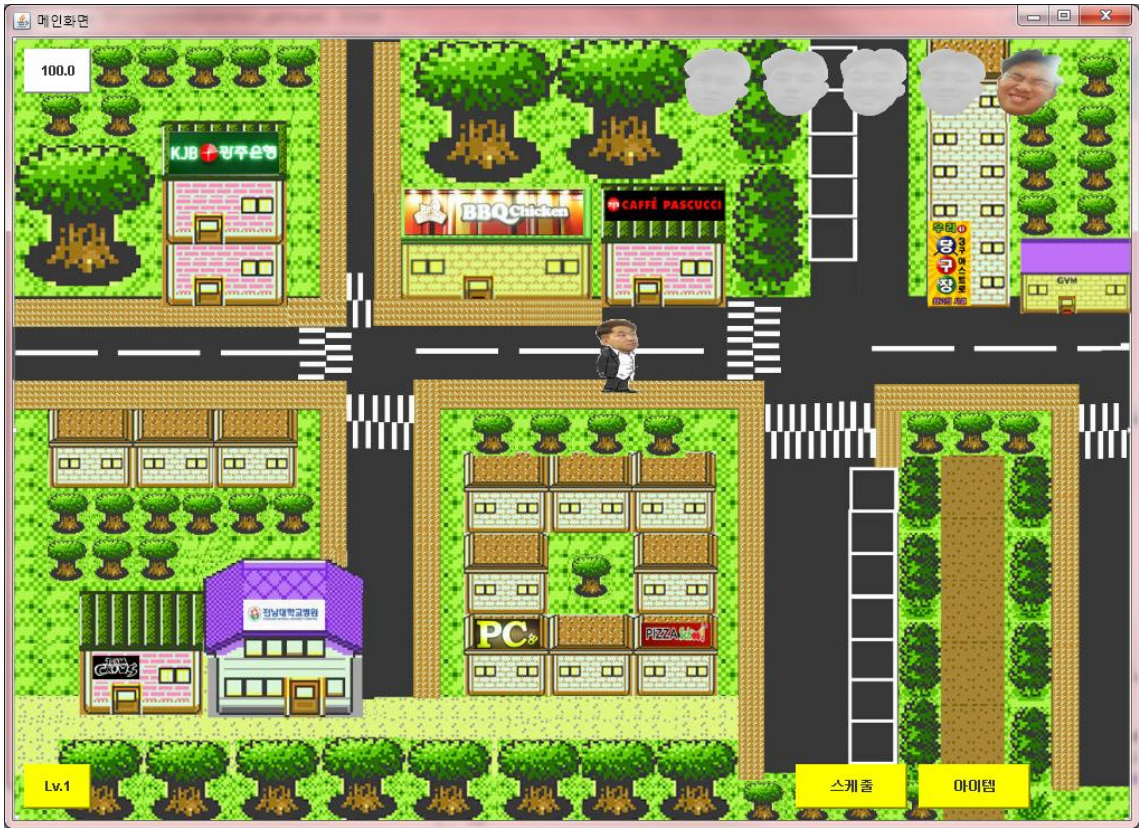
11. 프로그램 구동 스크린 샷

1) 초기화면



2) 메인화면



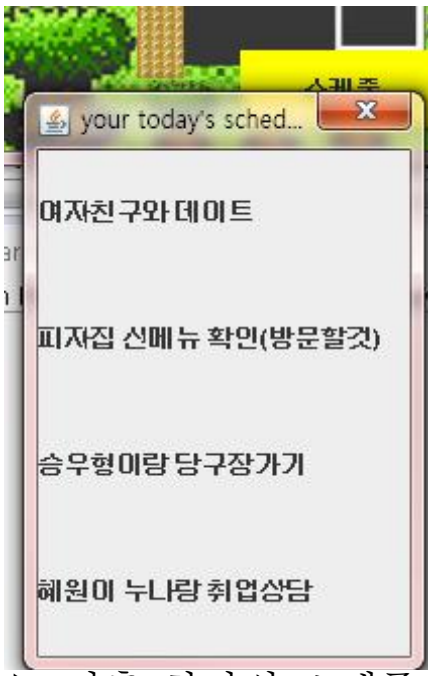


▶ 캐릭터가 맵 내에서 자유롭게 움직이고 우측 상단의 기분 아이콘이 일정 시간마다 다섯 가지 중 하나로 랜덤하게 바뀌고 있다.

3) 아이템 창

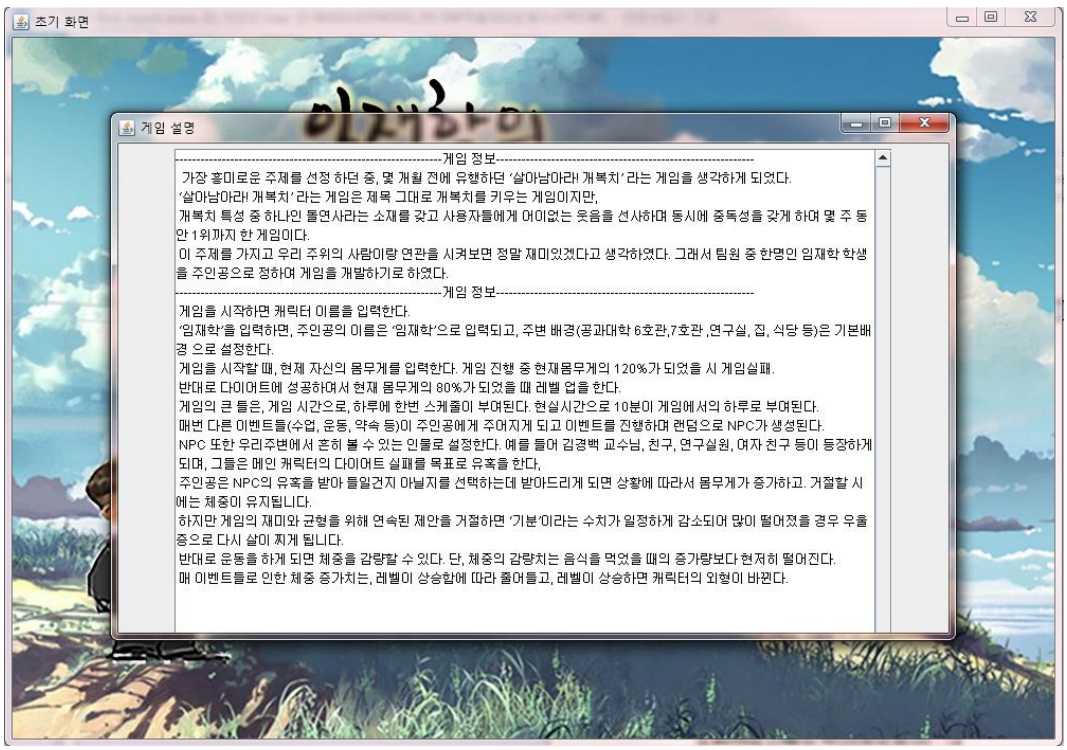
▶ 메인 화면의 우측 하단 아이템 버튼을 클릭하면 아이템 인벤토리 창(6칸)이 뜨고 해당 아이템을 클릭하면 몸무게 혹은 기분이 변한다. 아이템을 써서 몸무게가 감소하여 96kg가 된 화면을 캡쳐하였다.

