Term Project - Bowling

소프트웨어프로젝트 05분반



학과: 경영학과

학번: 20134789

이름: 김진관

**1. 개요**

2020년 1학기 소프트웨어 학부 소프트웨어 프로젝트 텀프로젝트 주제로 볼링 게임을 선택하고 JAVA로 프로그래한다. 실제 스포츠를 하는 것과 유사하게 시나리오를 구성하고, 진행 및 결과를 보여준다.

Github: <https://github.com/kkole3897/Bowling-Game-with-Java>

**주요 특징**

- 볼링의 일반적인 규칙과 점수 계산 방식을 그대로 적용한다..  
- 2명의 플레이어가 10프레임을 돌아가면서 진행한다.  
- 점수판을 실시간으로 업데이트하면서 보여준다.  
- 플레이어의 등급을 나눠 등급마다 얻는 점수의 차이를 준다.  
- 볼링핀은 앞에 위치한 볼링핀부터 차례대로 쓰러지도록 간소화한다.  
- 10프레임이 모두 진행된 경우 게임을 종료하고, 이긴 플레이어를 알려준다.

**규칙**

규칙과 관련된 사항은 Github Readme.md에 자세히 기록  
<https://github.com/kkole3897/Bowling-Game-with-Java>/README.md

**2. 코드 설명**

**클래스**

- Main: 프로그램을 실행할 클래스

- Bowling: 볼링 게임을 규칙에 맞게 진행하는 클래스

- Player: 게임에 참여하는 플레이어에 관한 클래스

- GameBoard: 게임 진행하면서 표시할 점수판에 관한 클래스

- Pins: 볼링핀과 관련된 클래스.

**Main 클래스**

main 메소드에서 Blowing 클래스를 이용해서 게임을 실행하는 클래스

Field

- PLAYER\_NUM: 게임 전체에 참여하는 플레이어 수를 final static int로 선언.

Method

- public static void main(String[]): Bowling 객체를 생성해서 게임을 실행하는 메소드

**Bowling 클래스**

생성자에서 게임에 필요한 진행을 구현.

Field

- players: Player 객체를 PLAYER\_NUM만큼 배열로 선언해서 사용. (**Player[]**)  
- turn: 현재 프레임이 어떤 플레이어의 턴인지 알려주는 **int** 변수. 이 게임에서는 2명의 플레이어가 참여하기 때문에 1과 2가 반복됨.  
- frame: 현재 프레임을 알려주는 **int** 변수. 1부터 시작해서 10까지 증가하고 10이 넘어가면 모든 프레임을 모든 플레이어가 수행했기 때문에 게임을 종료하는 기준이 된다.  
- gameBoard: 게임을 진행하면서 사용할 점수판 객체. (**GameBoard**)

