**어드벤처 디자인 고급레벨 프로젝트**

**최종 보고서**

**(Taffy & Ladybug)**

**어드벤처 디자인 2분반(김경수 교수님)**

**2조**

**컴퓨터공학과 20161163 조장호**

**컴퓨터공학과 20191064 정용환**

**컴퓨터공학과 20191248 허환**

**목차**

1. 개요
2. 첫번째 문제(태피 상점)
   1. 문제 정의 및 입출력 명세
   2. 추가 기능 구현
   3. 최종 모듈 구현 및 기능
   4. 프로그램 전체 구조
   5. 프로그램 동작
   6. 최종 구현 결과
   7. 최종 정리 및 논의점
3. 두번째 문제(술취한 무당벌레)
   1. 문제 정의 및 입출력 명세
   2. 최종 모듈 구현 및 기능
   3. 프로그램 전체 구조
   4. 프로그램 동작
   5. 최종 구현 결과
   6. 최종 정리 및 논의점
4. 역할 분담
5. 팀프로젝트 협업
6. 참고문헌

**1. 개요**

어드벤처디자인 과목 고급레벨 과정 최종 보고서로써 이번 고급레벨은 난이도가 보통인 문제를 두개를 선택하여 구현을 하는 것이 과제 요구 사항이다. 따라서 두가지 선택문제에 대해서 목차에 따라 세부사항을 상세히 명시하여 최종 구현 보고서를 작성하였다.

**2. 첫번째 문제**

첫번째 문제로 선택한 문제는 **태피의 상점 문제**이다.

태피 상점 문제의 내용으로는 태피 상점은 한 개의 창구만을 가지며 한 명의 점원이 한 번에 한 명의 고객만을 서비스 할 수 있다. 고객을 서비스하는 시간은 1분에서 10분 사이가 소요된다. 태피 상점은 하루 8시간을 일하며 주 7일을 일한다.

**2-1 문제의 입출력 명세사항**

우리는 태피 상점의 하루의 일을 시뮬레이션하기 위해 480분을 갖는 큐 모델을 생성하고 매장의 하루 일과를 시뮬레이션을 돌린다.

프로그램에서 시뮬레이션이 종료된 후 최종적으로 출력되어야 하는 값으로는 전체 고객의 수, 전체 서비스 시간, 평균 서비스 시간, 평균 대기시간 등의 통계 정보를 출력한다.

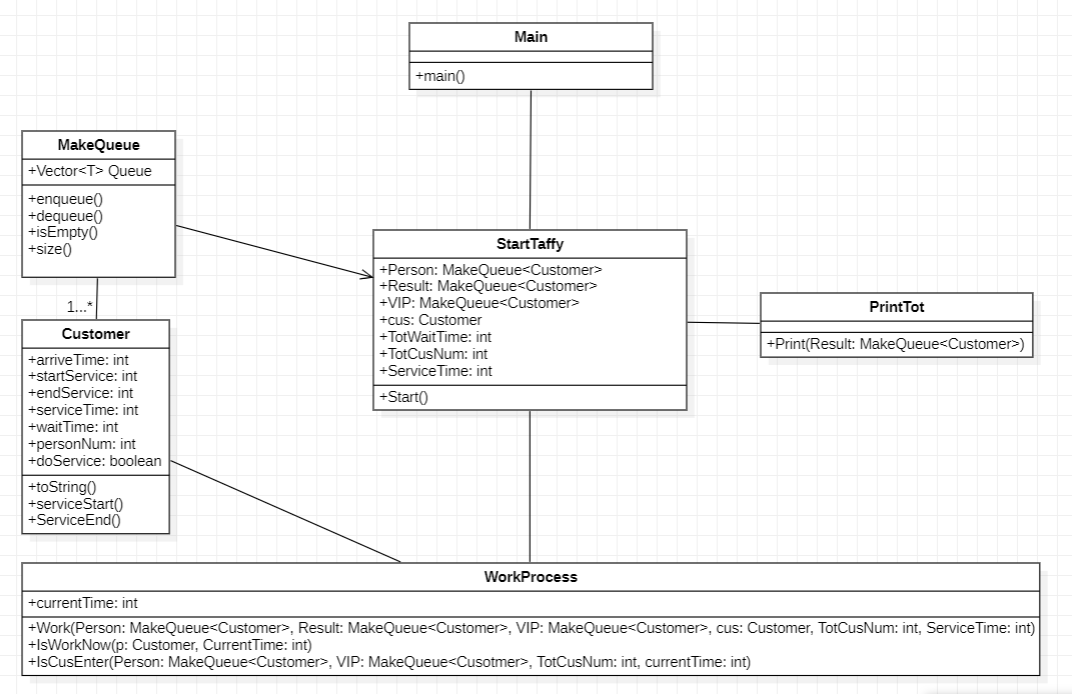
**2-2 추가 구현 기능**

1/50의 확률로 등장하고, 최우선 순위로 수행되는 **VIP 손님**을 구현하였다. 또한 **손님 숫자에 따라 태피의 작업 속도**를 상승시켜 대기 손님이 15명 이상인 경우 태피의 작업 속도 2배, 대기 손님이 30명 이상인 경우 태피의 작업 속도가 3배가 되도록 구현하였다. 마지막으로 **고객 명단을 txt 파일로 저장**돼 하룻동안 태피 상점에 방문한 손님들의 명단을 txt 파일로 확인할 수 있도록 구현하였다.

**2-3 최종 모듈 구현 및 기능**

| **클래스 이름** | **클래스 기능** |
| --- | --- |
| **Customer** | 들어온 손님의 정보를 바탕으로 Customer 객체를 생성 하는 클래스 |
| **MakeQueue** | Customer를 객체로 하는 Person, VIP, Result 큐 생성하는 클래스 |
| **StartTaffy** | 전체적인 프로그램을 수행하는 클래스 |
| **WorkProcess** | 현재 업무 중인지 검사하고, 업무를 수행한 뒤 Result 큐에 저장  손님이 들어오면 Customer 객체를 생성하고 경우에 따라 Person, VIP 큐에 추가하는 클래스 |
| **PrintTot** | 하루동안 입장한 모든 고객의 명단을 콘솔로 출력하고 txt파일로 저장 |

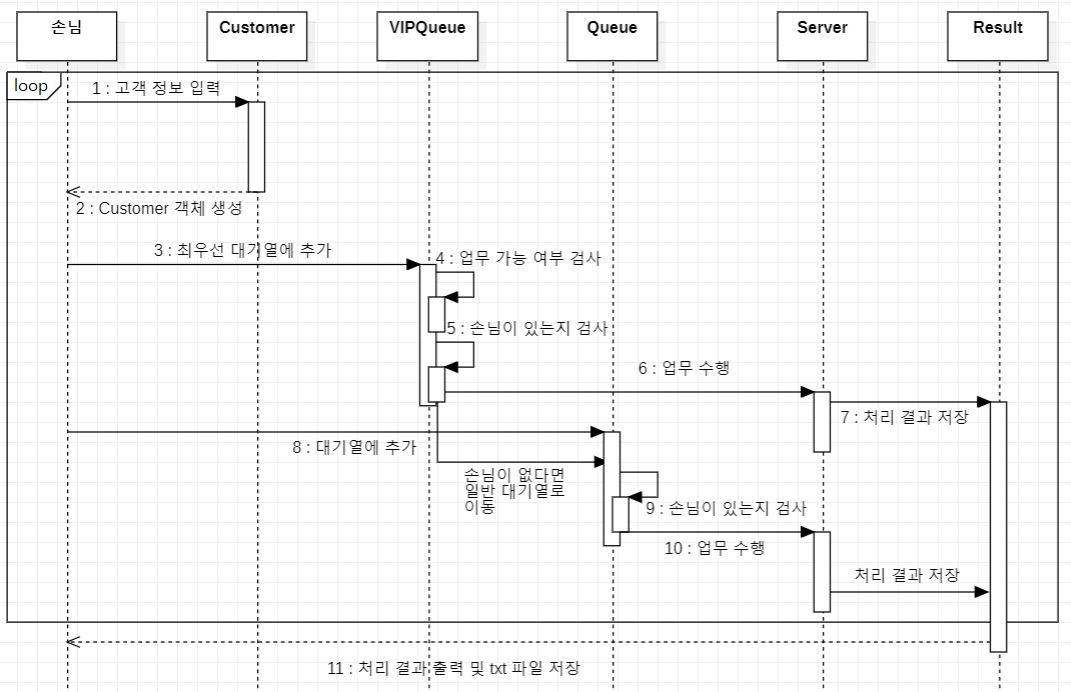
**2-4 프로그램 전체 구조**



프로그램 전체 구조는 다음과 같다. StartTaffy 클래스에서 MakeQueue를 통해 Customer를 객체로 하는 큐를 생성한다. 이때 일반 대기열인 Person, VIP 대기열인 VIP, 결과를 저장할 Result로 총 3개의 큐를 생성한다. StartTaffy 클래스에서 생성한 큐와 변수들을 바탕으로 전체적인 프로그램을 실행한다.