4) 스케줄 창

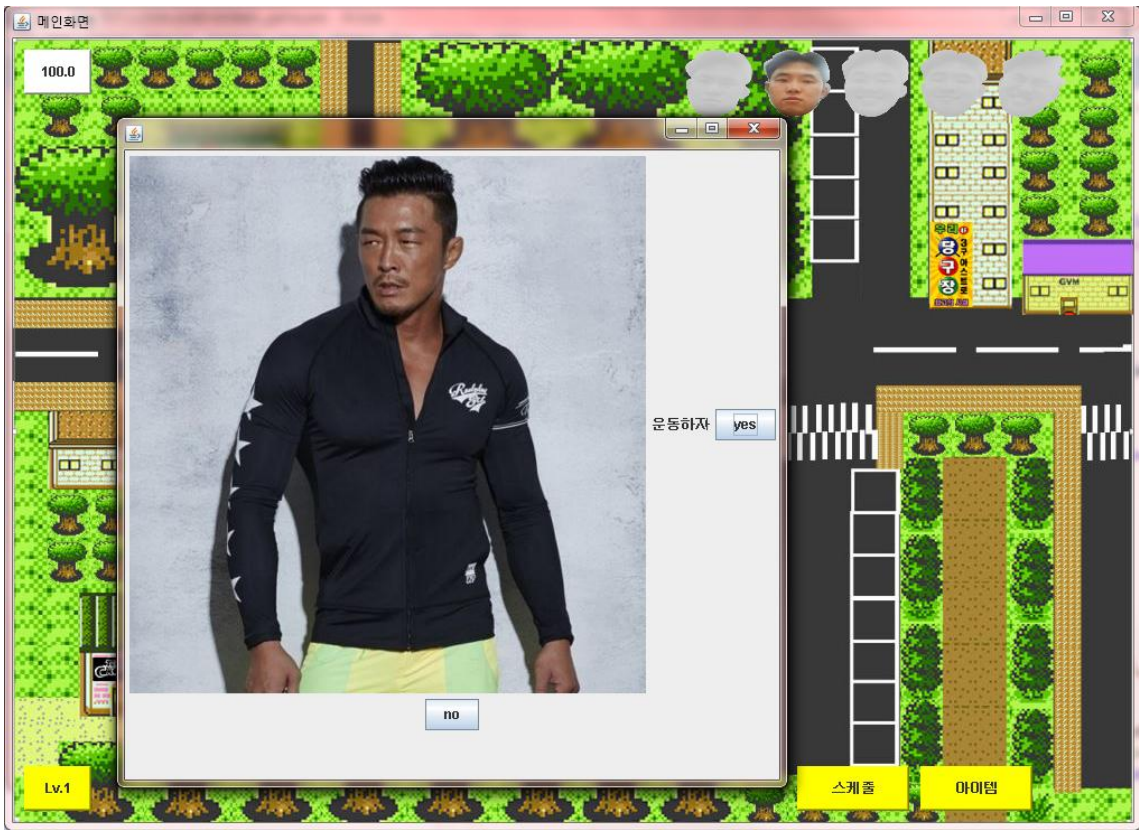


▶ 좌측 하단의 스케줄 버튼을 클릭하면 그 날 수행해야 할 스케줄이 나타난다.

5) 초기화면 - 게임 설명



6) 게임 내 미니게임

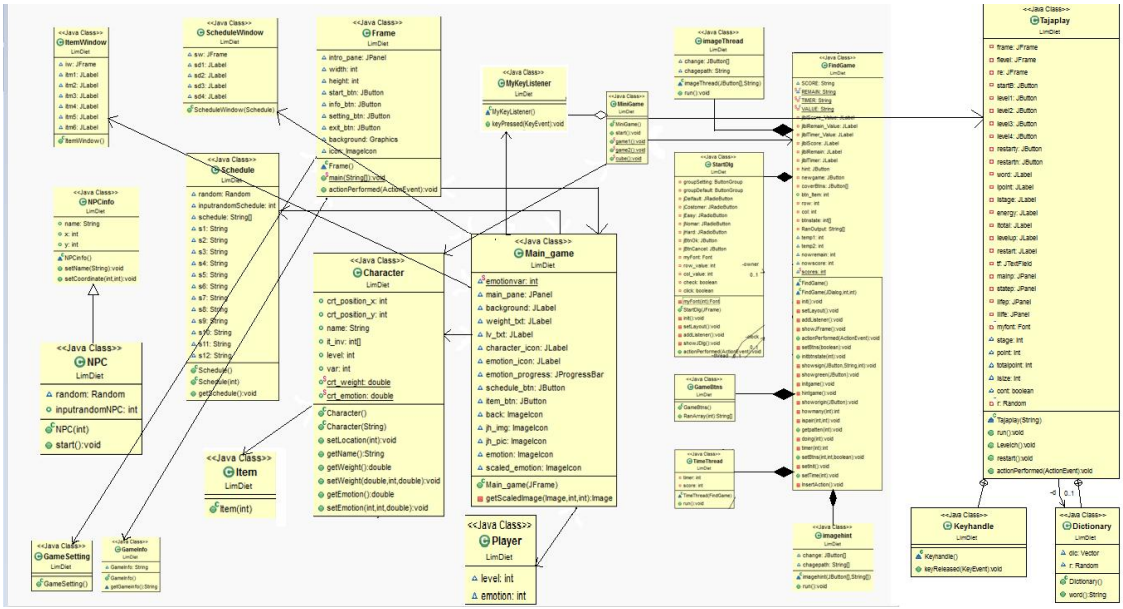


▶ 캐릭터가 스케줄을 소화하기 위해 맵을 돌아다니다 맵 내의 보이지 않는 NPC와 만나게 될 경우 메시지 창이 뜨게 된다.