Constructor

- 생성자에서 전체 흐름을 구현한다.  
- Scanner을 이용해서 플레이어의 등급을 결정한다. 여기서 입력은 0: 종료, 1: 초급, 2: 중급, 3: 고급으로 한다. 키 입력은 예외처리를 해서 잘못된 입력이 들어오지 않게 한다.  
- 정확한 입력을 했을 때 players 배열의 Player을 등급에 맞게 객체를 생성한다.  
- turn을 0, frame을 1로 초기화하고 본격적으로 게임을 시작한다.  
- 게임은 frame이 10이 넘어갈 때까지 반복하면서 진행한다.  
- 매 프레임은 시도 횟수와 어떤 플레이어의 차례인지 알려주고, 아무 키나 입력됐을 때 공을 던지도록하고, 던질 때마다 남은 핀의 모습과 점수판을 표시하도록 한다.  
- 프레임은 1~9와 10이 다르게 진행되기 때문에 분기한다.  
- 1~9프레임인 경우  
 - 1차 시도에서 Player 클래스의 throwBall() 메소드를 호출해서 쓰러뜨린 핀의 수와 남은 핀  
 의 수를 계산한다.  
 - 쓰러뜨린 핀의 수는 공을 던질 때마다 GameBoard 클래스의 writeHistory() 메소드를 호출   
 해서 기록한다.  
 - 10개의 핀을 모두 넘어뜨린 경우에는 스트라이크이므로 현재 프레임이 기본 점수와 다음 2  
 번의 시도에서 보너스 포인트를 얻을 수 있음을 GameBoard의 writeFrameScore() 메소드를   
 호출해서 기록한다.  
 - 10개의 핀을 모두 넘어뜨린 경우가 아닌 경우 기본 점수를 더해주고 아직 프레임이 끝나지   
 않았음을 GameBoard의 writeFrameScore() 메소드를 호출해서 기록한다. 이후에 2차 시도를   
 한다.  
 - 1차 시도와 동일하게 진행하고, 남은 핀을 모두 처리한 경우 스페어이기 때문에 기본 점수  
 를 더해주고 다음 1번의 시도에서 보너스 점수를 받을 수 있음을 GameBoard의   
 writeFrameScore() 메소드를 호출해서 기록한다.  
 - 남은 핀을 모두 처리하지 못한 경우에는 단순히 기본 점수를 더해주고 보너스 점수를 얻을   
 수 없음을 GameBoard의 writeFrameScore()을 호출해서 기록한다.  
 - 턴을 넘긴다. 다음 플레이어도 이와 동일하게 진행한다.  
- 10프레임인 경우  
 - 10프레임은 보너스 점수를 얻지 못하고, 스트라이크나 스페어 처리를 했을 경우 3차 시도가   
 주어진다.  
 - 매 시도마다 쓰러뜨린 핀의 수는 GameBoard의 writeHistory() 메소드를 호출해서 기록하고,   
 보너스 점수는 얻을 수 없기 때문에 GameBoard의 wirteFrameScore()에 넘어뜨린 핀의 수만   
 넘겨주고 보너스 점수는 얻을 수 없음을 매개 변수로 넘겨주어 기본 점수만 누적시킨다.  
 - 1차 시도에서 스트라이크를 친 경우, 핀을 리셋하고 2차 시도를 한다. 2차 시도에서도 스트  
 라이크를 친 경우, 핀을 리셋하고 3차 시도를 한다. 2차 시도에서 스트라이크를 치지 못한   
 경우는 남은 핀을 이용해서 3차 시도를 한다.  
 - 1차 시도에서 스트라이크를 치지 못한 경우, 남은 핀으로 2차 시도를 한다. 만약 2차 시도  
 에서 스페어 처리에 성공하면, 핀을 리셋하고 3차 시도를 하고, 스페어 처리를 못한 경우에   
 는 프레임을 종료한다.  
- frame 변수가 10을 초과하면 GameBoard의 show() 메소드를 호출해서 마지막 점수판을 보여주고 printWinner() 메소드를 호출해서 승자를 알려주고, 게임을 종료한다.

**Player 클래스**

Field

BEGINNER: 초급에 해당하는 정적 변수 int.  
INTERMEDIATE: 중급에 해당하는 정적 변수 int.  
ADVANCED: 고급에 해당하는 정적 변수 int.  
level: Player 객체의 등급을 저장하는 int.

Contructor

- level을 매개 변수로 넘겨주어 level을 초기화한다.

Method

- public String getLevel(): level에 따라 BEGINNER이면 초급, INTERMEIDATE이면 중급, ADVANECE이면 고급을 return.  
- public throwball(int): 남은 핀의 개수를 매개 변수로 넘겨주어 level에 따라 쓰러뜨리는 핀의 개수를 다르게 처리해서 넘어뜨린 핀의 개수를 return한다. 넘어뜨린 핀의 수는 0이상 남은 핀의 수 이하로 한다.  
 - 초급: 평균 100의 점수를 얻는 분포를 갖도록 하기 위해선 한 프레임당 10을 얻어야 한다.   
 따라서 매 시도당 평균 5개의 핀을 쓰러뜨릴 수 있도록 설정해주면 될 것이라고 판단했다.   
 하지만 이렇게 되면 매 프레임당 스페어 처리로 인한 보너스 점수로 인해 100을 훌쩍 넘는   
 점수를 얻게 된다. 따라서 평균 5에 분산을 크게 한 정규분포를 이용헸다.  
 - 중급: 평균 150의 점수를 얻기 위해선 대부분의 프레임에서 스페어 처리가 가능하게 해야  
 했다. 따라서 평균 7에 초급보다는 조금 작은 분산의 정규분포를 이용했다.  
 - 고급: 평균 200의 점수를 얻기 위해선 스페어와 스트라이크가 번갈아가며 발생해야 한다고   
 판단했다. 평균 9에 분산은 중급보다 더 작게 한 정규분포를 이용했다.

