2024 Spring OOP Assignment Report

과제 번호 : Assignment #5
학번 : 20230563
이름 : 김홍근
Povis ID : hongsimi7

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. 프로그램 개요

- 본 프로그램은 class와 class 상속을 이용한 객체 지향 프로그래밍을 통해 텍스트 기반의 '구스구스덕'을 모티브로 한 게임이다.
- Class를 통해 게임 속의 다양한 객체를 생성하고 관리한다.
- 생성자와 소멸자를 통해 객체가 생성되고 소멸될 때에 객체를 관리한다.
- 상속을 통해 효율적으로 class를 생성하고 객체를 관리한다.
- Bird 클래스는 abstract클래스로 모든 새의 상위 클래스이며, 새의 필요한 정보를 담는다.
- Goose, Duck, AssassinDuck, Falcon, DodoBird, DetectiveGoose, MorticianGoose 클래스는 각각의 직업의 클래스로 각 직업 특징마다 다른 정보를 담는다.
- BirdList, BirdNode 클래스는 모든 플레이어의 새를 Linked list 형식으로 담는다.
- GGD 클래스는 전반적인 게임 플레이를 진행시키는 요소를 담고 있는 클래스이다.

2. 프로그램의 구조 및 알고리즘

- [Class Bird]

1. Bird 클래스는 모든 새의 abstract 클래스로 job, player_name, kill, dead, elect, killability, electeddead, pointed 멤버변수는 각각 플레이어의 직업, 이름, 살조 여부, 사망 여부, 투표 당한 횟수, 라운드 당 살조 가능 횟수, 투표사 여부, 가장 많은 투표를 당한 새 여부이다.
2. 생성자는 살조 여부, 사망 여부, 투표사 여부, 가장 많은 투표를 당한 새 여부를 false로 설정하고, 투표수 또한 0으로 설정한다.

3. SetPlayerName(std::string name): 플레이어의 이름을 설정한다.
4. GetPlayerName() const: 플레이어의 이름을 반환한다.
5. GetPlayerKill() const: 플레이어가 다른 플레이어를 죽였는지 여부를 반환한다.
6. GetPlayerjob() const: 플레이어의 직업을 반환한다.
7. GoDead(): 플레이어의 상태를 사망으로 설정한다.
8. checkdead(): 플레이어가 죽었는지 여부를 반환한다.
9. elected(): 플레이어의 투표 수를 증가시킨다.
10. resetelect(): 플레이어의 투표 수를 초기화한다.
11. checkelect(): 플레이어가 받은 투표 수를 반환한다.
12. windead(): 플레이어의 상태를 투표사로 설정한다.
13. checkelectdead(): 플레이어의 투표사 여부를 반환한다.
14. check_killability(): 플레이어의 살조 횟수를 반환한다.
15. change_killability(int x): 플레이어의 살조 횟수를 변경한다.
16. change_kill(): 플레이어의 살조 상태를 초기화한다.
17. electpointed(): 플레이어가 가장 많은 투표를 받았음을 표시한다.
18. resetpointed(): 가장 많은 투표를 받은 상태를 초기화한다.
19. checkpointed(): 플레이어가 가장 많은 투표를 받았는지 여부를 반환한다.
20. virtual Skill(BirdList*)은 가상 함수로 상속되었을 때 직업마다 각기 다른 스킬이 사용되도록 한다.

□ [class Goose]

1. Goose 클래스는 Bird 클래스를 상속받은 클래스이며, 능력이 없는 클래스이다.
2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 5로 설정한다.
3. Skill(BirdList*): 거위에게는 아무 능력이 없기 때문에 이에 대한 문구만 출력한다.

□ [Class DetectiveGoose]

1. DetectiveGoose 클래스는 Bird 클래스를 상속받은 클래스이며, '조사'라는 스킬을 사용

할 수 있는 클래스이다.

2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 3으로 설정한다.

3. Skill(BirdList*): 탐정 거위의 스킬을 정의하는 메서드로, 해당 라운드의 조사하고 싶은 플레이어의 살조 여부를 확인할 수 있는 메서드이다.

□ [Class MorticianGoose]

1. MorticianGoose 클래스는 Bird 클래스를 상속받은 클래스이며, '염습'라는 스킬을 사용할 수 있는 클래스이다.

2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 4으로 설정한다.

3. Skill(BirdList*): 장의사 거위의 스킬을 정의하는 메서드로, 해당 라운드의 염습하고 싶은 플레이어의 직업 여부를 확인할 수 있는 메서드이다. 이 때, 죽은 플레이어가 없다면 사용할 수 없다.

□ [Class Duck]

1. Duck 클래스는 Bird 클래스를 상속받은 클래스이며, '살조'라는 스킬을 사용할 수 있는 클래스이다.

2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 2으로 설정하고 라운드 당 살조 가능 횟수는 1로 초기화한다.