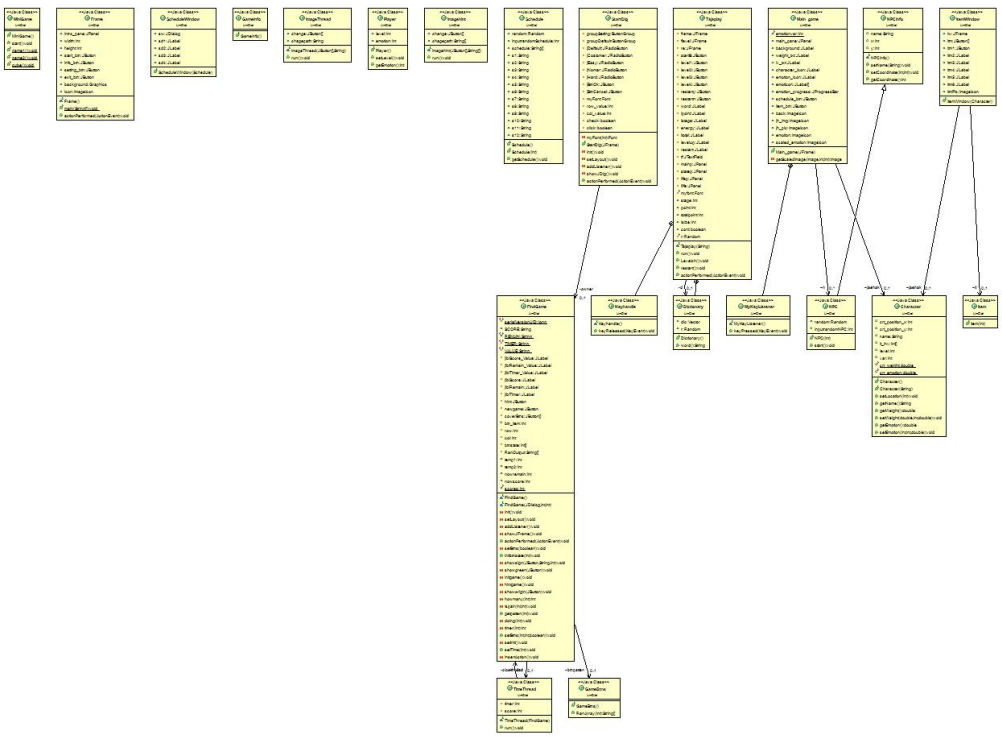
12. 기능 구현 상황

1) 임재학

a)클래스 다이어그램 구현



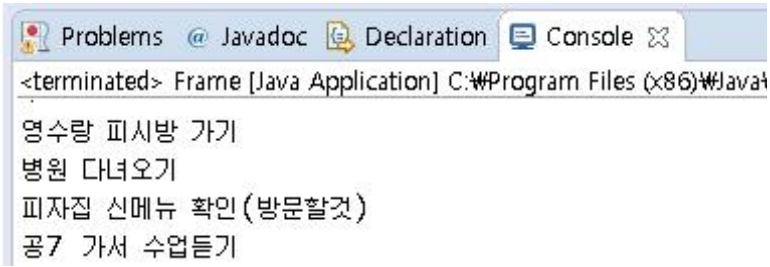
이클립스의 클래스 다이어그램 그려주는 기능을 사용했는데,



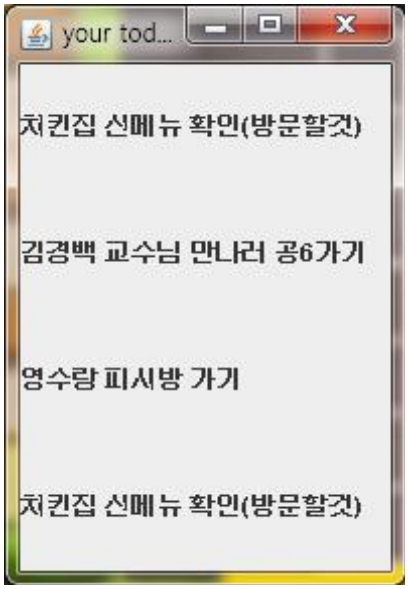
위에 보이는 것과 같이 약간 이상하게 나와서 보기 좋게 다시 그리는 작업을 진행 하였고 추후 클래스가 더 추가 되지 않는 이상 변동이 없으므로 90%이상의 완성률이라고 할 수 있습니다.

b)Schedule Class

스케줄 클래스는 지난 알파버전과 크게 다를 바 없는 상태입니다. 게임에서 NPC등장을 위해 스케줄 클래스 내부에 좌표를 설정하는 코드를 구현하려 했는데 오히려 코드의 복잡성만 증대되는 것 같아서 다른 곳으로 뺐습니다.



이런 형식으로 겹치지 않는 4가지 스케줄이 뜨고 실제 게임 상에서는 아래와 같이 보입니다.




```
public void getSchedule(){
        int input[] = new int[4];
        for(int i=0;i<4;i++){
                input[i] = random.nextInt(12)+1;
                if(i==1){
                        if(input[i-1]==input[i]){
                                i--;
                                continue;
```

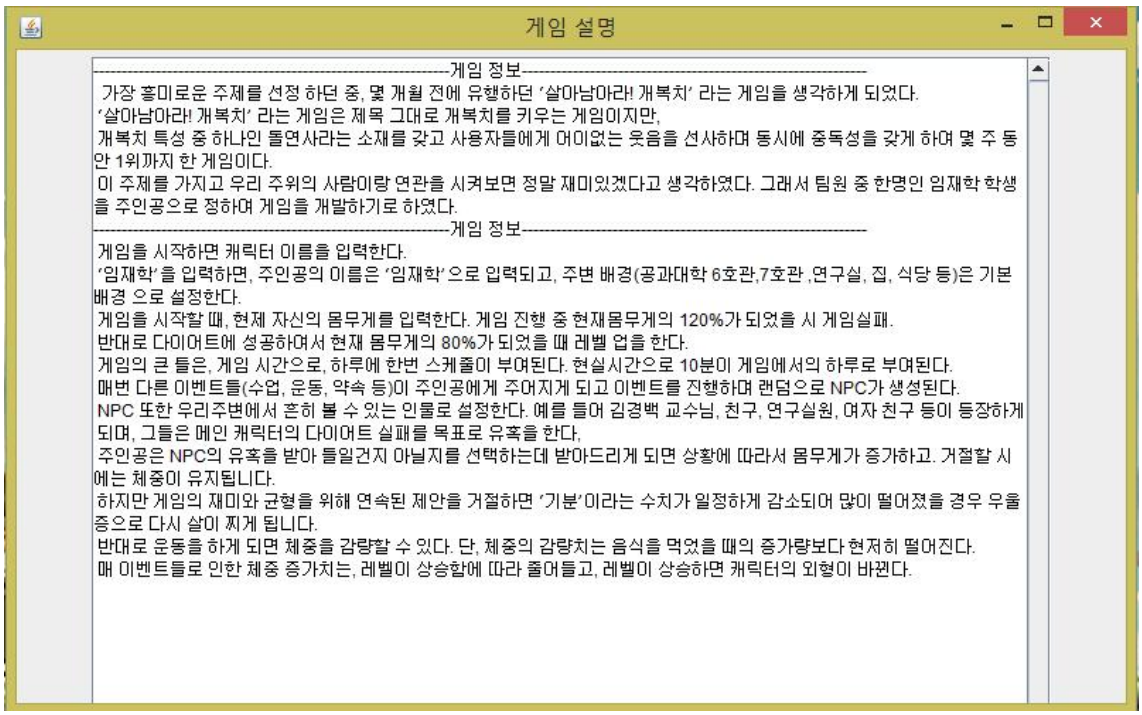
```
            }
        }
        if(i==2){
            if(input[i-1]==input[i]){
                i--;
                continue;
            }
        }
        if(i==3){
            if(input[i-1]==input[i]){
                i--;
                continue;
            }
        }
        switch(input[i]){
        case  1:schedule[i]=s1;
        break;
        case  2:schedule[i]=s2;
        break;
        case  3:schedule[i]=s3;
        break;
        case  4:schedule[i]=s4;
        break;
        case  5:schedule[i]=s5;
        break;
        case  6:schedule[i]=s6;
        break;
        case  7:schedule[i]=s7;
        break;
        case  8:schedule[i]=s8;
        break;
        case  9:schedule[i]=s9;
        break;
        case  10:schedule[i]=s10;
```

```
                    break;
                    case  11:schedule[i]=s11;
                    break;
                    case  12:schedule[i]=s12;
                    break;
                    }
            }
            for(int  i  =  0;  i  <  4  ;  i++){
                    System.out.println(schedule[i]);}}
```

위와 같은 메소드를 사용하였습니다.

c)Gameinfo Class

게임인포 클래스는 게임에 대한 설명과 게임 방법을 간단하게 독립적인 프레임으로 나타낸 것입니다.




```
text = new JTextArea(30,60);
text.setLineWrap(true);
text.setEnabled(false);
text.setDisabledTextColor(Color.black);
```

위와 같은 메소드를 사용하여서 수정 불가능하게 하였고 수정불가능 하게 하면 글자가 연해져서 안보이게 된 것을 setDisabledTextColor 메소드를 통해 다시 검정이라는 색을 설정하여

잘 보이게 만들었습니다.

d) Player Class

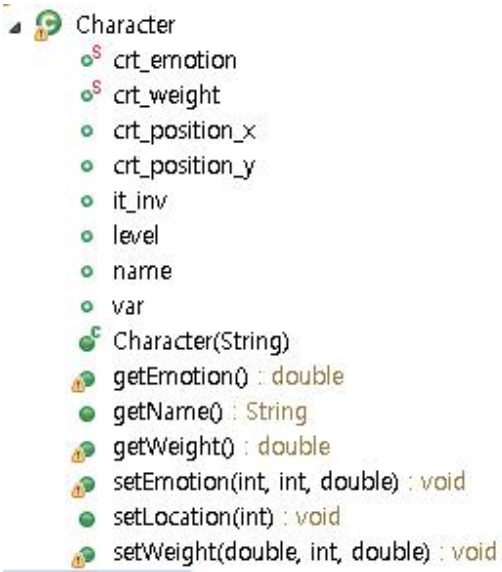
기존의 플레이어의 정보를 저장 이후 계속 사용할 수 있도록 레벨과 감정 변수를 저장해놓은 플레이어 클래스가 있습니다.

```
public class Player {

    int level;
    int emotion;

    public void setLevel() {
        this.level++;
    }
    public int getEmotion() {
        emotion = (int)(Math.random()*5);

      return emotion;
    }
}
```

2) 정혜원

a) Character class 구현

캐릭터 클래스는 아래 사진과 같이 구현되었습니다.