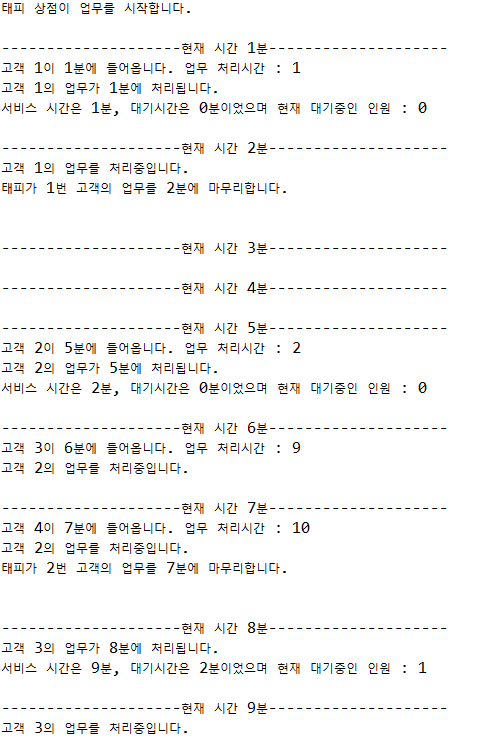
StartTaffy가 실행되면 WorkProcess를 통해 업무 알고리즘이 실행된다. 현재 업무 중인지 IsWorkNow로 검사한 뒤 IsCusEnter로 손님이 입장하는지 여부를 체크한다. 모든 업무가 종료되면 Result 큐에 저장된 모든 손님의 명단을 PrintTot로 넘겨주어 콘솔에 명단을 출력하고, 동시에 txt 파일에 저장한다.

**2-5 프로그램 동작**

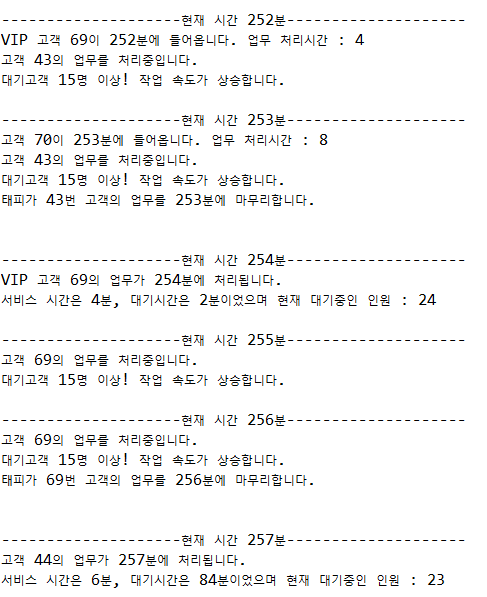


프로그램은 다음과 같은 순서로 동작한다. 손님이 입장하면 손님의 정보를 넘겨주어 Customer 객체를 생성한다. 이때 손님이 VIP인 경우 최우선 대기열에 추가하고 그렇지 않은 경우 일반 대기열에 추가한다. 이후 VIP 큐부터 검사를 시작한다. 현재 태피의 업무가 가능한지 여부를 검사한 뒤 VIP 큐에 손님이 있다면 Dequeue를 해 업무를 수행한 뒤 처리 결과를 Result에 저장한다. 만약 VIP 큐에 손님이 없다면 일반 대기열로 이동해 손님이 있는지를 검사한 뒤 손님이 있다면 업무를 수행하고 처리 결과를 저장한다. 업무 시간이 종료되면 Result에 저장된 모든 손님의 처리 결과를 콘솔로 출력함과 동시에 txt 파일로 저장한다.

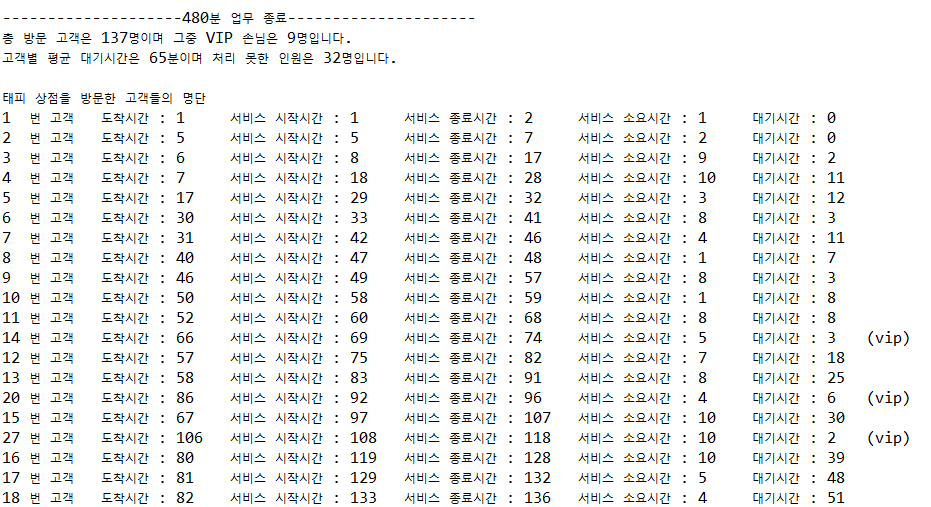
**2-6 최종 구현 결과**

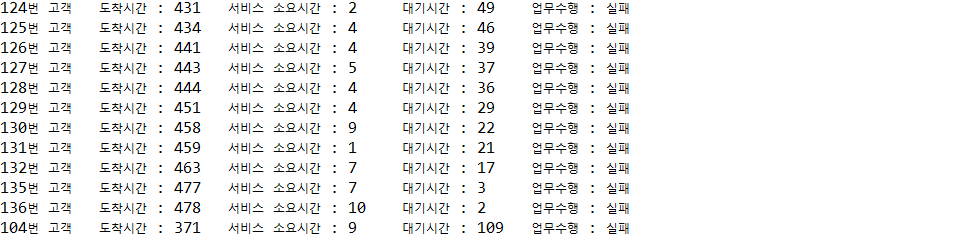


태피 상점이 업무를 시작하면 매 분마다 손님이 들어오는지, 태피가 업무 중인지를 출력한다. 만약 손님이 들어온다면 고객번호와 입장 시간, 업무 처리시간을 출력한다. 고객의 업무가 처리된다면 처리될 고객의 서비스 시간과 대기 시간, 현재 대기중인 인원을 출력한다.



1/50의 확률로 VIP 손님이 입장하게 되면 일반 대기열에 손님이 있더라도 최우선적으로 업무를 수행한다. 또한 대기 고객이 일정 수 이상 존재하게 되면 태피의 작업 속도가 상승하고 이를 출력해 알 수 있게 하였다.

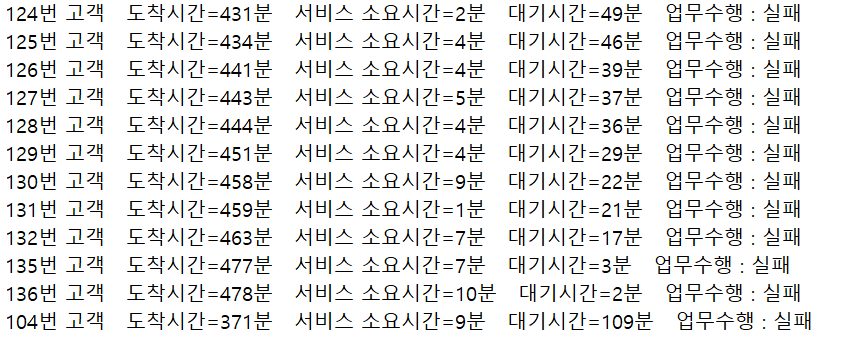




모든 업무가 종료되면 업무 수행 중이던 손님은 기존의 큐에 다시 집어넣는다. 처리되지 못한 손님들의 업무 수행 여부인 doService를 false로 설정한 뒤 Result 큐로 모두 집어넣는다.

