**Project – 볼링게임**



|  |  |
| --- | --- |
| **과 목 :** | **소프트웨어프로젝트** |
| **제출 일자 :** | **2019. 6. 9** |
| **담당 교수 :** | **이남규** |
| **학 과 :** | **사회복지학부** |
| **학 번 :** | **20170299** |
| **이 름 :** | **김유빈** |

* + 초보자(100), 중급(150), 상급(200)
  + Random number와 확률을 이용하여 스페어 처리나 스트라이크 확률을 계산하여 평균에 근접하도록 나오게 한다.

초보자 평균(100)

* 100점에 근접하게 하는 스페어 처리와 스트라이크 각각의 개수가 몇인지 알아야한다. 랜덤으로 게임을 100번정도 진행시켜서 스페어 처리가 몇 개일 때, 스트라이크가 몇 개일 때 점수가 100점에 가까워지는 지에 대한 데이터가 필요하다. 이를 위한 데이터 수집 스포르램도 따로 제작한다.

중급자 평균(150), 상급자 평균(200) 도 마찬가지다.

**<게임 룰>**

1. 사용자가 초보, 중급, 상급이 되는 것은 랜덤이다.

2. 컴퓨터 즉, 게임 난이도 초보, 중급, 상급 중 선택할 수 있도록 한다.

3. 사용자가 먼저 플레이 하는 것으로 한다.

4. 볼링 규칙 – 사용자의 스페어 처리나 스트라이크의 개수는 확률적으로 이미 정해져 있다.

① 한 프레임에 2회 진행한다.

② 스페어 처리: 총점은 다음 프레임을 종료할 때 나온다.

다음 프레임의 1회 점수가 보너스 점수

스트라이크는 프레임의 첫 회에 한방에 해야함. 첫 회가 거터(-)인데 두번째 회에 스트라이크여도 스페어 처리된다.

③ 하나 스트라이크: 스페어 처리와 마찬가지로 다음 프레임을 종료할 때 총점이 나온다.

다음 프레임의 총점이 보너스 점수

④ 두개 스트라이크(더블): 다다음 프레임(스트라이크x)가 끝나고 나서 총점이 나온다.

다음 프레임의 총점(10점) + 그 다음 프레임 1회 점수 = 보너스 점수

⑤ 세 개 스트라이크(터키): 터키가 확정이면 첫 스트라이크에 총점이 바로 나온다.

왜냐하면 보너스 점수 = 스트라이크2(10점) + 스트라이크3(10)점 이므로 이미 확정된 점수이기 때문이다.

⑥ 마지막에 스페어 처리나 스트라이크를 했다면 1회 더 주어진다.

5. 게임이 끝나면 사용자와 컴퓨터의 총점을 비교하여 승자를 가린다.

6. 게임을 계속할 건지 말건지 물어본다.

**<차별화된 볼링 게임 프로그램>**

총 클래스는 4개이다

**1)** 1개는 데이터 수집을 위한 BowlingData 클래스

본 프로그램은 초보, 중급, 상습의 평균 점수에 가까울 스트라이크, 스페어 처리의 확률을 보다 정확하게 구하기 위해 방대한 데이터를 수집하려는 목적으로 따로 데이터 수집용 프로그램을 만들었다. 이 BowlingData 클래스는 1000번, 10000번 그 이상의 횟수로 게임을 실행하여 데이터 수집을 돕는다.

나머지 3개는 게임용 클래스인데

**2)** BowlingGame 클래스: 실제로 게임하는 곳

이때, BowlingGame 클래스에서 사용자는 자신의 레벨(초,중,상)을 몰라서 컴퓨터를 매번 이길 수 없어서 게임적이다. 그리고 사용자는 게임의 난이도를 마음대로 고를 수 있다. -> 컴퓨터의 레벨(초,중,상)을 구하는 게 여기에 해당한다.

사용자는 “pin”이라고 입력하여 직접 볼링을 못 치더라도 자신이 치는 느낌을 받을 수 있다.