> character class

캐릭터 클래스에는 게임의 성패를 결정하는 몸무게, 감정 수치 변수가 있습니다. 아이템을 적용할 때나, 미니게임 결과를 반영할 때, 메인 게임 화면에서 캐릭터의 정보를 나타낼 때 등 캐릭터 객체의 몸무게와 감정 수치를 필요로 하는 경우가 많이 있으므로 몸무게, 감정 수치를 반환하는 메소드를 구현하였습니다.

몸무게, 감정 수치를 적용하는 set메소드도 각각 구현하였습니다. 몸무게의 경우에는 증가할 때, 감소할 때를 구분하여 레벨이 높아질수록 원래 증가할 양보다 덜 증가하게 하고 원래 감소할 양보다 더 감소하도록 Player Class에서 레벨 변수를 받아와서 아래 사진과 같이 구현하였습니다.

```
public void setWeight(double i , int wt_v, double lv ){
   if(i==-1) this.crt_weight -= wt_v * (1 + lv);
   else if(i==1) this.crt_weight += wt_v * lv ;
   else System.out.println("error");
}
```

▶ setWeight method

감정 수치를 적용하는 메소드의 경우에는 감정 수치가 증감되는 양이 일정하면 패턴을 예측할 수 있어 게임이 지루할 수 있을 것이라는 다른 팀의 피드백을 적용하였습니다. 일정 시간마다 변하는 감정변수를 도입하여 증감되는 양을 감정 변수에 따라 바뀌게 하였습니다. 감정 변수는 0.5, 0.75, 1, 2, 3 중 하나로 원래 기분이 증감되는 양에 곱해져서 적용됩니다. 따라서 사용자가 증감되는 패턴을 예측하기 힘들도록 아래 사진과 같이 구현하였습니다.

```
public void setEmotion(int i, int emotion_v, double var){
   if(i==-1) this.crt_emotion -= ( emotion_v * var ) ;
   else if (i==1) this.crt_emotion += ( emotion_v * var );
   else System.out.println("error");
}
```

▶ setEmotion method

마지막으로 캐릭터의 위치와 관련된 변수와 메소드입니다. 생성자에서 캐릭터의 초기위치를 설정하고 캐릭터의 위치를 설정하는 메소드를 구현하였습니다. 위치를 설정하는 메소드는 정수형 파라미터를 받아와 각 값에 따라 x좌표, y좌표를 조정하였습니다. 아래 사진과 같이 구현하였습니다.

```
public void setLocation(int i){
    if(i==6) this.crt_position_x += 10;
    if(i==4) this.crt_position_x -= 10;
    if(i==8) this.crt_position_y += 10;
    if(i==2) this.crt_position_y -= 10;
}
```

▶ setLocation method

b) frame class 구현

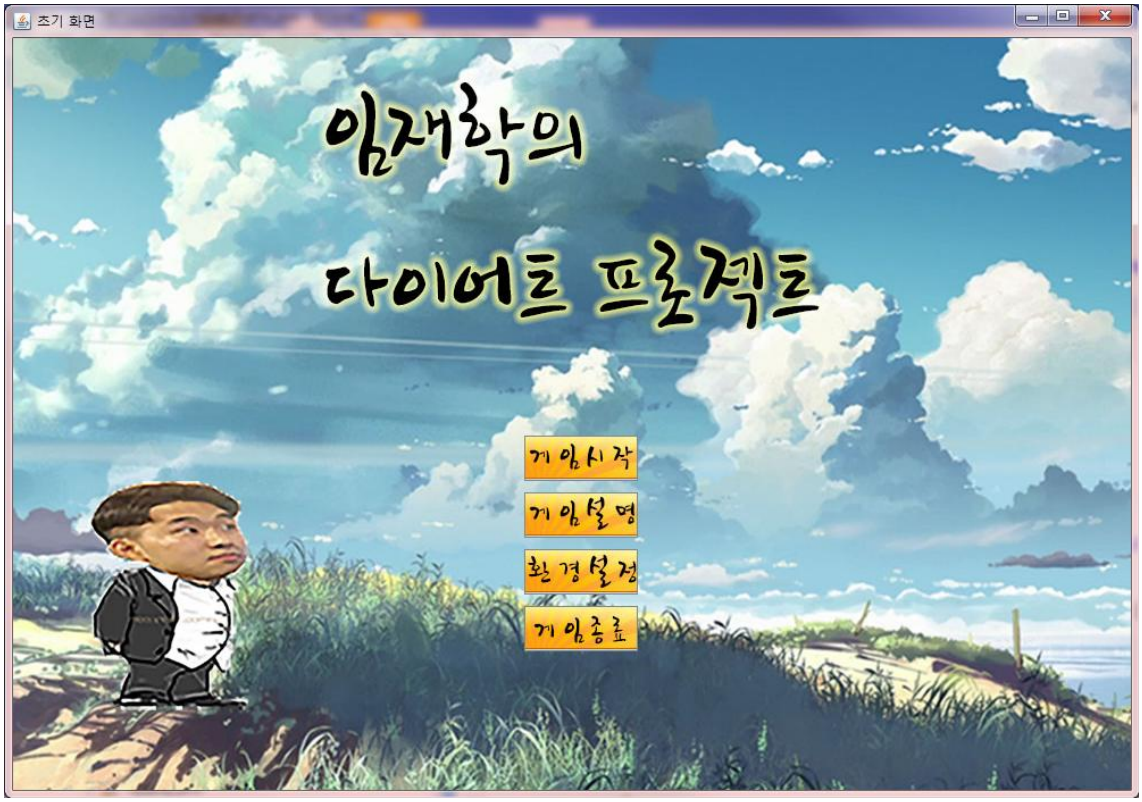
프레임 클래스는 아래 사진과 같이 구현되었습니다.

▶ Frame class

main() 메소드가 실행되면 아래 사진과 같이 바로 프레임 객체를 생성하게 됩니다.

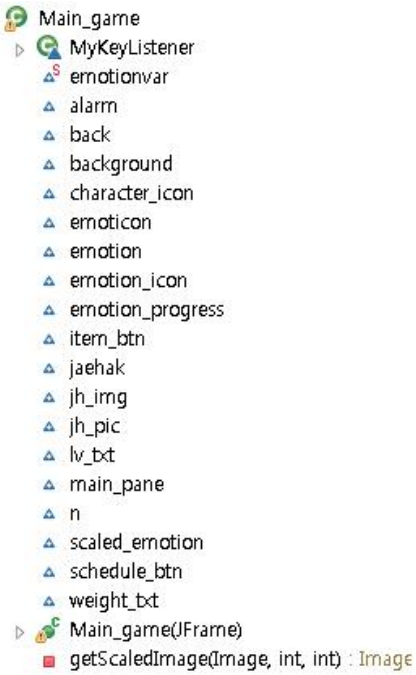
```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    Frame frame = new Frame();
}
```

프레임 클래스에서는 게임 초기 화면 창을 생성하였습니다. 게임 초기화면에는 게임 시작, 게임 설명, 환경 설정, 게임 종료 네 개의 버튼이 아래 사진과 같이 포함되어있습니다.



c) Main_game class 구현

메인 게임 클래스는 아래 사진과 같이 구현되었습니다.