통계정보인 총 방문 고객과 그 중 VIP 손님의 수, 고객의 평균 대기시간과 처리하지 못한 손님의 수를 출력한다. 이후 방문한 모든 손님들의 명단을 출력하는데 vip 손님은 알아볼 수 있게 표시하고, 처리되지 못한 손님들은 다른 형식으로 출력해 업무 수행에 실패했음을 알린다.





콘솔로 업무 결과를 출력하는 동시에 생성한 txt 파일에 이어쓰기를 해 태피 상점에 방문한 모든 고객들의 명단을 생성한다.

**2-7 최종 정리 및 논의점**

**프로그램의 문제점**

손님의 업무가 처리되는 도중 업무 시간이 종료되면 해당 고객은 대기열에도 결과 큐에도 속하지 않게 된다. 때문에 해당 문제의 경우 본 프로그램에서는 업무 중에 업무 시간이 종료된 고객의 경우 원래 속하던 큐로 다시 Enqueue를 한다. 이후 대기열에 있는 손님들의 업무가 종료되었는지 판단하는 boolean 값 doService를 false로 설정한 뒤 Result 큐에 입력하는 방식으로 문제를 해결하였다. 이렇게 하면 업무에 실패한 손님들과 성공한 손님을 구분해 다른 형식으로 구분해 출력할 수 있고, 업무 중에 업무 시간이 종료된 손님 문제도 처리할 수 있다.

또한 종업원은 한명인데 1/4의 확률로 서비스 시간이 1~10분인 손님이 매분 들어오니 필연적으로 대기열이 밀릴 수 밖에 없다. 그렇게 되면 태피가 작업중인지 검사했을 때 초반 몇분을 제외하고는 퇴근까지 1초도 쉬지않고 작업해야 한다.그렇기 때문에 종업원의 작업 속도를 늘리거나 종업원의 수를 늘리거나 손님의 수를 줄여야 태피의 부담을 줄일 수 있다.

**효율적인 개선 방안 및 추가 기능**

이러한 태피 상점 프로그램의 개선 방안으로는 종업원의 수를 늘리기, 작업 속도를 늘리기, 입장하는 손님의 수를 줄이는 방법이 있다.

또한 구현하면 좋을 추가 기능으로는 현재까지 받은 손님의 서비스 시간을 계산하여 480분이 됐을때 정시퇴근 할 수 있도록 특정 시점부터는 손님을 받지 않기, 일정 시간 이상 대기한 손님은 기다리다 지쳐 돌아가는 기능, 대기열의 손님 수가 일정 이상이면 30분간 직원이 한 명 추가 등이 있다.

**3. 두번째 문제**

두번째 문제로 선택한 문제는 술 취한 딱정벌레 문제이다.

술취한 딱정벌레 문제의 내용으로는 술 취한 딱정벌레가 같은 크기의 타일이 깔린 방 안에 있다. 이 벌레는 임의의 위치를 선택하여 여기저기 걸어 다니며, 현재 위치에서 주위의 8개 타일로 걸어갈 확률은 동일하다고 가정한다.

술 취한 딱정벌레 문제에서는 문제가 더 세분화되어 있는데

문제 1로써 해당 프로그램의 수행 과정을 GUI로 출력하여야 한다.

문제 2로써 해당 프로그램에 술집을 찾는 n마리의 딱정벌레 문제가 있다.

문제 2의 내용으로써는 n마리의 딱정벌레가 같은 타일 위에 있는 경우, 이 n마리는 더듬이를 통해 술집이 있는 곳의 정보를 주고 받을 수 있다. 술집이 있는 타일을 방문한 딱정벌레도 술집이 있는 곳을 알 수 있다.

**2-1 문제 입출력 명세사항**

결과값으로 출력해야 되는 것은 딱정벌레가 그리는 경로는 어떤 모양일까, 딱정벌레가 방 안 모든 타일을 한 번씩 지나가는데 걸리는 시간은 얼마일까를 여러 번 실행시켜 평균값을 구하고 방의 크기를 변화시키면서 실행하여 방 크기와 시간 사이의 관계를 예측해보는 것이다.

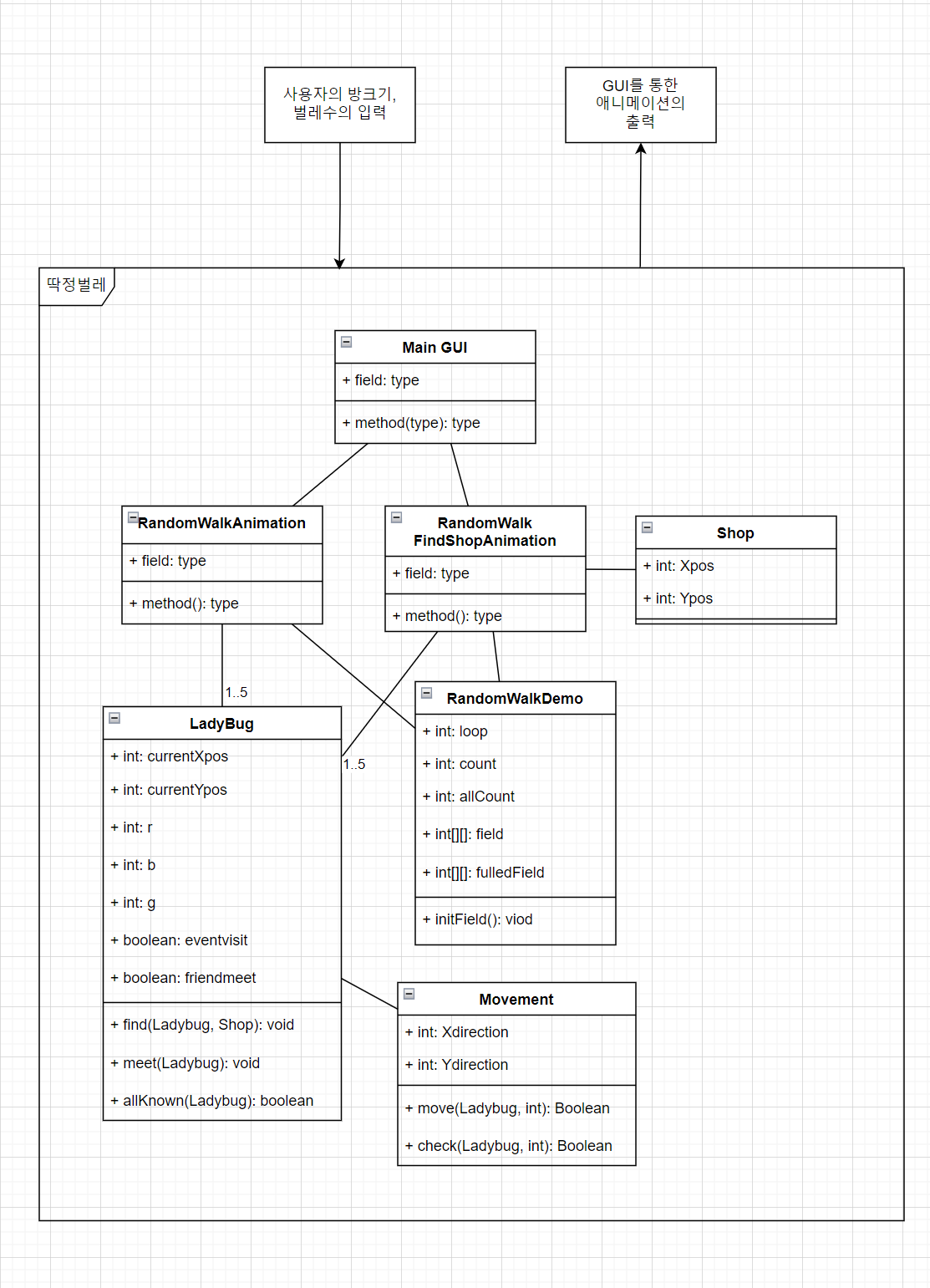
**술집 이벤트의 경우**

임의의 위치에 각각 술집과 n 마리의 딱정벌레를 두고 시작하여 n마리가 모두 술집의 위치를 아는데 걸리는 시간을 구하는 것, 여러번 실행하여 시간의 평균을 구하는 것, 방의 크기를 변화시키면서 통계적으로 유의한 값을 구하는 것, 딱정벌레의 수를 증가시키면서 걸리는 시간의 변화를 측정하고 그 관계를 알아보는 것이 문제의 요구 출력사항이다.