**GameBoard 클래스**

Field

- history: 플레이어가 쓰러뜨린 핀의 개수를 저장할 2차원 int 배열. 전체 시도 횟수는 10프레임의 3차 시도까지 21번이므로 플레이어 수 X 21이 된다.  
- historyWritten: 점수판을 출력한 때 아직 던지지 않은 시도의 history를 공백으로 출력하도록 history를 기록했는지 확인하는 2차원 boolean 배열. 크기는 history와 동일.  
- score: 프레임에 얻은 점수. 기본 점수와 보너스 점수를 모두 더해서 저장한다. 크기는 플레이어수 X 10이 된다.  
- bonus: 프레임마다 보너스 점수를 얼마나 더 더해줘야 하는지 알려주는 2차원 int 배열. 크기는 score과 동일하고, -1, 0, 1, 2만 저장한다. -1은 아직 진행하지 않은 프레임, 0은 보너스 점수를 얻지 못했거나 모두 계산한 프레임, 1과 2는 해당 프레임이 아직 보너스 점수를 계산해야 한다는 것을 뜻한다.

Constructor

- 모든 Field 변수들을 초기화한다. historyWritten은 모두 false로, score은 0, bonus는 -1로 초기화한다.

Method

- public void writeHistory(int, int, int, int): 넘어뜨린 핀의 수, 플레이어id, 현재 프레임 위치, 시도를 매개 변수로 넘겨주어 넘어뜨린 핀의 수를 history에 저장하고 해당 위치의 writtenHistory를 true로 변경한다. 인덱스 계산은 한 프레임당 2번의 시도이고 10프레임만 3회이므로 (2 \* (프레임 위치 – 1) + 시도 – 1)로 한다.  
- public void writeFrameScore(int, int, int, int): 넘어뜨린 핀의 수, 계산해야 할 보너스 점수 개수, 플레이어id, 프레임 위치를 매개 변수로 넘겨주어 해당 프레임의 점수와 보너스 점수를 계산하고 저장하는 메소드이다. 우선 현재 프레임의 기본 점수를 넘어뜨린 핀의 수만큼 score 배열의 알맞은 위치에 더해주고 보너스 점수를 받을 수 있는 개수를 bonus의 알맞은 위치에 설정한다. 보너스 점수는 최대로 이전 2번째 프레임까지만 확인하면 되기 때문에 반복문으로 2번 이전의 프레임까지 반복하면서 해당 위치의 bonus가 1, 2이면 현재 넘어뜨린 핀의 개수만큼 해당 프레임의 점수를 증가시키고 bonus는 1만큼 줄여준다.  
- private int sumScoreTo(int, int): 플레이어의 어떤 프레임까지의 누적점수를 계산해서 반환하는 메소드.  
- public void printWinner(): 게임이 종료될 때 호출되는 승자를 출력하는 메소드이다. sumScoreTo() 메소드로 플레이어의 마지막 프레임까지 누적 점수를 비교해서 승자를 출력하고, 점수가 같으면 무승부임을 알려준다.  
- public void show(): 게임판의 기록을 출력하는 매소드이다. 아직 기록되지 않은 부분은 공백으로 출력되도록 한다. 각 프레임 별로 1차, 2차, 10프레임인 경우 3차까지 쓰러뜨린 핀의 개수를 출력하고 그 다음 줄에는 보너스 점수까지 모두 계산한 경우 해당 프레임까지 얻은 점수의 누적을 기록한다. 쓰러뜨린 핀의 개수는 스트라이크인 경우 X, 스페어인 경우 /로 출력한다.

**Pins 클래스**

Field

- pinState: 핀이 서있는지, 넘어져 있는지 상태를 저장하는 boolean 배열, 크기는 10이다.

Constructor

- pinState를 true로 초기화한다.

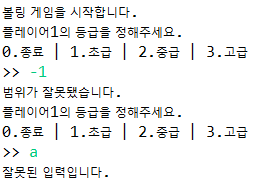
Method

- public int getRemainNum(): 서 있는 핀의 개수를 반환한다.  
- public void drop(int): 넘어뜨릴 핀의 수를 매개변수로 넘겨주어 뒤에서부터 차례대로 pinState를 false로 바꾼다.  
- public void show(): 역삼각형 형태로 서 있는 경우 O, 넘어진 경우 X로 출력한다. 첫째줄에 4개, 둘째줄에 3개, 셋째줄에 2개, 마지막 줄에 1개가 출력된다.