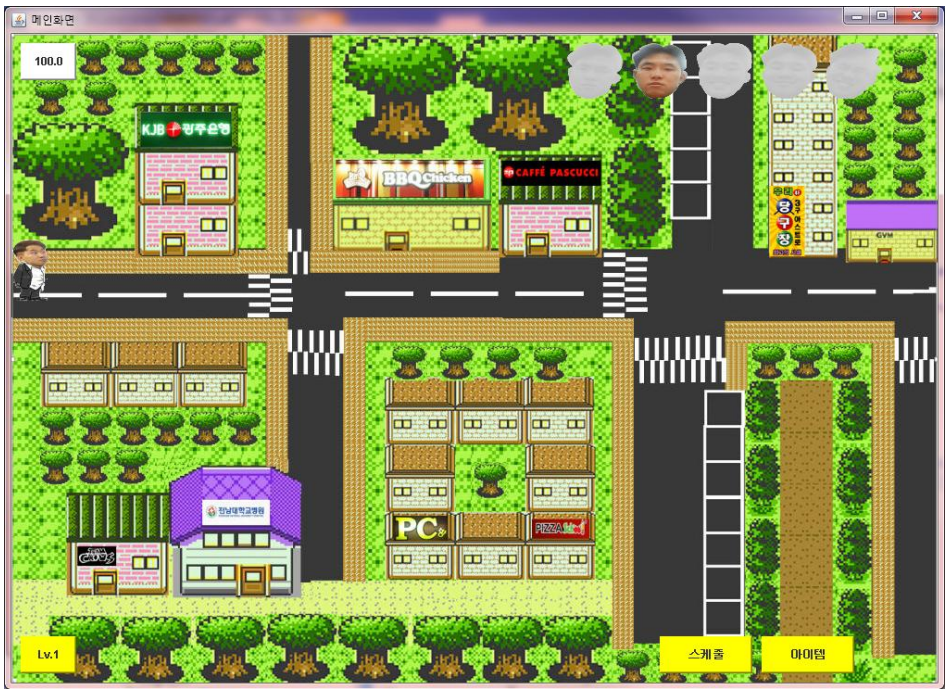
▶ Main_game class

게임 초기화면에서 게임 시작 버튼을 눌렀을 경우 나타나는 메인 화면 창을 구현하였습니다.게임 진행에 필요한 캐릭터의 레벨, 몸무게, 감정 이모티콘, 스케줄, 아이템 등 여러 컴포넌트 들이 포함되어있습니다.

또한 일정 시간마다 계속 체크해야할 항목들을 위해 java timer 객체를 사용하였습니다. 총 두 가지의 타이머 객체를 사용했는데 먼저 감정 이모티콘 타이머입니다. 감정 변수가 바뀔 때마다 감정 이모티콘으로 사용자에게 간접적으로 알리게 됩니다. 일정시간마다 바뀌는 감정 이모티콘을 나타내기 위한 메소드로써 바뀐 이모티콘만 보이고 나머지 이모티콘들은 setEnabled(false)처리하여 보이지 않게 하였습니다. 감정 변수는 플레이어 클래스에서 1~5 사이의 숫자로 선언되기 때문에 플레이어 객체를 선언하여 감정 변수를 받아옵니다. 아래 사진과 같이 구현하였고, 해당 결과 사진입니다.

```
        public void run() {
            jaehak.var=p.getEmotion();
            System.out.println(jaehak.var);


             for(int i=0;i<5;i++){
                    if(emoticon[i] != null)
                       if( i == jaehak.var)
                          emoticon[i].setEnabled(true);
                       else
                          emoticon[i].setEnabled(false);
                 }
        }
    }, 1000,1000);
```

▶ 첫 번째 타이머. 즉각적인 확인을 위해 1초마다 바뀌는 것으로 설정해두었습니다.

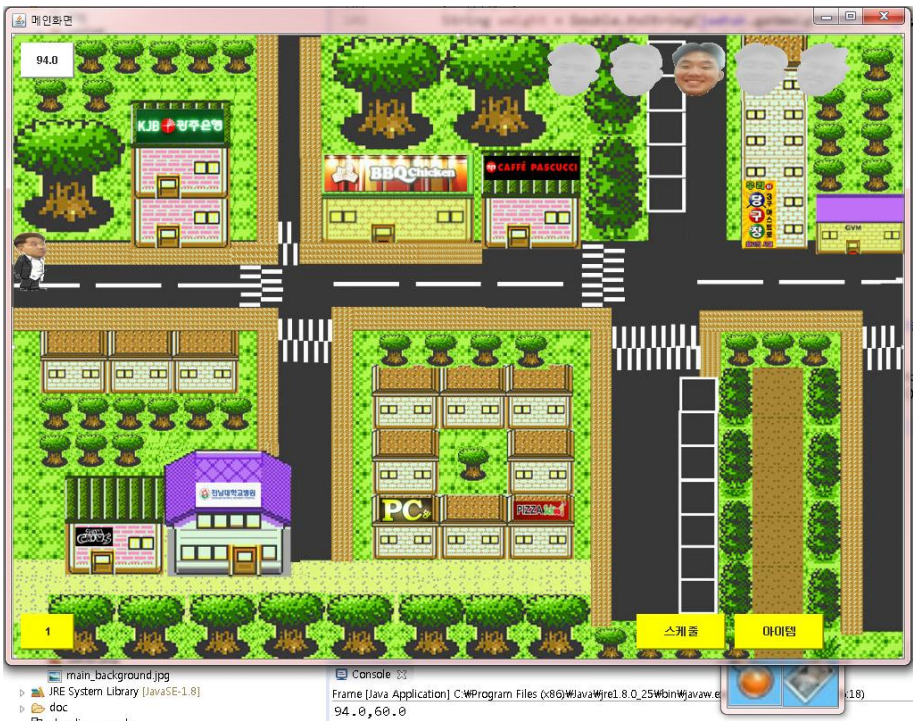


▶ 우측 상단의 이모티콘.

두 번째 타이머 객체는 몸무게, 레벨의 변동을 체크하는 타이머입니다. 메인 게임 프레임 내에 있는 몸무게, 레벨을 나타내는 부분은 캐릭터 정보에 변경이 있을 때마다 가져와서 즉시 나타내야하므로 일정시간마다 캐릭터의 정보를 체크하여 변동이 있을 시 나타내도록 아래와 같이 구현하였습니다.

```
        public void run() {
            String newWeight = Double.toString(jaehak.getWeight());
            if(weight!=newWeight) weight_txt.setText(newWeight);
            String newLevel = "Lv."+Integer.toString(jaehak.level);
            if(level!=newLevel) lv_txt.setText(newLevel);
        }
    }, 500,500);
```

▶ 두 번째 타이머



▶ 아이템을 적용한 결과 감소한 몸무게가 좌측 상단에 바로 표시되는 것을 확인하였습니다.

그리고 메인 프레임에 keyListener를 달아서 메인 게임 실행 중 키보드 방향키가 입력 될 때마다 캐릭터의 위치를 조정하여 화면에 나타내도록 구현하였습니다. 캐릭터가 메인 프레임 내의 좌표(0<x<1000,0<y<700)에 머물러 있는지, 좌표를 벗어났는지 체크하여 맵을 벗어나게 되면 새로 알림 창을 띄워 사용자에게 경고합니다. 또 맵 내의 보이지 않는 NPC와 마주치는지도 체크합니다. 캐릭터가 맵 안에서 움직이다가 NPC의 좌표와 같게 되는 순간 해당 NPC를 호출하게 됩니다.

메인 게임 클래스에서 프로젝트 초기에 설계한 화면 정의서대로 모두 구현하였으나 감정 변수를 보여주는 JProgressBar를 구현하지 못하였습니다.
아래 사진과 같이 코딩하였으나 메인 화면 내에 제대로 나타나지 않았습니다. 그래서 사용자는 캐릭터의 감정 수치가 대략 어느 정도인지 볼 수 없게 되었습니다.

```
/*감정 게이지바*/
JPanel p_panel = new JPanel();
emotion_progress = new JProgressBar();
emotion_progress.setValue((int)jaehak.crt_emotion);
emotion_progress.setStringPainted(true);
emotion_progress.setIndeterminate(true);
emotion_progress.setOpaque(true);

p_panel.setLayout(new FlowLayout());
p_panel.setLocation(750, 100);
p_panel.add(emotion_progress);
p_panel.setVisible(true);

emotion_progress.setMinimum(0);
emotion_progress.setMaximum(100);
emotion_progress.setVisible(true);
mf.getContentPane().add(p_panel);
mf.add(p_panel);
```

d) ItemWindow class 구현

아이템 윈도우 클래스는 아래 사진과 같이 구현되었습니다.