**3-2 최종 모듈 구현 및 기능**

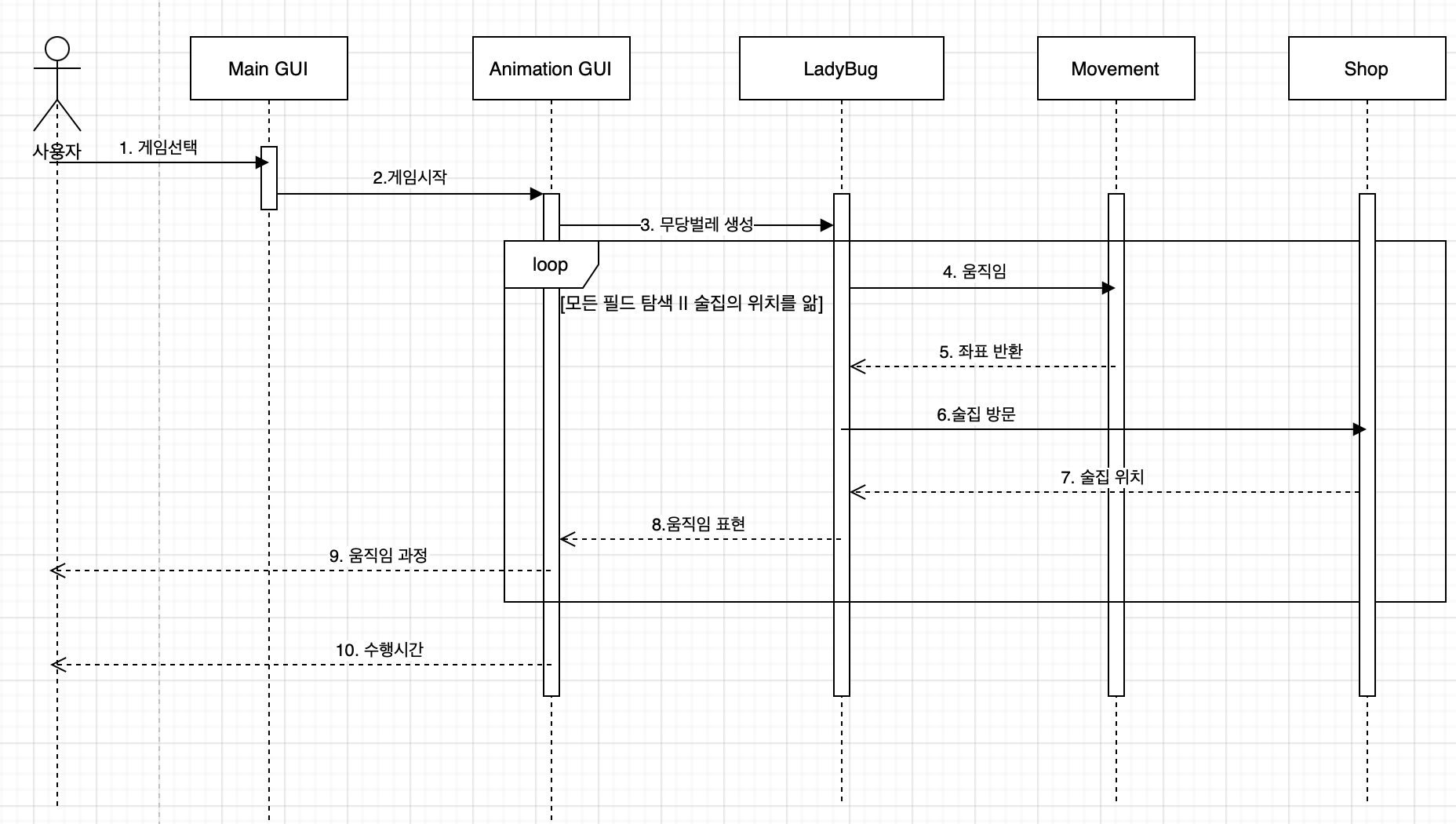
| **클래스 이름** | **클래스 기능** |
| --- | --- |
| **LadyBug** | 딱정벌레 객체를 생성하고 현재 위치와 Shop의 위치를 아는지에 대한 정보를 가지고 있고 Shop의 위치를 모르는 딱정벌레를 만날 경우 위치 정보 전달 |
| **Movement** | RandomWalk 알고리즘을 이용해 LadyBug의 움직임을 수행  벽을 뚫고 지나가지 않도록 예외처리도 담당 |
| **Shop** | 술집 문제 선택시 술집의 위치를 저장하는 클래스 |
| **RandomWalkDemo** | 콘솔 상으로 알고리즘을 테스트및 10회 실행시의 평균 소요시간을 계산하는 클래스 |
| **Main GUI** | 수행할 문제와 방의 넓이, 딱정벌레의 갯수를 사용자로부터 입력 받는 GUI 클래스 |
| **RandomWalkAnimation** | 딱정벌레의 방탐색 문제 해결과정을 애니메이션으로 출력하는 GUI 클래스 |
| **RandomWalkFindShopAnimation** | 딱정벌레의 술집찾기 문제 해결과정을 애니메이션으로 출력하는 GUI 클래스 |

**3-3 프로그램 전체 구조**



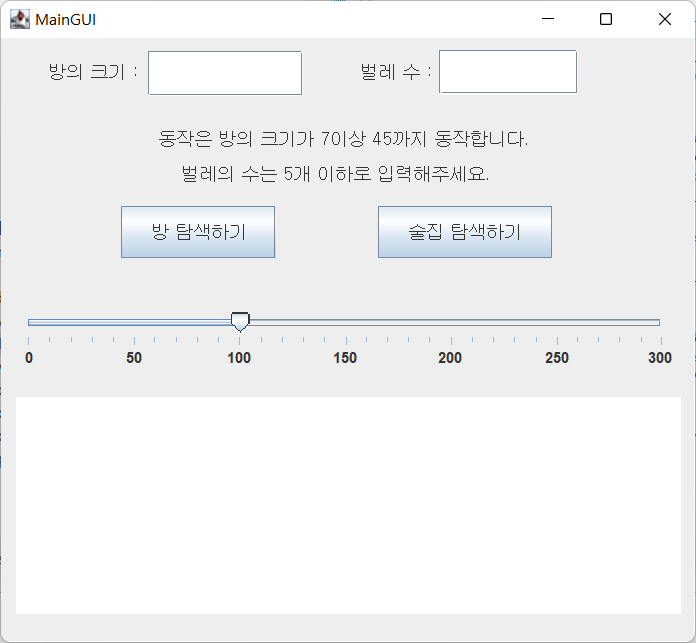
다음은 클래스다이어그램과 프로그램의 입출력을 명세하여 나타낸 자료이다. 메인 GUI에서는 사용자의 방크기와 벌레의 수를 입력받고 RandomWalkAnimation 클래스와 RandomWalkFindShopAnimtion 클래스중 하나를 선택하게 된다. 두 Animation 클래스에서는 LadyBug 객체가 생성되는데 다형성에 의해서 1~5마리의 마리의 객체가 생성된다. 각각의 LadyBug객체는 Movement 클래스에서 move() 메서드를 통해서 움직임을 구현하고 만약 필드의 범위를 벗어나게 될 경우 오류가 발생하기 때문에 check() 메서드를 통해서 객체들이 허용되는 범위 내에서 이동하는 지를 검사한다. LadyBug객체들은 find()메서드를 통해 Shop 객체 방문여부를 검사하고, 이미 방문한 객체와 아직 방문하지 못한 객체가 필드 위 하나의 좌표에서 만나게 되면 Meet()메서드를 통해 boolean 타입의 friendmeet가 true값으로 변환되어 술집의 위치를 아는것으로 간주한다. RandomwalkDemo 클래스는 평균적인 수행시간을 구하기 위한 클래스로써 선택한 Animation 클래스를 내부적으로 수행한 후 수행시간들의 평균을 구하여 Main GUI에 반환함으로서 사용자가 확인할 수 있도록 한다.