“pin”말고도 “end”, “restart”를 입력하여 프로그램 종료나 아예 게임을 처음부터 다시 시작할 수 있다.

**3)** PlayBowling 클래스: 게임하는 데 필요한 기본적인 도구

**4)** ComputerGame 클래스: 사용자의 경쟁상대인 컴퓨터가 게임하는 곳

**< PlayBowling >**

두 가지의 함수가 있다!

1. 한 게임의 핀 횟수(총21)를 저장하는 배열

**public** **void** rollarray(**int** pins) {

rolls[currnetroll++] = pins;

}

2. 한 프레임의 총점을 구하는 함수

**public** **int** getScore(**int** past\_score, **int** currentframe) {

**int** getscore = past\_score;

**int** i = currentframe \* 2; //현재 핀의 횟수

**if**(rolls[i] == 10 && rolls[i+2] != 10) {//원 스트라이크 getscore += (10 + rolls[i+2] + rolls[i+3]);

}

**else** **if**(rolls[i] == 10 && rolls[i+2] == 10 && rolls[i+4] != 10){ //투 스트라이크

getscore += (10 + 10 + rolls[i+4]);

}

**else** **if**(rolls[i] == 10 && rolls[i+2] == 10 && rolls[i+4] == 10) { //트리 스트라이크

getscore += 10 \* 3;

}

**else** **if**(rolls[i] + rolls[i+1] == 10) { //스트라이크가 아닌 스페어 처리

getscore += (10 + rolls[i+2]);

}

**else** { //일반적인 경우

getscore += (rolls[i] + rolls[i+1]);

}

**return** getscore;

}

**< BowlingData>**

이 클래스의 목적: 초보, 중급, 상급의 스트라이크, 스페어 처리의 평균적 개수를 구하는 것이다.

구하고 나서 파일로 출력하여 보관하고자 한다.

80 <= 초보자 점수 <= 120

130 <= 중급자 점수 <= 170

180 <= 상급자 점수 <= 220

이 구간 내 스트라이크, 스페어 처리 개수를 구한다.

**int** MAX\_NUM = 1000; //게임 최대 몇번 돌리지 -> 1000번을 돌려서 최대한 정확한 값이 나오도록

ArrayList<Integer> spare\_count = **new** ArrayList<Integer>(MAX\_NUM); //한 게임당 스페어 처리 개수

ArrayList<Integer> strike\_count = **new** ArrayList<Integer>(MAX\_NUM); //한 게임당 스트라이크 개수

ArrayList<Integer> totalScore = **new** ArrayList<Integer>(MAX\_NUM); //한 게임당 총점 배열

컬랙션을 통해 스페어 처리, 스트라이크 개수, 총점을 쉽게 저장한다.

여기서의 ‘게임 원리’는 **BowlingGame 클래스**와 흡사하다.

게임 원리는 ‘BowlingGame’ 클래스 설명할 때 설명하려고 한다.

한 게임이 끝날 때마다

spare\_count.add(spare);

strike\_count.add(strike);

totalScore.add(score[9]);

컬렉션에 정보를 저장한다.

게임을 MAX\_NUM만큼 돌리면 초보자, 중급자의 빈도 수, 스페어 처리 개수, 스트라이크 개수를 변수에 저장하여 이를 파일에 출력한다.

**for**(i = 0; i<MAX\_NUM;i++) {

//1. 초급(100)인 경우의 스트라이크, 스페어 개수

**if**(totalScore.get(i) >= 80 && totalScore.get(i) <= 120) {

easy\_spare += spare\_count.get(i);

easy\_strike += strike\_count.get(i);

easy\_count++; }

//2. 중급(150)인 경우

**else** **if**(totalScore.get(i) >= 130 && totalScore.get(i) <= 170) {

middle\_spare += spare\_count.get(i);

middle\_strike += strike\_count.get(i);

middle\_count++;