▶ ItemWindow class

아이템윈도우 객체가 호출되면 아이템윈도우 생성자에서 GridLayout layout manager를 가진 새로운 프레임을 생성합니다. 게임 내의 모든 아이템(6개)이 보이며 사용자가 해당 아이템을 클릭할 경우 아이템을 사용하여 몸무게, 감정 변수에 적용할 수 있게 하였습니다. 아이템에 대한 정보는 아이템 클래스와 연동하여 캐릭터 객체 정보가 잘 변하는지 확인하였습니다.

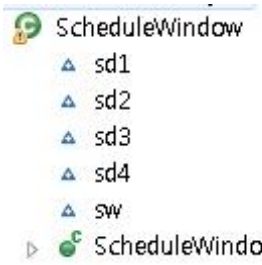
메인 게임 실행 중 사용자가 아이템 창을 접근하게 되면 새로운 창이 생기는데 원래 프레임이 포커스를 잃게 되어 메인 화면의 keyListener가 작동을 멈추게 되었습니다. 이 문제를 해결하기 위해 아이템윈도우 생성자를 선언할 때 jframe을 인자로 받아옵니다. 아이템윈도우 객체는 메인프레임에서 호출되는데 그때 메인프레임을 인자로 넘기면 아이템윈도우 클래스에서

windowListener를 추가하여 해당 작업이 끝나면 원래 프레임으로 다시 포커스를 맞춰주는 작업을 진행합니다. 이 방법은 스케줄윈도우 클래스에서도 동일하게 적용됩니다.

▶ 아이템 적용 결과

e) ScheduleWindow class 구현

스케줄 윈도우 클래스는 아래 사진과 같이 구현되었습니다.



▶ ScheduleWindow class

스케줄 윈도우 객체가 호출되면 jframe, schedule 객체를 인자로 받고 스케줄 윈도우 생성자에서 GridLayout layout manager를 가진 새로운 프레임을 생성합니다. 게임 내 하루의 스케줄이 보이며 캐릭터는 하루 스케줄을 모두 소화해야 하루를 끝낼 수 있게 됩니다. 스케줄에 대한 정보는 스케줄 클래스에 접근해서 내용을 가져온 후 스케줄 윈도우 클래스를 통해 화면에 보이게 됩니다.

스케줄 버튼을 누를 때마다 새로운 스케줄 객체를 생성하므로 그때그때 다른 스케줄을 볼 수 있습니다.

3) 이승우

a) 다양한 효과의 아이템 구현

```
/** 배열을 이용하여 게임아이템들을 선언 */
String[] item = new String[] {"머슬워터","근육보충제","우유","랜덤박스","청정해양수","헬스장쿠폰"};

/** 배열안의 각 아이템들의 효과를 설정 */
switch(input){

case 0 :{

Character.crt_weight = Character.crt_weight-2;
Character.crt_emotion = Character.crt_emotion-10;
break;
}

case 1 : {
   Character.crt_weight = Character.crt_weight-2;
   Character.crt_emotion = Character.crt_emotion-15;
   break;
```

배열을 이용하여 6개의 아이템을 공간에 고정시켜 그 공간마다의 아이템에 대한 효과를 switch를 이용하여 주었다. 각 효과는 캐릭터에 대한 몸무게와 감정변화에 영향을 주며 이 아이템은 밑에서 설명할 미니게임의 승패에 따라 얻을 수 있게 된다. 더욱 많은 아이템을 설정하려 했지만 가방 특성 상 6개의 아이템이 들어갈 때 보기 좋았기 때문에 6개에서 멈추게 되었다.

b) 아이템과 캐릭터의 연동

```
int item_num = (int)(Math.random()*6) ;
jaehak.it_inv[item_num]= 1 ;
Item it = new Item(item_num);
```

Character Class 에 it_inv(아이템 인벤토리)라는 배열을 선언하여 캐릭터 아이템 인벤토리를 만들었고 MiniGame Class 에서 MiniGame을 승리 하였을 때 아이템이 지급 될 수 있게 설정하였다.

c) MiniGame 들의 구현

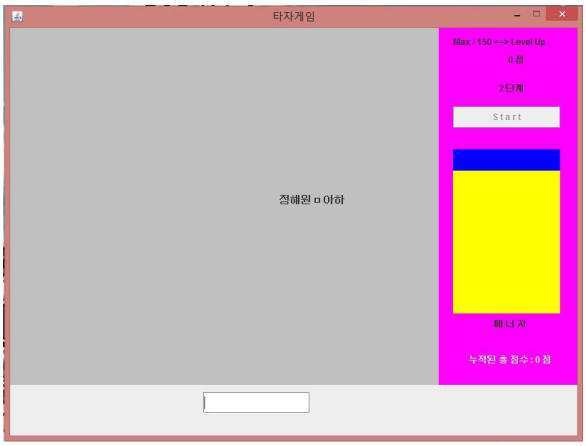
```
/**MiniGame 을 불러오는 start 메소드*/
public void start() {
   int random = (int)(Math.random()*3) + 1 ;

   /** MiniGame의 승리조건을 설정 후 아이템 보상범위 설정, 패배 할 경우의 패널티 설정 */
   switch(1){
   case 1:
      Character jaehak = new Character("jaehak");
      Tajaplay tg = new Tajaplay("타자 게임");
      if(tg.stage == 5 && tg.point == 500){

         int item_num = (int)(Math.random()*6) ;
         jaehak.it_inv[item_num]= 1 ;
         Item it = new Item(item_num);
      }else{

         int random_num = (int)(Math.random()*3);
         Character.crt_weight = Character.crt_weight+random_num;
         Character.crt_emotion = Character.crt_emotion-random_num;
      }
```

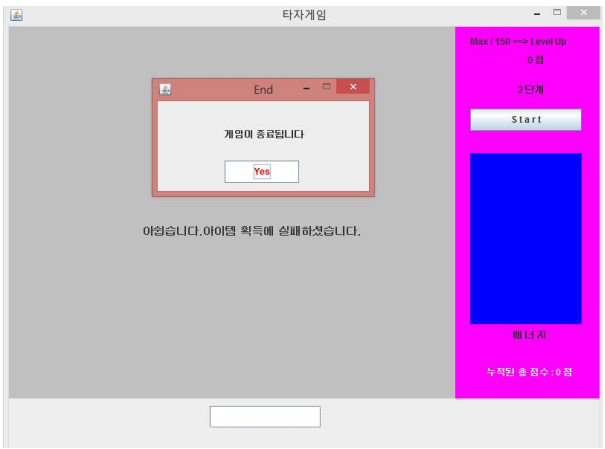
최영수 학생이 먼저 만들어 놓은 MiniGame Start 메소드안에 미리 찾아놓은 MiniGame을 Switch 안에서 NPC를 만났을 때 랜덤으로 실행 될 수 있게 하였고, 우선 위에 있는 타자게임의 승리조건을 설정하여 승리 할 시 아이템을 지급 할 수 있게 설정하였다. 위에서 만들어 놓은 6개의 아이템을 랜덤으로 지급하게 설정하였고, 게임 승리조건에 도달 할 수 없게 되면 실패하게 하였다. 게임이 실패하면 실패에 대한 패널티가 부여되게 하였다.

```
case 4:
   /** MiniGame(카드찾기) 의 승리조건을 설정 한 후 아이템이 지급될 수 있게 설정, 실패시 패널티가 부여 될 수 있게 설정 */
   Character jaehak1 = new Character("jaehak");
   FindGame fg = new FindGame();
   if(Integer.parseInt(fg.REMAIN) == 1 &&Integer.parseInt(fg.SCORE)>100){


      int item_num = (int)(Math.random()*6) ;
      jaehak1.it_inv[item_num]= 1 ;
      Item it = new Item(item_num);
   }else{

      int random_num = (int)(Math.random()*3)+1;
      Character.crt_weight = Character.crt_weight +random_num;
      Character.crt_emotion = Character.crt_emotion -random_num;
   }
   break;
}
```

위의 코드는 카드 찾기 게임에 대한 실행 코드이다. 타자게임과 같이 승리조건을 설정하였으며 6개의 아이템을 랜덤으로 지급하는 방식으로 설정하였다. 또한 실패 시 패널티를 부여할 수 있게 설정하였다.