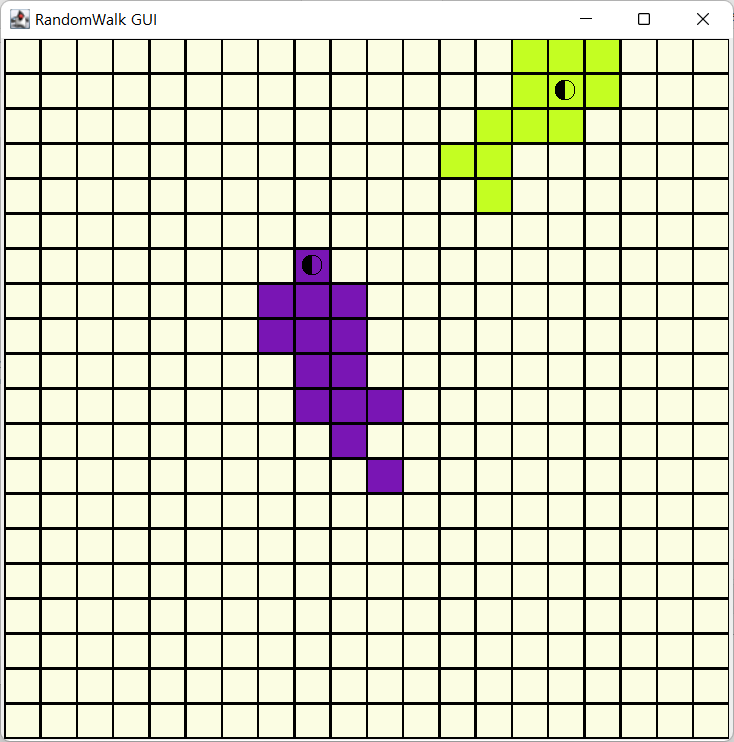
**3-4 프로그램 동작**

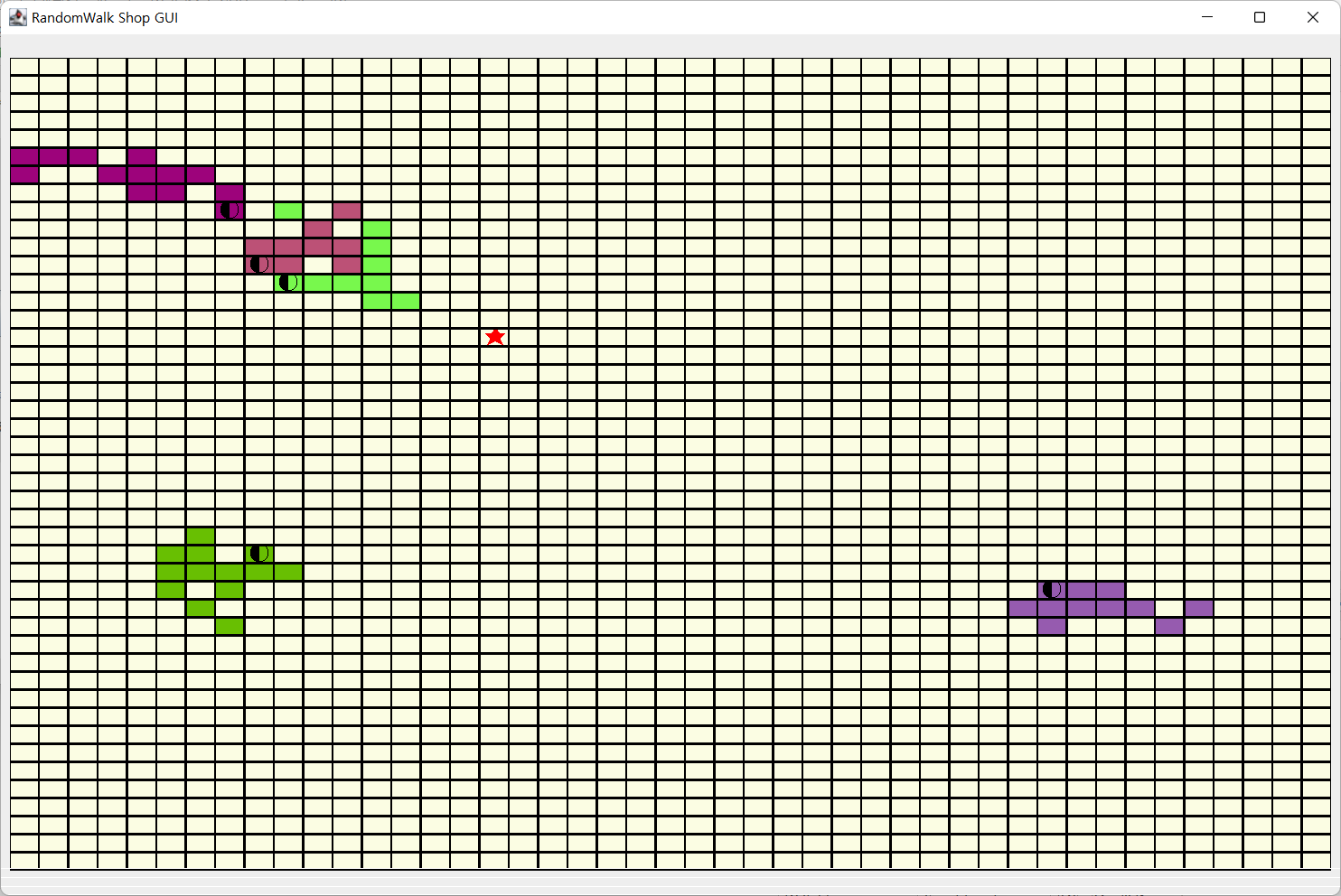


3-3에 기술한 모듈들의 상호작용을 시퀀스 다이어그램을 통해서 나타내면 사용자가 벌레의 수와 방의 크기를 입력한 후 게임을 시작하면 입력받은 값을 토대로 무당벌레 객체가 생성되고 한 번 움직일때마다 좌표를 변경하여 사용자에게 보여줌으로써 움직임 과정을 나타낸다. 이 모든 과정은 게임의 완료 조건인 모든 필드 탐색 호은 모든 객체가 술집의 위치를 알때까지 반복하며 완료 조건을 만족하여 프로그램의 수행이 끝나게 된다면 수행시간을 사용자에게 반환한다.

**3-5 최종 구현 결과**

****

****

****

최종 구현 결과의 캡쳐본 일부이다. MainGUI에서 사용자에게 방크기와 벌레의 갯수를 입력받아 사용자가 선택한 이벤트를 애니메이션을 통해 보여준다.

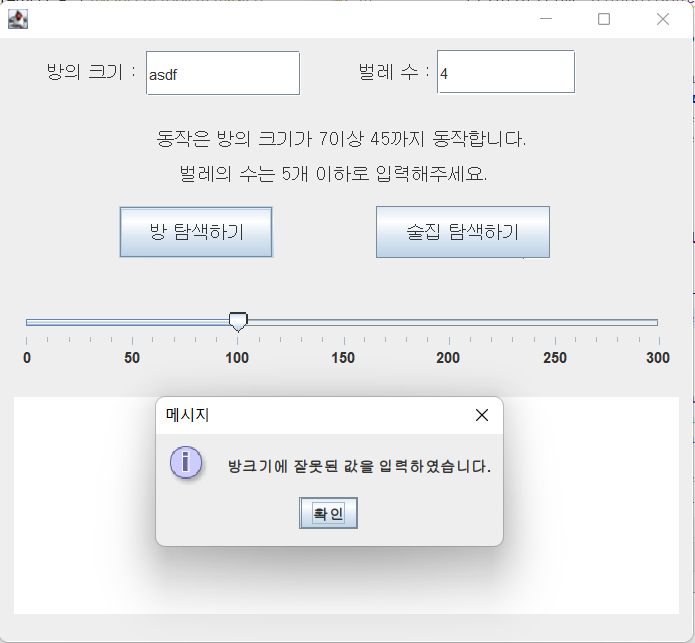
또한 사용자가 딱정벌레의 움직임을 슬라이더를 통해 속도를 조절 할 수 있게 하였으며 딱정벌레의 지나간 자리를 표시하는 딱정벌레 객체마다의 고유 색상은 프로그램을 실행 할 때마다 랜덤하게 표현되도록하였다.