}

}

\*\* 상급자의 점수는 빈도수가 거의 없으므로 상급자의 정보는 중급자의 정보에서 유추하기로 했다. \*\*\*\*\*\*

이를 파일에 출력한다. 초보자 경우만 살펴보면

**try** {

FileWriter fout = **new** FileWriter("C:\\Users\\mk\\eclipse-workspace\\Project\\bowlingdata.txt");

fout.write("초보자");

fout.write("\r\nspare: " + Double.*toString*(easy\_spare) + " strike: " + Double.*toString*(easy\_strike) + " count: " + Double.*toString*(easy\_count));

fout.write("\r\n스페어처리 확률 = " + Double.*toString*((**double**)easy\_spare/easy\_count/10 \* 100) + "%");

fout.write("\r\n스트라이크 확률 = " + Double.*toString*((**double**)easy\_strike/easy\_count/10 \* 100)+ "%");

fout.close();

}

**catch**(IOException e){

System.***out***.println("파일 입출력 오류");

}

초보자의 스페어 처리 확률 = (스페어 처리 개수) / (초보자 빈도 수)

초보자의 스트라이크 확률 = (스트라이크 개수) / (초보자의 빈도 수)

스크린샷이(가) 표시된 사진

자동 생성된 설명

* 파일의 결과 화면

이를 통해 초보자, 중급자, 상급자의 스트라이크, 스페어 처리 확률 구하기

* + **초보자 스페어 처리 확률 = 21.05%**
  + **초보자 스트라이크 확률 = 10.00%**
  + **중급자 스페어 처리 확률 = 45.00%**
  + **중급자 스트라이크 확률 = 40.31%**
  + **상급자 스페어 처리 확률 = 40.00%**
  + **상급자 스트라이크 확률 = 57.31%**

이때 중급자의 확률이 메모지와 다른 것을 확인 할 수 있다.

중급자의 평균 점수(150)에 가까워 지려고 여러 번의 시도 끝에 내린 결과이다.

상급자의 확률은 중급자에서 유추했다.

초보자인 경우(스트라이크 < 스페어 < 나머지)

중급자인 경우(나머지 < 스트라이크 < 스페어)

상급자인 경우(나머지 < 스페어 < 스트라이크)

이 규칙에 맞춰서 스페어 처리, 스트라이크 확률을 계산했다.

**<BowlingGame 클래스> - 게임 작동 방식**

**1. 사용자 아이디 입력 : 자신이 원하는 이름을 붙이면 된다.**

name = s.nextLine();

스크린샷이(가) 표시된 사진

자동 생성된 설명

**2. 게임을 본격적으로 시작한다.**

ComputerGame com = **new** ComputerGame();

**int** [] user\_score = **new** **int** [10]; //한 게임의 총점 10개

**byte** spare = 0, strike = 0; //스트라이크, 스페어 유무

이 변수들은 한 게임이 시작할 때마다 리셋되도록 한다.

**3. 사용자의 레벨을 랜덤하게 정하기**

user\_level = (**int**)random.nextInt(10);

random number와 확률을 통해 레벨을 정하기

상급자 = 20%

초보자 = 40%

중급자 = 40%

레벨이 정해지면(user\_level) 각 레벨의 스트라이크, 스페어 처리 확률을 가지고 오기

**ex)상급자의 경우**

**if**(user\_level < 2) {

user\_spare = 4000;

user\_strike = 5731;

}

**4. 컴퓨터의 레벨을 선택하기 (게임 난이도)**

com.computerlevel(Integer.*parseInt*(s.nextLine()));

이때 com => ComputerGame com = **new** ComputerGame();

ComputerGame 클레스의 함수인 computerlevel()로 컴퓨터의 스트라이크, 스페어 처리 확률을 정한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

**5. 볼링 게임 ( 한 게임 = 총 10 개의 frame)**

**frame = 0번지부터 시작 ~ 9번지**

**1) 핀 던지기**

**int** pin1 = 0, pin2 = 0, pin3 = -1; //핀(1회), 핀(2회), 핀(3회-마지막프레임)

order변수에 사용자가 “pin” 입력하면 핀을 던지는 것

“end” : 프로그램이 완전히 종료

“restart” : 게임을 다시 시작( 이름부터 다시 정하기)