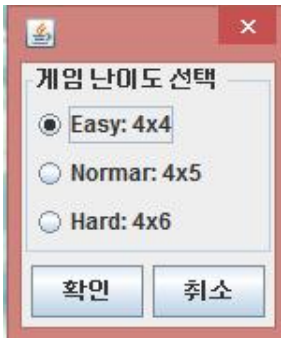
1) 타자 게임





타자게임의 실행 결과는 이렇다. Start 버튼을 누르게 되면 승리조건을 설명해주며 시작 버튼을 누르게 되면 게임이 시작된다. 총 4단계로 나눠져 있으며 4단계를 Clear 하여 그 점수로 인해 승리를 판단하게 된다. 승리를 한다면 아이템이 지급되고, 실패하면 패널티를 부여한다. 처음 이 게임소스를 가져왔을 때는 게임 종료가 되지 않는 현상이 발생하여 약간의 코드를 변경하여 게임이 종료될 수 있게 하였다.

2) 카드 찾기 게임



카드 찾기 게임의 실행화면은 이렇다. 시작을 하여 난이도를 설정하고 같은 카드를 뒤집게 되어 Score 와 Remain 의 개수로 게임 승패를 판별한다. 게임에 승리하게 되면 자동으로 게임은 꺼지게 되고, 아이템이 지급 될 수 있게 하였다. 또한 게임에 실패하게 된다면 패널티를 부여 하게 만들었다.

4) 최영수

a) 상황에 따른 Npc 구현

```
switch(inputrandomNPC){
/*추성훈 npc
 * 운동관련
 * 거절시, crt_emotion -8*/
case 1 :
    setName("추성훈");
```

Npc 생성자를 부르면 호출되는 방식으로, switch 안에 하나의 케이스로 만들었다. 이러한 '포*몬스터'와 같은 게임의 특성상, 하나의 Npc를 구현하면, 시나리오나, 게임 진행에 맞춘 NPC 들의 추가는 간단하다. image file과, 좌표, 대화 내용만 수정하면 NPC 생성은 간단하기 때문에, 하나의 NPC 의 정확한 구현과 창 크기 조절에 힘썼다.
(총 NPC 5개, 총 코딩 라인 379)

b) Main Game 과 Npc 클래스의 좌표 연동

```
class NPCinfo{
    /**npc name*/
    public String name;
    /**npc coordinates (for find npc)
    public int x, y;
    /**setname -> 아래서 npc에서 받기위해*/
    public void setName(String name){
    /**setCoordinate -> 아래서 좌표를 받기위해
     * main game 에서 불러올때 바로 사용할 수 있도록 이 좌표들은
     * 생성자에서 getCoordinate로 받아와 switch 에 넣는다.*/
    public void setCoordinate(int x, int y){
}
```

NPC class 는 NPCinfo class를 상속받고, info 에는 setCordinate 가 있다. NPC 의 x, y 좌표를 입력해줘서, main game에서 불러오도록 하기 위함이다.

```
inputrandomNPC = input;
/**main game 에서 좌표를 받을 때, NPC 생성자 실행*/
switch(input){
case 1 :
    setCoordinate(250, 220);
    break;
case 2:
    setCoordinate(350,220);
    break;
```

이 부분은 NPC 의 생성자 부분이다. NPC 클래스에서 Coordinate를 설정하고, 그에 따라 start 메소드에 따라 각자의 npc 들이 실행되게 만들었다.
사실 처음에는, npc 에 각자의 위치를 부여하고, 그 위치에서는 특정한 npc 가 실행되는 동시에, 캐릭터의 이동에 따라 매번 랜덤하게 npc를 뿌려주려 했지만, main 이 너무 복잡해지고, 매 움직임마다 검사하기에는 무리가 있어서 목표했던 기능은 구현하지 못했다.

c) Minigame 들의 구현

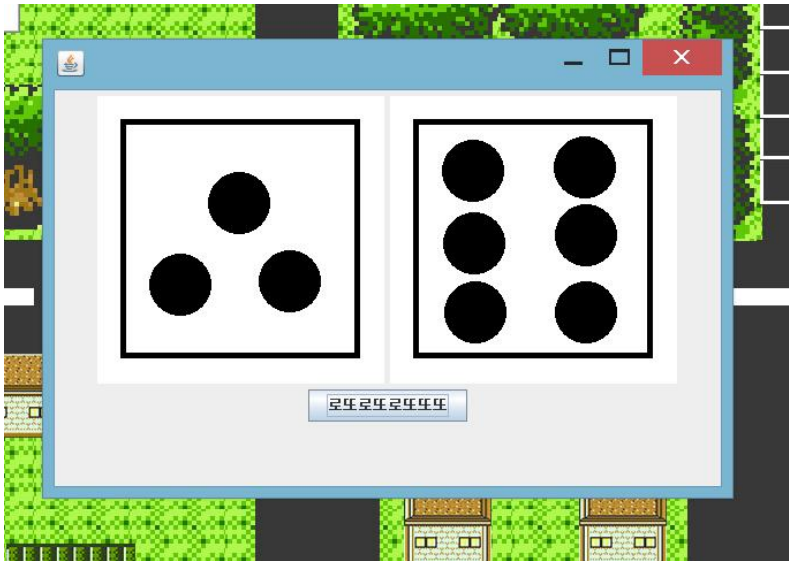
```
public class MiniGame {
    /**MiniGame 을 불러오는 start 메소드*/
    public void start() {

    /**start에서의 case 에 넣을 게임 1, 2, etc.. */
    public static void dice(){
}
```

Minigame 클래스는 나와 이승우 학생이 같이 작업을 했다. 이승우 학생의 minigame을 간단하게 불러오기 위하여, start method를 선언하였고, start 안에는 각자의 미니게임이 swtich 안에 들어있다.

```
case 2:
    MiniGame.dice();
    break;
```

이런 식으로, 각자의 게임에 이름을 붙이고, 아래 static void 로 선언하거나, 다른 클래스에서 불러오는 등의 방식으로 실행되게 하였다. minigame 의 객체는 NPC에 있기 때문에 따로 생성자는 존재하지 않는다.

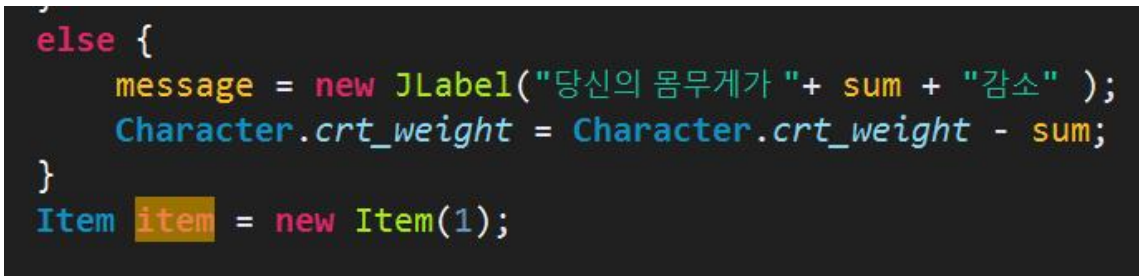
1. 주사위게임
이 주사위 게임의 실행 결과는 이렇다.