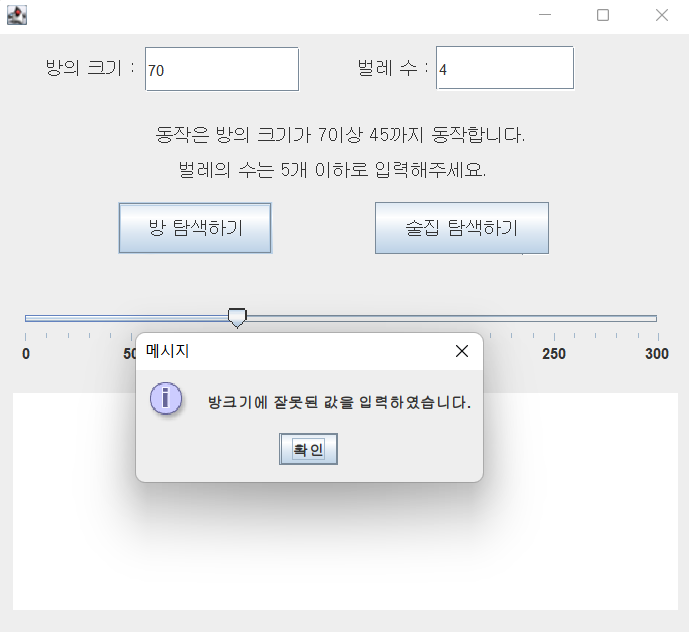
방찾기에서는 딱정벌레가 모두 방을 탐색하면 애니메이션은 종료되며 MainGUI의 TextArea에 애니메이션에서 탐색한 횟수, 프로그램 내부에서 방탐색을 9번 실행하여 애니메이션에서 탐색한 횟수를 더해 10번의 평균값을 출력한다.

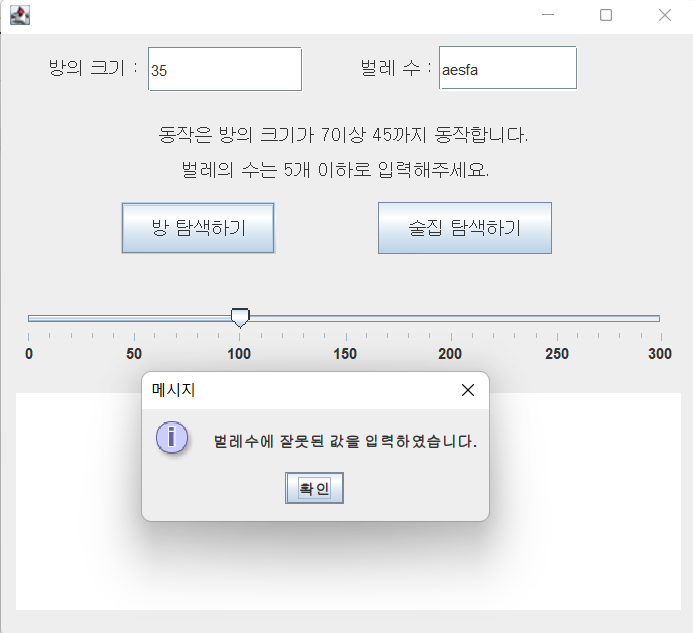
술집이벤트를 선택할 경우 애니메이션에서 빨간색 별로 술집의 위치를 표현하여 사용자가 술집이 어디 있는지 알 수 있게하였고 모든 무당벌레들이 술집을 찾거나 다른 벌레를 통해 술집의 위치를 알게 될 경우 MainGUI에서 이러한 이벤트 발생을 알 수 있도록 텍스트로 몇번 벌레가 술집을 직접 찾았는지 술집의 위치를 알고 있는 어떤 벌레와 모르고 있는 다른 벌레가 만났을 경우 술집의 위치를 알려주는 이벤트 또한 출력하게 하였다.

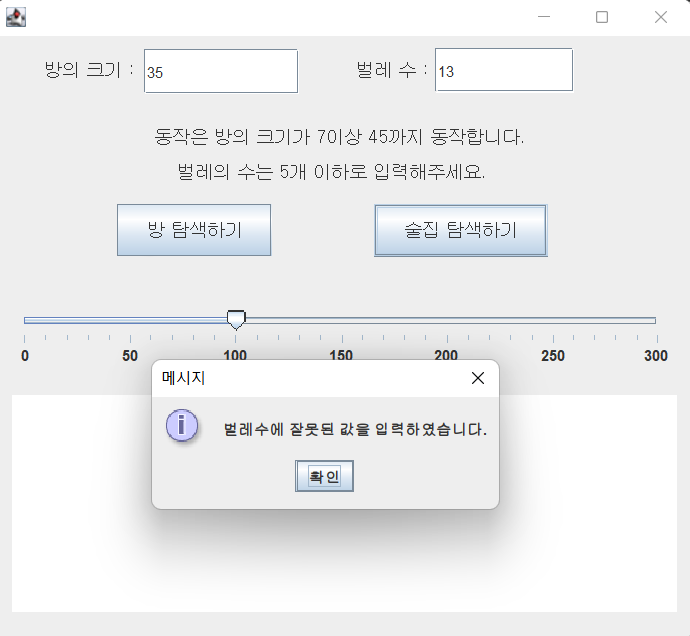
술집 이벤트에서도 동일하게 프로그램 내부적으로 9번 실행하고 애니메이션 소요시간을 더하여 10번의 실행 평균값을 출력한다.

또한 입력창에 문자열이나 범위를 벗어나는 값을 입력하면 경고창이 뜨고 실행이 되지않게 예외처리를 해주어 프로그램의 오작동을 방지하였다.









[딱정벌레 최종구현 동영상 JAVA swing RandomWalk Animation](https://www.youtube.com/watch?v=TFUhdr73IEQ)

위 링크는 최종 구현이 완료된 프로그램의 작동을 영상으로 녹화하여 프로그램의 동작을 볼 수 있는 영상이다. 최종 구현결과는 위 영상을 통하여 자세히 확인할 수 있다.

**2-6 최종 정리 및 논의점**

**1. 방의 크기를 변화시키면서 통계적으로 유의한 값은?**

- 방의 크기가 증가할수록 방탐색 문제와 술집 찾기 두 문제에서는 수행시간과 10번의 수행시간 평균이 **방의 크기가 넓어질수록 수행시간이 비례적으로 증가하였다.**

**2. 딱정벌레의 수를 증가시키면서 걸리는 시간의 변화를 측정하고 그 관계를 알아보자**

- 딱정벌레의 수를 증가 시키면 방탐색 문제에서는 수행시간이 **딱정벌레의 수에 반비례하며 수행시간이 줄어들었다.**

- **하지만 술집 이벤트에서는 딱정벌레의 수에대해 수행시간과의 연관성을 찾기가 어려웠다.**

이러한 술집 이벤트의 비례적이지 않은 수행시간의 연관성을 생각해보았다. 술집 이벤트의 경우 방을 모두 탐색하지 않고 술집을 만약 일찍 찾게되고 술집을 찾지 못한 친구를 빠르게 만나게 된다면 프로그램이 일찍 종료 될 것이고 반대로 술집을 느리게 찾고 술집을 찾은 다른 친구를 빨리 만나지 못한다면 프로그램의 수행시간은 많이 늘어날 것이다. 따라서 방탐색과는 달리 술집 찾기 이벤트의 경우 **Random 알고리즘 중에서도 Random한 값의 영향을 많이 받게되어 수행시간에 많이 영향을 주어 수행시간의 균일성을 찾지 못하게 되었다.**

**따라서 수행시간이 매우 빨라지거나 수행시간이 매우 느려지거나 두 가지의 케이스가 자주 도출되었다.**

이외의 프로그램과 이번 과제에서의 아쉬웠던 사항으로는 구현에 충분한 여유가 있지 않아 다양한 추가기능 구현이나 프로그램 코드의 완성도를 많이 올리지 못한 것 같아 아쉬움이 있다.