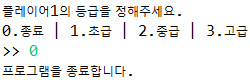
**3. 실행 결과**

**등급 설정**

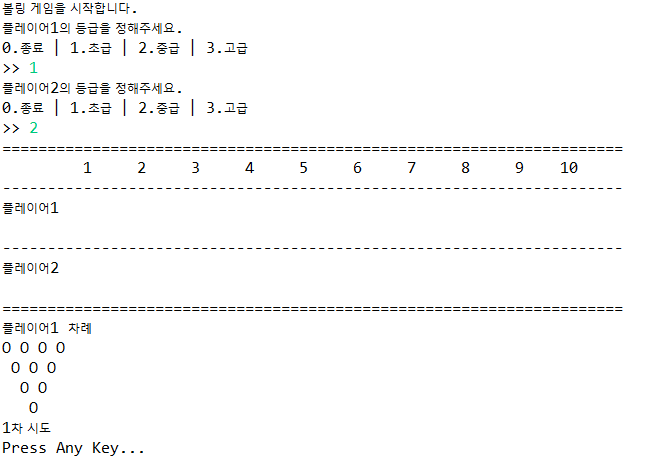
- 예외 처리



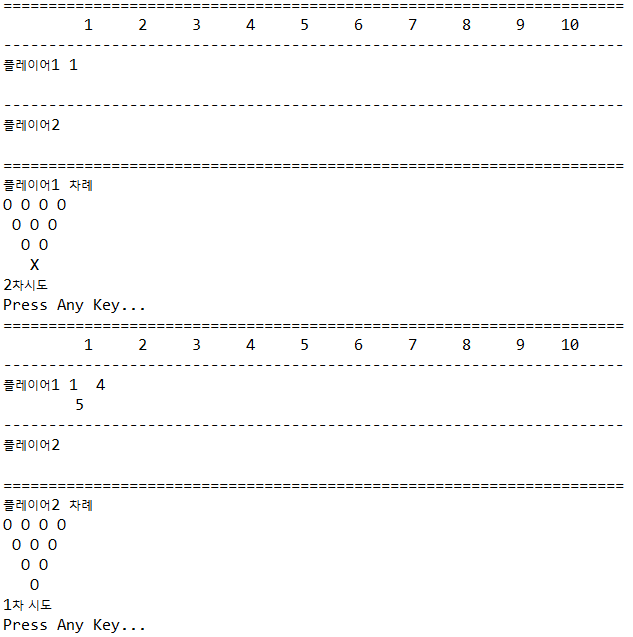
- 종료

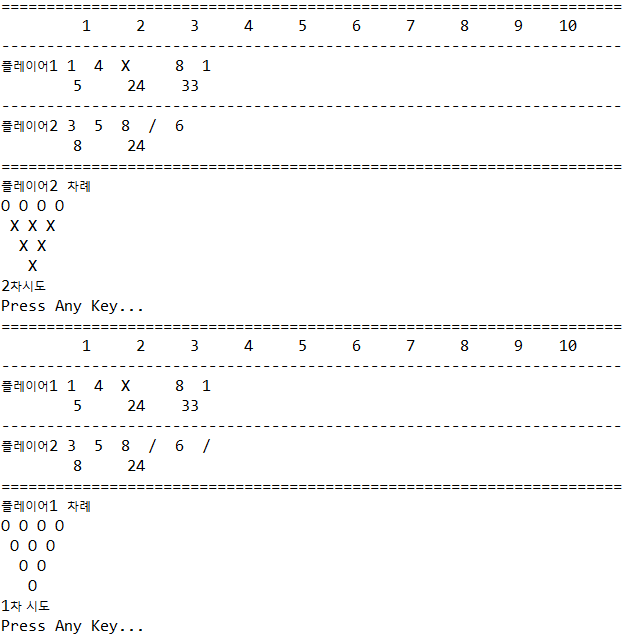


- 등급 설정

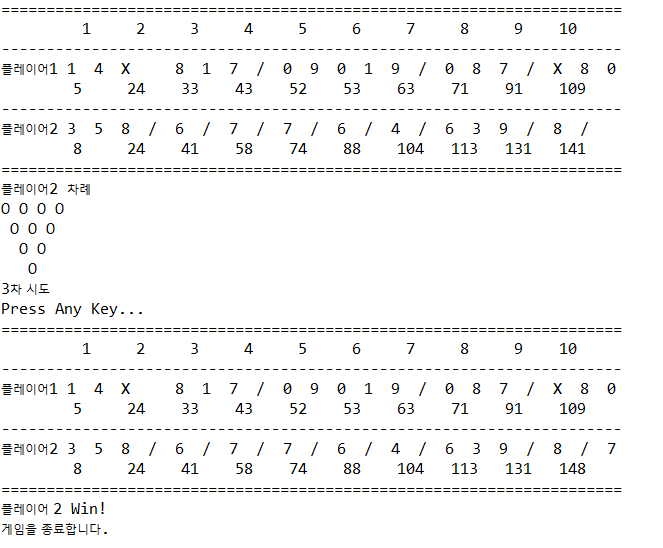


**게임 진행**





**게임 완료**

****