간단하게 6개의 변수에 이미지를 입히고, 더한 값을 랜덤하게 character 의 weight 에 + 혹은 −를 시켜준다.

현재, 이러한 똥 피하기 게임을 만들었는데, minigame에서 받아오는 과정에서 에러가 나서 수정중이다.
(코딩라인 : 180 + 252 = 430)

d) Character, Item 등의 다른 클래스들과 연동



주사위 게임이 끝난 후의 내용이다. 모든 게임과, NPC 에는 시나리오에 맞춰서 몸무게와, 감정변화, 그리고 아이템을 만들어 주고 있다. 하지만 목표처럼, 레벨링 시스템이나 감정 수치에 따른 변화는 주지 못하고 있다.

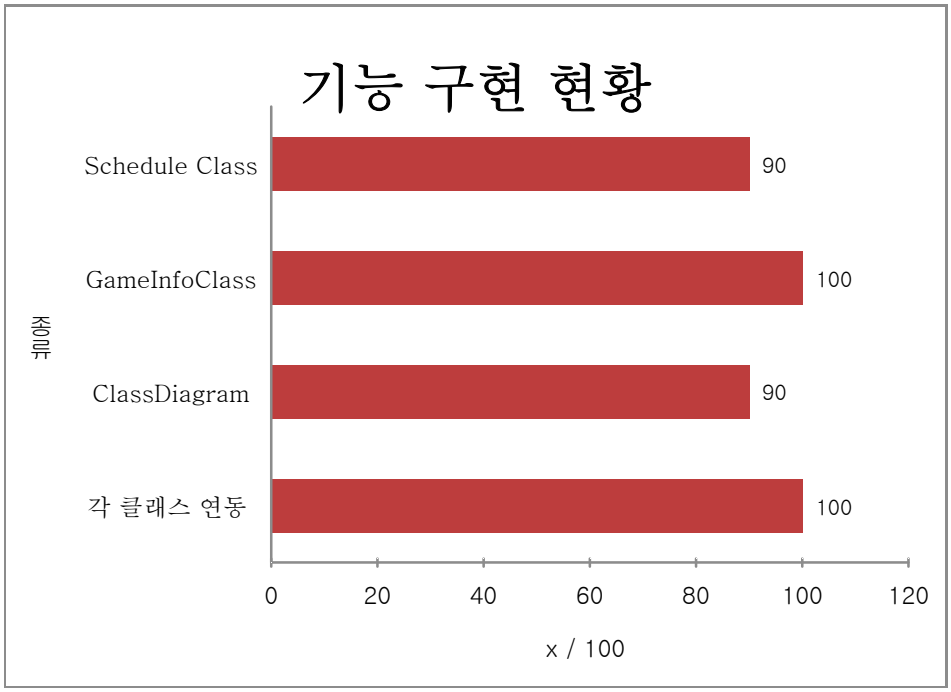
e) Npc, Mini Game 의 다양성

현재 NPC는 5개, 미니게임은 내것 2개 + 이승우 학생의 n 개다. 처음의 예상만큼 디테일하게 npc 들과 game 들을 연관 짓고, 만들어 내지 못했다.

13. 작업 진행 결과

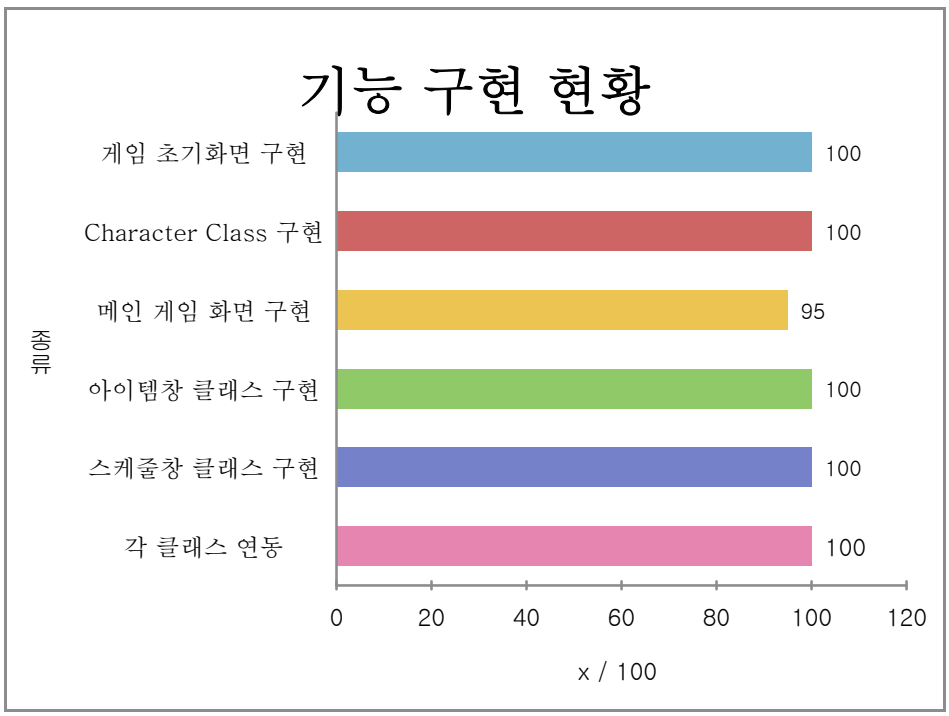
1) 임재학 : 총 217라인

최종 프로그램 기능 구현 정도 : 95%



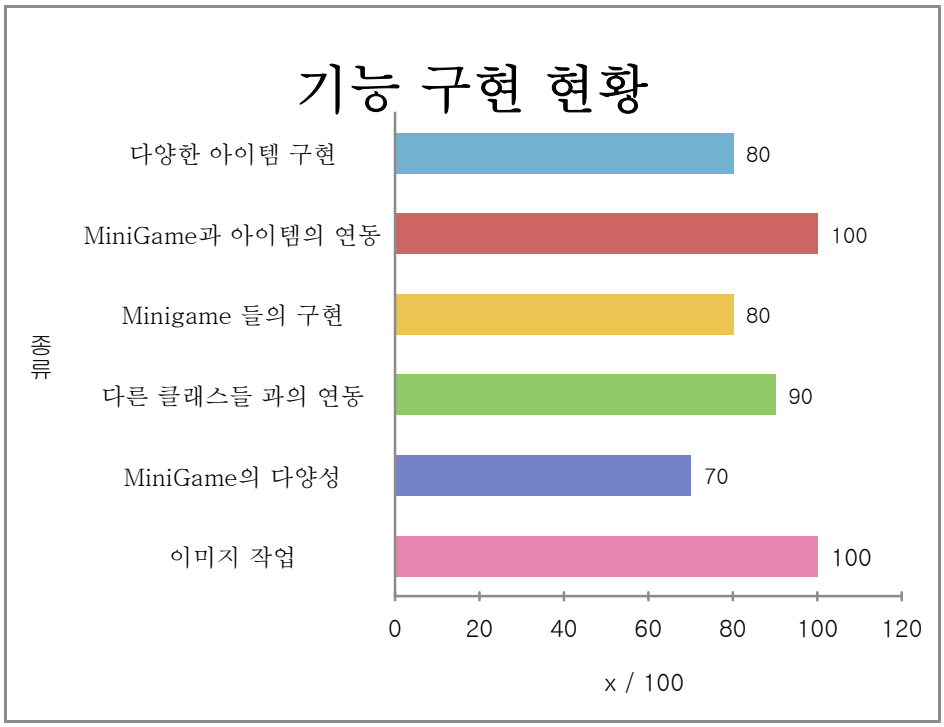
2) 정혜원 : 총 787라인

최종 프로그램 기능 구현 정도 : 99%



3) 이승우 : 총 1194라인

최종 프로그램 기능 구현 정도 : 80%



4) 최영수 : 총 809라인

최종 프로그램 기능 구현 정도 : 85%