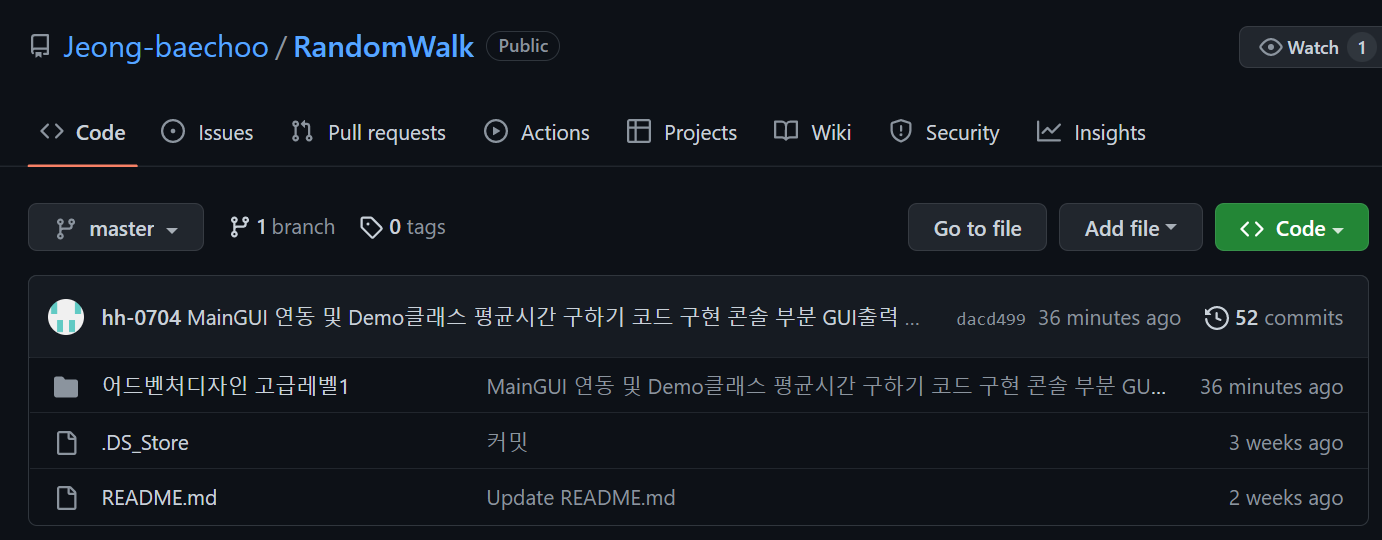
이번 딱정벌레 과제에서 추가 구현으로 RandomWalk 알고리즘을 활용해 모의주식투자 게임도 만들고 싶었지만 메인 과제인 딱정벌레 문제와 연관성을 뚜렷하게 찾지 못하여 구현을 하지 않았다. 하지만 나중에 시간이 날때 한번 구현해보고 싶은 마음이 있다.

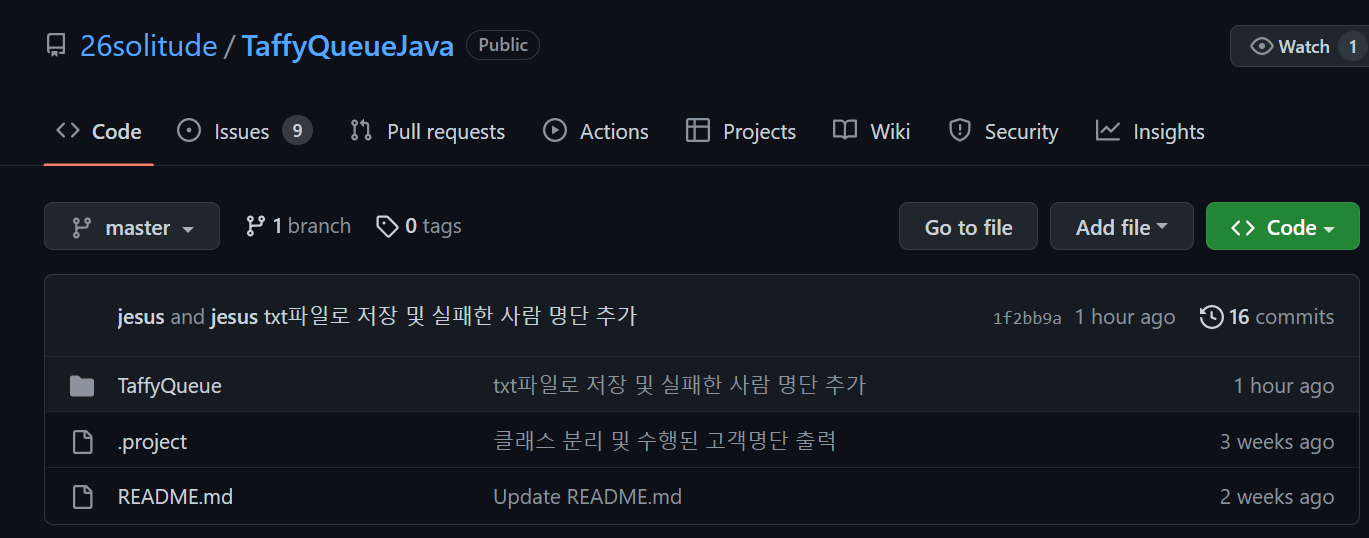
**4. 역할분담**

| 조장호 | 태피 상점 구현 및 추가기능 구현 | |
| --- | --- | --- |
| 정용환 | 술 취한 무당벌레 구현 및 추가 기능 구현 | |
| 허환 | 술 취한 무당벌레 구현 및 추가 기능 구현 | |

**5. 팀프로젝트 협업**

팀프로젝트는 **깃허브**를 통하여 이루어졌고 각자 맞은 프로젝트를 깃허브에 올린 후 서로 만든 코드를 리뷰하며 나중에 유지보수와 커뮤니케이션의 효율성을 높이기 위하여 이전에 수강하였던 **클린코드** 강의를 참고하여 코드의 가독성을 높혔다.