**2) 사용자의 레벨에 따라 핀의 개수 정하기 -> 스트라이크, 스페어 처리 확률 적용**

초보자인 경우(스트라이크 < 스페어 < 나머지)

중급자인 경우(나머지 < 스트라이크 < 스페어)

상급자인 경우(나머지 < 스페어 < 스트라이크)

이 기준으로 가장 확률이 적은 것부터 적는다 -> 그래야 random number 와 확률 적용됨

초보자인 경우(user\_spare == 2105)

**if**(user\_spare == 2105) {

**if**(chance < user\_strike) { //strike인 경우

pin1 = 10;

pin2 = 0;

System.***out***.println("X"); //화면에 이번에 던진 핀을 보여주기

}

**else** **if**(chance < user\_spare) { //spare인 경우

pin1 = random.nextInt(10); //스트라이크 제외 pin2 = 10 - pin1;

**if**(pin1 == 0) System.***out***.println("-,/");

**else** System.***out***.println(pin1 + "," + "/");}

**else**{

pin1 = random.nextInt(10); //스트라이크 제외

pin2 = random.nextInt(10);

**while**(pin1 + pin2 >= 10) { //스페어 제외

pin2 = random.nextInt(11);

}

**if**(pin1 == 0 && pin2 != 0) System.***out***.println("-," + pin2);

**else** **if**(pin1 != 0 && pin2 == 0) System.***out***.println(pin1 + ",-");

**else** **if**(pin1 == 0 && pin2 == 0) System.***out***.println("-,-");

**else** System.***out***.println(pin1 + "," + pin2);

}

}

이번에 친 핀을 보여주는데

스페어인 경우: “/”

거터인 경우: “-“

스트라이크인 경우 : “X”

경우를 나누어 출력한다.

단, 중급자와 상급자는 스트라이크 확률이 높아서 스트라이크가 연속으로 4번이상 나올 수 있다. 혼란을 방지하기 위해 중급자와 상급자의 STRIKE는

**if**(strike >= 3) { //스트라이크를 스페어 처리함

pin1 = random.nextInt(10); //스트라이크 제외

pin2 = 10 - pin1;

**if**(pin1 == 0) System.***out***.println("-,/");

**else** System.***out***.println(pin1 + "," + "/");

}

**else**{

pin1 = 10;

pin2 = 0;

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

}

각 핀 두개를 PlayBowling 클래스의 함수인 rollarray에 저장

user.rollarray(pin1);

user.rollarray(pin2);

**3) 이전에 스트라이크 였는지 조사하여 총점 계산**

**- 단, 현재가 스트라이크인 경우는 제외**

이전에 총점 보류해둔 스트라이크가 있는지 확인하고 있다면 총점을 계산한다.

그전에 총점 계산하는 소스코드를 살펴보면

**try** {

user\_score[frame] = user.getScore(user\_score[frame - 1], frame);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame] = user.getScore(0, frame);

}

한 게임의 총점을 저장하는 배열 user\_score에 PlayBowling 클래스의 함수인 getScore()로 총점을 구한다. 단, 배열은 예외처리를 해줘야 안전하다.

getScore(이전 총점, 현재 프레임) 함수이다.

본격 소스코드를 살펴보면

**if**(pin1 != 10 && frame >= 1 && user.rolls[(frame - 1) \* 2] == 10) { //스트라이크

**if**(frame >= 2 && user.rolls[(frame - 2) \* 2] == 10) { //더블

**if**(frame >= 3 && user.rolls[(frame - 3) \* 2] == 10) { //터키

strike--;

**try** {

user\_score[frame - 3] = user.getScore(user\_score[(frame - 3) - 1], frame - 3);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame - 3] = user.getScore(0, frame - 3);

}

}

strike--;

**try** {

user\_score[frame - 2] = user.getScore(user\_score[(frame - 2) - 1], frame - 2);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame - 2] = user.getScore(0, frame - 2);