3. Skill(BirdList*): 오리의 스킬을 정의하는 메서드로, 해당 라운드에 죽이고 싶은 플레이어를 죽이는 메서드이다. 이 때, 라운드 당 살조 가능 횟수를 넘겼다면, 스킬을 사용할 수 없다.

□ [Class AssassinDuck]

1. AssassinDuck 클래스는 Bird 클래스를 상속받은 클래스이며, 암살 가능 횟수를 뜻하는 assassinability라는 멤버 변수로 가지고 있고, '살조'와 '암살'이라는 스킬을 사용할 수 있는 클래스이다.

2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 1으로 설정하고 라운드 당 살조 가능 횟수는 1로, 암살 가능 횟수를 2로 초기화한다.

3. Skill(BirdList*): 암살자 오리의 스킬을 정의하는 메서드로, '암살'을 진행한 뒤, '살조'가 진행되는 메서드이다. '암살'은 한 플레이어를 선택하여 그 플레이어의 직업 번호를 고른다. 만약 직업 번호가 맞을 경우 그 플레이어를 죽이지만, 반면에 틀린다면 자기 자신이 죽는다. 암살은 게임당 2번 사용할 수 있으며, 만약 횟수가 넘어가면 사용하지

못한다. '살조'는 Duck클래스의 '살조' 스킬과 마찬가지로 해당 라운드에 죽이고 싶은 플레이어를 죽이는 메서드이다. 이 때, 라운드 당 살조 가능 횟수를 넘겼다면, 스킬을 사용할 수 없다.

□ [Class Falcon]

1. Falcon 클래스는 Bird 클래스를 상속받은 클래스이며, '살조'라는 스킬을 사용할 수 있는 클래스이다. 또한, 다른 조류와는 다르게 투표는 무효표만 가능한 클래스이다.
2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 0으로 설정한다.
3. Skill(BirdList*): 오리의 스킬을 정의하는 메서드로, 해당 라운드에 죽이고 싶은 플레이어를 죽이는 메서드이다.

□ [class DodoBird]

1. DodoBird 클래스는 Bird 클래스를 상속받은 클래스이며, 능력이 없는 클래스이다. 그러나 만약 투표로 죽게 된다면, 도도새가 승리하는 클래스이다.
2. 생성자는 Bird 클래스를 상속받아 초기화하고, 직업 번호를 6로 설정한다.
3. Skill(BirdList*): 거위에게는 아무 능력이 없기 때문에 이에 대한 문구만 출력한다.

□ [Class BirdList]

1. Bird 객체들을 리스트 형태로 관리하기 위해 만들어진 Class이다. 멤버 변수로 head, tail이 있고, 각각 첫 번째 노드와 마지막 노드를 가리키는 용도로 사용된다.
2. 생성자에서 head, tail을 각각 NULL로 초기화 한다.
3. GetHead() const: 리스트의 첫 번째 노드를 반환한다.
4. GetTail() const: 리스트의 마지막 노드를 반환한다.
5. AddBirdNode(BirdNode * node): 새 노드를 리스트에 추가합니다. 리스트가 비어 있는 경우, 새로운 노드를 헤드와 테일로 설정합니다. 그렇지 않은 경우, 현재 테일 노드의 다음 노드로 설정하고, 테일을 업데이트한다.

□ [Class BirdNode]

1. BirdNode 클래스는 BirdList의 노드 역할을 하는 클래스로 멤버 변수로 Bird 객체를 가리키는 bird와 다음 노드를 가리키는 next를 가지고 있다.

2. 생성자는 두 멤버 변수를 NULL로 초기화 한다.
3. 소멸자는 bird 포인터가 가르키는 Bird 객체를 삭제하는 역할을 한다.
4. SetBird(Bird bird): 노드의 bird 포인터를 설정한다.
5. SetNext(BirdNode next): 노드의 next 포인터를 설정한다.
6. GetBird() const: 노드의 bird 포인터를 반환한다.
7. GetNext() const: 노드의 next 포인터를 반환한다.

□ [Class GGD]