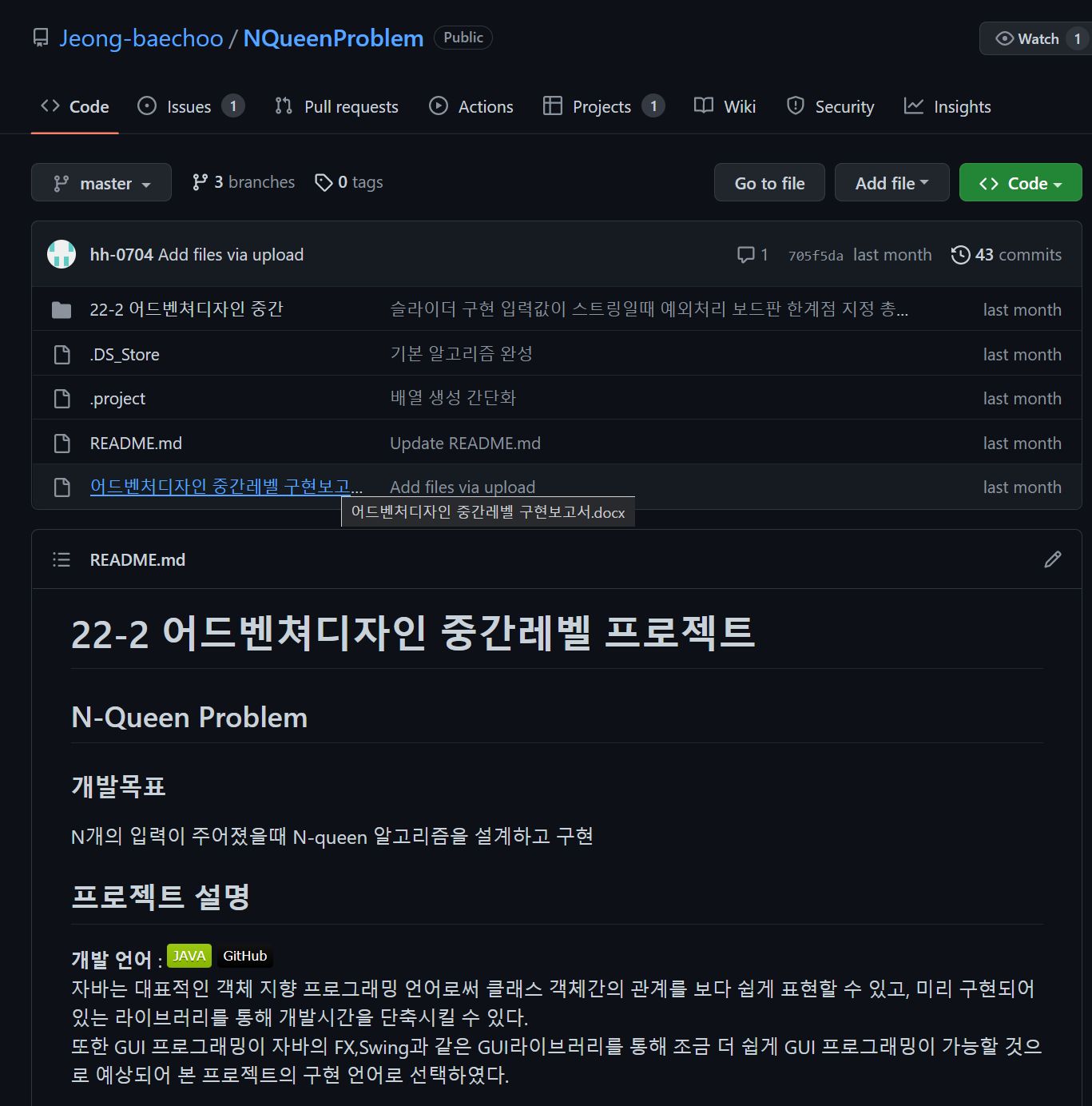


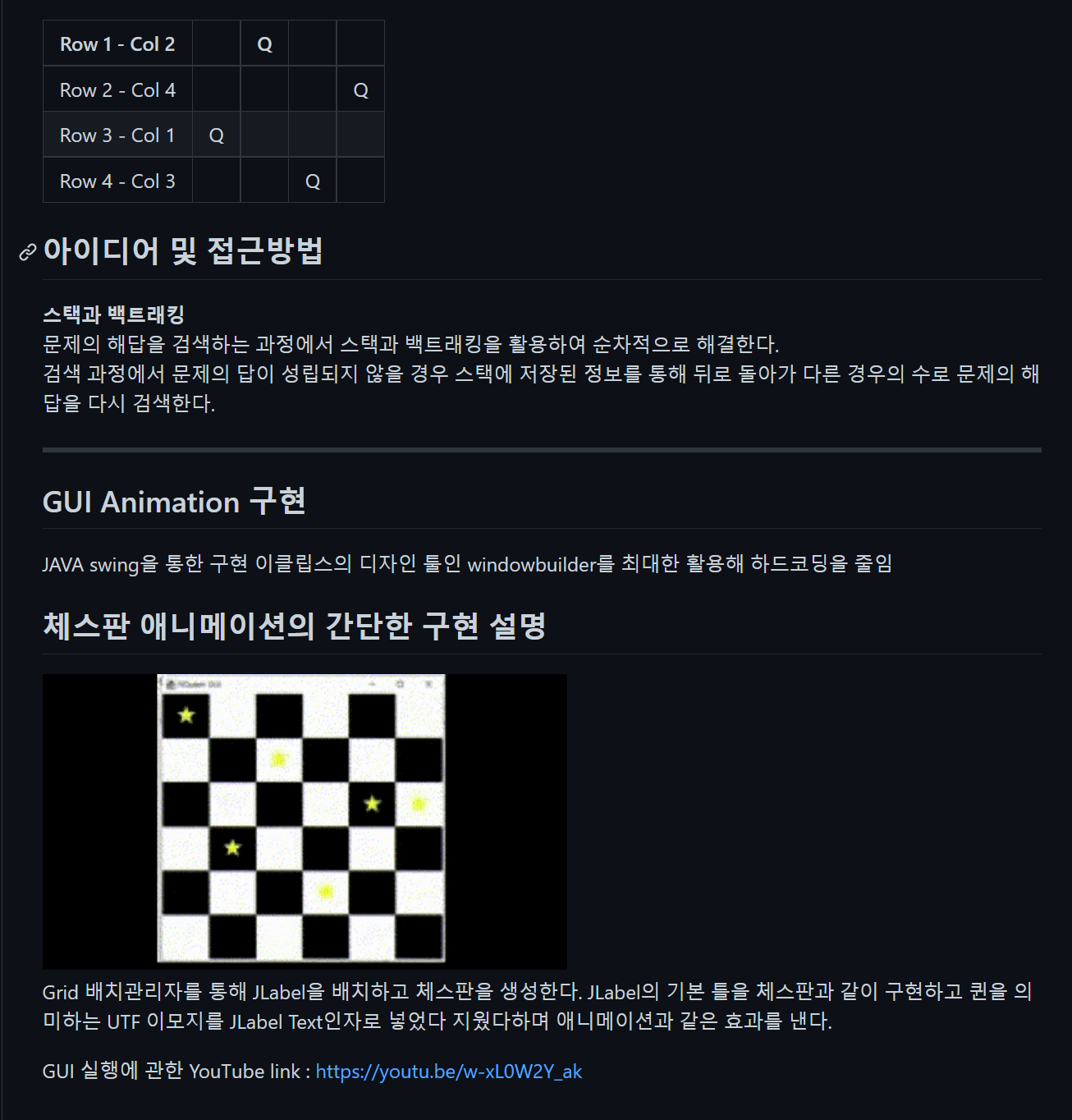


위와 같이 이번 프로젝트에서 구현하는 프로그램이 두개이다 보니 Git 레포지토리도 두개 모두 생성하여 프로그램을 구현하는데 Git을 많이 활용하게 되었다. 이렇게 Git을 활용하다보니 취업에 도움될 수 있는 포트폴리오를 하나씩 생성해나가게 된 것 같아 학기를 끝내고 돌아보니 성취감이 있었다.

또한 회사현업에서 협업툴로 Git을 많이 쓰는데 Git을 자주 많이 사용하게 되다 보니 사용법이나 오류에 관한 대처방법, 협업 방식에 대해 많이 알아가는 시간이 되었던 것 같다.

또한 중간레벨 Git 레포지토리 또한 과제의 마지막 최종보고서를 레포지토리에 추가 업로드하여 프로그램의 설명과 의도가 충분히 설명이 될 수 있게하였고 README 파일또한 깔끔하게 만들어 레포지토리를 보는 이들에게 이것이 무엇을 하는 프로그램인지 어떤 의도와 문제를 어떻게 풀었는지 한눈에 간략히 알 수 있도록 만들었다.





이러한 점을 이번 고급레벨에서도 시험과 시간이 나는데로 똑같이 최종보고서 업로드와 README 파일의 깔끔한 구성으로 취업 포트폴리오에 유용하게 쓸 수 있도록 할 예정이다.

**6. 참고문헌**

**6-1 태피상점 문제**

https://velog.io/@moomyung1013/%ED%83%9C%ED%94%BC%EC%83%81%EC%A0%90%EC%9D%98-%ED%81%90-%EC%8B%9C%EB%AE%AC%EB%A0%88%EC%9D%B4%EC%85%98

https://github.com/moomyung1013/Taffy-Store-Simulate

https://velog.io/@junsu1222/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%ED%81%90-%EC%9D%91%EC%9A%A9%EC%9D%80%ED%96%89-%EC%84%9C%EB%B9%84%EC%8A%A4-%EC%8B%9C%EB%AE%AC%EB%A0%88%EC%9D%B4%EC%85%98-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%A8

https://lifelife7777.tistory.com/6

https://hi-pinpoint.tistory.com/15

<https://st-lab.tistory.com/184>

**6-2 무당벌레 문제**

https://quickgrid.blogspot.com/2016/09/Simple-NQueen-Visualizer-in-JAVA-to-Visualize-Backtracking.html

<https://codereview.stackexchange.com/questions/34054/random-walk-on-a-2d-grid>