}

}

strike--;

**try** {

user\_score[frame - 1] = user.getScore(user\_score[(frame - 1) - 1], frame - 1);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame - 1] = user.getScore(0, frame - 1);

}

}

터키 -> 더블 -> 스트라이크 = 터키

더블 -> 스트라이크 = 더블

스트라이크

이런 순으로 작동한다.

strike--; -> 처리 되지 않은 스트라이크의 개수 = strike

총점이 처리가 되었으므로 마이너스

**4) 이전에 스페어 처리가 있었는지 조사하여 총점 보류된 것을 총점 계산하기**

**if**(frame >= 1 && user.rolls[(frame-1) \* 2] != 10 && user.rolls[(frame-1) \* 2] + user.rolls[(frame-1) \* 2 + 1] == 10) {

spare--;

**try** {

user\_score[frame - 1] = user.getScore(user\_score[(frame - 1) - 1], frame - 1);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame - 1] = user.getScore(0, frame - 1);

}

}

spare--; -> 스페어 처리 안된 것 개수 = spare

**5) 이전 것들을 다 처리했다면 이제는 현재 총점 계산**

: 스트라이크/ 스페어 처리 -> 총점 보류

나머지 -> 총점 계산

총점은 매 프레임마다 보여주기

**if**(pin1 == 10) { //이번에 스트라이크인 경우 - 총점 보류

strike++;

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

}

**else** **if**(pin1 + pin2 == 10) { //이번에 스페어 처리인 경우 - 총점 보류

spare++;

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

}

**else** { //현재 아무것도 아닌 경우 - 총점 계산

**try** {

user\_score[frame] = user.getScore(user\_score[frame - 1], frame);

}

**catch**(ArrayIndexOutOfBoundsException e) {

user\_score[frame] = user.getScore(0, frame);

}

}

**6) 마지막 프레임에서 스트라이크, 스페어 처리인 경우 ★**

pin3를 던지기 (한번 더 던지기)

pin3 = random.nextInt(11); //21번째 핀

user.rollarray(pin3);

System.***out***.print("/" + pin3);

* + 마지막 프레임이 스트라이크인 경우

터키, 더블은 그대로이지만

스트라이크의 경우: 총점 = 10(본인) + pin3; -> 예외이다.

user\_score[9] = user\_score[8] + 10 + pin3;

* + 마지막 프레임이 스페어 처리인 경우

user\_score[9] = user\_score[8] + 10 + pin3;

**7) 한 프레임이 끝날 때 마다**

**여태까지 던진 핀 수와 총점들을 나열하여 화면에 출력하기**

단, 스페어인 경우: “/”

거터인 경우: “-“

스트라이크인 경우 : “X”

21번째 핀이 있는지 없는지에 따라 출력도 달라진다.

**if**(pin3 != -1) //21번쨰 핀수

System.***out***.print("," + user.rolls[20]);

\*\*\*총점 출력의 주의점\*\*\*

총점 보류된 지점 전까지만 출력되어야 한다.

즉, 스페어 처리, 스트라이크 전까지 총점 출력하기

**for**(j = 0;j <= frame - (spare + strike);j++)

System.***out***.printf("%3d ",user\_score[j]);

spare, strike : 총점 보류가 몇 개 있는지 확인 가능!

단, 21번째 핀 == 마지막에 총점 보류에 해당하는 경우 따로 처리해야함

**if**(pin3 != -1) {//21번쨰 핀수 - 총점 마저 보이기

**for**(**int** k = j; k <= frame ;k++)

System.***out***.printf("%3d ",user\_score[k]); }

**8) 컴퓨터 플레이 하기**

ComputerGame 클래스를 이용하기

com.computerplay(frame);

ComputerGame 클래스의 함수인 computerplay( )는 현재 BowlingGame 클래스 소스코드와 대단히 유사하다. BowlingGame 클래스의 게임원리와 거의 그대로 복사한 함수이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

* 게임 첫 진행 화면

**<게임 화면>**

스크린샷이(가) 표시된 사진

자동 생성된 설명

사용자 이름(id)을 입력한다.