1. GGD 클래스는 게임의 전체적인 흐름을 관리하는 클래스이다. 이 클래스는 게임의 설정, 플레이어 추가, 라운드 진행, 게임 종료 여부 확인, 게임 결과 출력, 라운드 당 오리 살조 횟수 제한의 기능을 한다. 멤버 변수로는 bird_list, killability 각각 게임ㅇ 참여하는 플레이어의 리스트를 가리키는 포인터와 오리의 살조 가능 횟수이다.
2. 생성자에서 bird_list는 동적할당되고, killability는 1로 초기화된다.
3. GameStart(): 게임을 시작하는 메서드로, 게임 설정 메뉴를 제공하고, 게임 시작 조건을 확인한다. select (int)는 사용자의 메뉴 선택을 저장하는 변수이고, game_start (bool)은 게임 시작 여부를 나타내는 플래그이다. 초기값은 false로 설정된다. Duck_count, Goose_count, Falcon_count, dodobird_count(int)는 각각 게임 내의 오리, 거위, 송골매, 도도새의 수를 저장하는 변수이다. current는 새 리스트를 순회하기 위한 포인터이다. 1을 선택했다면, AddPlayer 메서드를 호출한다. 2을 선택했다면, Changekillnumber 메서드를 호출한다. 3을 선택했다면, 전체 게임의 새의 수를 카운트한다음, 플레이어가 없거나 어떤 진영의 승리가 정해져 있다면, 게임을 진행하지 않고, 그렇지 않다면 게임을 진행한다.
4. RoundProgress(): 한 라운드를 진행하는 메서드로, 각 새의 스킬을 실행하고, 투표 과정을 처리한다. Count(static int)는 라운드 카운트를 저장하는 정적 변수로 초기값은 1이다. game_killability(int)는 라운드 당 오리의 살조 가능 횟수를 저장하는 변수이다. current (BirdNode*)는 리스트를 순회하기 위한 포인터이다. abstain (int)는 무효표 수를 저장하는 변수이다. mostelected (BirdNode*)는 가장 많은 투표를 받은 새를 가리키는 포인터이다. 먼저 라운드를 출력하고, 각 플레이어의 직업 순서별로 스킬을 실행한다. 만약 사망했을 시에는 스킬이 발동하지 않는다. 그리고 플레이어가 등록된 순서대로 투표를 진행한다. 이 때, 송골매인 경우에는 자동으로 무효표가 되고 나머지는 투표 여부를 묻고 투표를 진행한다. 가장 많은 투표

를 받은 새는 mostelected 포인터가 가리킨다. 그리고 동일한 투표 수를 처리하고 무효표가 같거나 더 많은지에 대해 처리를 하여 추방 여부를 결정한다. 만약 도도새가 추방되었다면 도도새에게 windead 메서드가 실행될 수 있도록 한다. 그리고 다음 라운드를 위해 플레이어의 멤버 변수를 초기화한다.

5. IsGameOver(): 게임이 종료되었는지 확인하는 메서드이다. Res(bool)은 게임 종료 여부를 확인하는 변수로, false을 초기값으로 한다. current (BirdNode*)는 리스트를 순회하기 위한 포인터이다. Duck_count, Goose_count, Falcon_count, dodobird_count(int)는 각각 게임 내의 오리, 거위, 송골매, 도도새의 수를 저장하는 변수이다. 만약 도도새가 투표로 죽었다면, res를 true로 변경 후, 리턴한다. 그렇지 않다면 이 때 살아있는 새만 카운트하고, 각 진영의 승리가 만족한다면, res를 true로 하고 리턴한다. 그렇지 않다면 그냥 res를 리턴한다.

6. PrintGameResult(): 게임의 결과를 출력하는 메서드이다. current (BirdNode*)는 리스트를 순회하기 위한 포인터이다. Duck_count, Goose_count, Falcon_count, dodobird_count(int)는 각각 게임 내의 오리, 거위, 송골매, 도도새의 수를 저장하는 변수이다. 만약 도도새가 투표로 죽었다면, 도도새의 승리를 출력한다. 만약 오리와 송골매가 아무도 살아있지 않다면, 거위의 승리를 출력한다. 만약 오리가 1명 이상이고, 오리의 수가 나머지 직업 새보다 많거나 같다면, 오리의 승리이지만, 오리, 송골매가 각각 한 마리 살아남아 총 두 마리라면, 송골매의 승리를 출력한다. 만약 송골매가 살아있고 나머지 직업의 새가 송골매의 수보다 작거나 같다면, 송골매의 승리를 출력한다.

7. AddPlayer(): 플레이어를 추가하는 메서드이다. newplayer (BirdNode*)는 새롭게 추가될 플레이어를 나타내는 노드 포인터이다. player_name (std::string)은 사용자로부터 입력받은 플레이어의 이름을 저장하는 변수이다. role_code (int)은 사용자로부터 입력받은 역할 번호를 저장하는 변수이다. newrole (Bird*)은 생성된 플레이어 역할 객체를 가리키는 포인터이다.

8. Changekillnumber(): 라운드 당 오리의 살조 횟수를 변경하는 메서드이다.

3. 토론 및 개선

□ 클래스를 정의하는 방법 및 클래스를 통해 객체를 생성하여 유용하게 사용하는 방법에 대해 알 수 있었다. 클래스에는 private와 public뿐만 아니라, protected라는 개념을 알게 되어 각각의 정의와 특징을 다시 한 번 재정립할 수 있었다.

□ 클래스 상속을 통해 Bird 클래스를 상속해 많은 직업의 새를 효율적으로 프로그래밍을 할 수 있었다.

□ 생성자를 통해 멤버 변수를 초기화할 수 있도록 했다.

□ GGD에서 search 함수를 만들어 더 효율적으로 코드를 짤 수 있었을 거라 생각했다. 따라서 한 번 구현해보는 것도 좋을 것 같다.

4. 참고 문헌

□ 해당사항 없음.