개체이(가) 표시된 사진

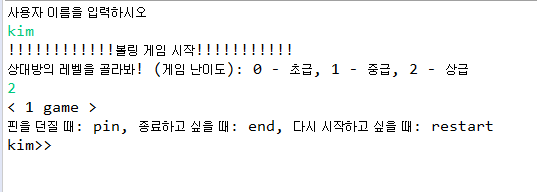
자동 생성된 설명

상대방 즉, 컴퓨터의 레벨 고르기

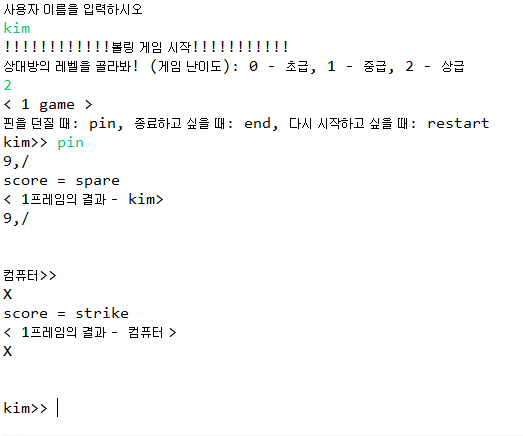
초급 = 0

중급 = 1

상급 = 2



명령어를 입력한다. -> pin, end, restart

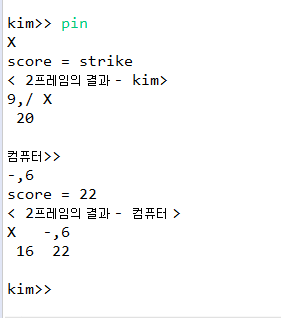


한 프레임이 진행되었다.

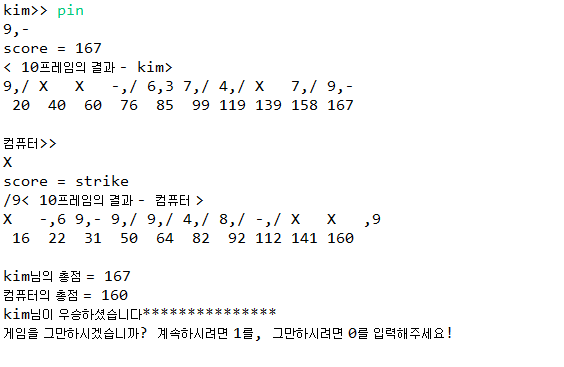
9,/ : 사용자가 던진 핀 수

score = spare : 이번 프레임의 총점

<1프레임의 결과 – kim> : 여태까지의 핀 수와 총점 나열

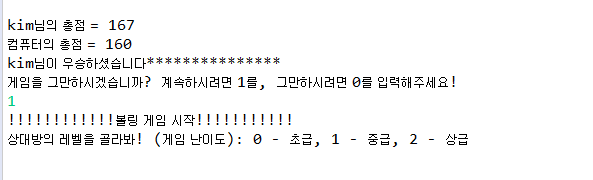


또 핀을 던진 화면

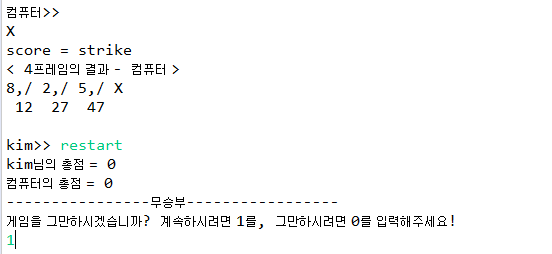


계속 핀을 던지고 한 게임이 종료된 화면

게임을 계속할 건지 물어본다



게임을 계속 진행하는 모습이다.



중간에 다시 시작하려고 명령어 restart 를 친 화면