REPORT

과제 08

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 과목명 : | | | 객체지향프로그래밍 | | | | | | | | |
| 담당교수 : | | | 엄진영 | | | | | | | 교수님 | |
| 제출일 : | 2021 | | | 년 | | 05 | 월 | | 12 | | 일 |
| 공과 | | 대학 | | | 컴퓨터공학 | | | 과 | | | |
| 학번 : | | 2016112154 | | | 이름: | | | 정동구 | | | |

문제 1.

텍스트이(가) 표시된 사진

자동 생성된 설명

각 동물 클래스가 Animal 클래스를 상속하고

public class Dog extends Animal{

Dog()

{

this.sound = "Bow wow";

}

}

와 같이 sound 값을 변경하여 bark를 했을 시 결과가 다르게 나오도록 하였다.

문제 2.

-결과

텍스트이(가) 표시된 사진

자동 생성된 설명

게임 시작 시 각각 플레이어에게 카드를 5장 배부하고 한장을 뒤집어 놓는다.

직접 플레이 하는 유저의 경우 1을 눌러서 카드를 뽑거나 2를 눌러서 낼 카드를 정할 수 있다. 손에 들고 있는 카드는 3을 통해서 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

2를 입력하여 카드를 낼 시 낼 카드의 모양과 숫자를 차례대로 입력한다. 이 때 손에 없는 카드일 경우 낼 카드를 다시 입력하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 검은색이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 검은색이(가) 표시된 사진

자동 생성된 설명

오타로 인해 없는 카드 취급되어 다시 입력을 요구하는 화면

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

카드를 다 낸 플레이어가 나오면 즉시 게임이 종료되며 그 다음으로 카드가 가장 적은 사람이 2등이 된다. 게임 종료 이후 재시작 여부를 결정 할 수 있다.

(추가)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

기존 코드에 컴퓨터가 낸 카드를 계속 쥐고 있다 다시 내는 문제를 해결하여 추가합니다.

-코드

public class User {  
  
 public ArrayList<Card> hand = new ArrayList<Card>();//손에 쥔 카드  
 Scanner sc = new Scanner(System.*in*);  
 public boolean compare(Card a, Card b)  
 {  
 boolean same = false;  
 if(a.getShape()==b.getShape() || a.getNum()==b.getNum())  
 same = true;  
 else  
 same = false;  
  
 return same;  
 }  
  
 public void drawCard(Card draw)  
 {  
 this.hand.add(draw);  
 }//카드 뽑기  
  
 public boolean searchCard(Card table)//테이블에 카드가 패에 존재하는지 확인  
 {  
 for(int i =0; i< hand.size();i++)  
 {  
 if(compare(table,hand.get(i)))  
 return true;  
 }  
 return false;  
 }  
  
  
 public Card playCard(Card table)//테이블에 존재하는 카드와 일치할 경우 낸다.  
 {  
 String shape;  
 int num;  
 // System.out.println(table.getShape() + " " + table.getNum());  
 System.*out*.println("choose card to play");  
 System.*out*.print("shape:");  
 shape = sc.nextLine();  
 System.*out*.print("num:");  
 num = sc.nextInt();  
 sc.nextLine();  
 while(table.num != num && !table.shape.equals(shape))//일치하지 않는 경우  
 {  
 System.*out*.println("cannot play card");  
 System.*out*.println("choose card to play");  
 System.*out*.print("shape:");  
 shape = sc.nextLine();  
 System.*out*.print("num:");  
 num = sc.nextInt();  
 sc.nextLine();  
 }  
 int index=0;  
 for(int i = 0 ; i < hand.size(); i ++)  
 {  
 if(shape.equals(hand.get(i).getShape()) && num == hand.get(i).getNum())  
 index = i;  
 }  
 Card play = hand.get(index);  
 hand.remove(index);//낸 후 손에서 제거  
 return play;  
 }

User Class.

Player, computer1 computer2 class의 부모클래스 이며 게임에 참여하는 인원 별 손에 쥔 카드 패와 관련된 기능이 구현되어있다. 카드를 뽑거나 내는 행위이다.

public Card playCard(Card table)//메소드 오버라이딩  
{  
 ArrayList<Card> same\_card = new ArrayList<Card>();  
 same\_card.clear();  
 for(int i = 0;i<hand.size();i++)  
 {  
 if(compare(hand.get(i),table)==true)  
 same\_card.add(hand.get(i));  
 }  
 Collections.*sort*(same\_card, new Comparator<Card>() {//숫자 기준 내림차순 정렬. 일치하는 카드 중 숫자가 가장 큰 카드 우선 제출  
 @Override  
 public int compare(Card o1, Card o2) {  
 if(o1.getNum() > o2.getNum())  
 return -1;  
 else if(o1.getNum() < o2.getNum())  
 return 1;  
 return 0;  
 }  
 });  
 for(int i = 0;i<hand.size();i++)  
 {  
 if(hand.get(i).getShape().equals(same\_card.get(0).getShape()) && hand.get(i).getNum() == same\_card.get(0).getNum())  
 hand.remove(i);//낸 카드 패에서 삭제  
 }  
 return same\_card.get(0);  
}

computer1 class 와 computer2 class에서 playCard() 메소드의 오버라이딩이다. 카드를 내는 행위를 자동으로 한다. 우선 모양 또는 숫자가 일치하는 손에 쥔 카드를 전부 arrayList에 추가하고 그중 숫자가 가장 큰 것을 우선으로 오게 정렬하여 가장 큰 숫자 패를 낸다..

CardGame()  
{  
 players.add(new Player());  
 players.add(new Computer1());  
 players.add(new Computer2());  
}  
public void createCardset()//카드셋 생성  
{  
 deck.clear();  
 for(int i = 0; i < 4; i++)  
 for(int j = 0; j < 9; j++)  
 {  
 deck.add(new Card(cardName[i],cardNum[j]));//모양별 숫자 2-10까지 add  
 }  
 Collections.*shuffle*(deck);//무작위로 섞는다  
}  
  
public void initializeCard()  
{  
 createCardset();  
 for(int i = 0 ; i < 5;i++){//5장씩 배부  
 for(int j =0 ;j<players.size();j++)  
 {  
 players.get(j).drawCard(deck.get(0));  
 deck.remove(0);  
 }  
 }  
 table\_card = deck.get(0);//카드 한장 뒤집기  
 deck.remove(0);  
}  
  
public Card dealCard()//유저가 카드를 뽑아 갈 때  
{  
 Card temp = deck.get(0);  
 deck.remove(0);  
 return temp;  
}

CardGame Class로 게임 진행에 주요한 역할을 하는 클래스이다. 처음에 카드를 섞고 나눠주고, 뒤집어 놓을 카드를 정하고, 유저가 뽑을 카드를 나눠준다.

for(int i = 1;i<3;i++)  
{  
 System.*out*.println(game.players.get(i).getClass().getName()+" is playing");  
 if(game.players.get(i).searchCard(game.getTable\_card())==true) {  
 game.setTable\_card(game.players.get(i).playCard(game.getTable\_card()));  
 System.*out*.println(game.players.get(i).getClass().getName()+"has played ["+game.getTable\_card().getShape()+" "+game.getTable\_card().getNum()+"]");  
 }  
 else  
 game.players.get(i).drawCard(game.dealCard());  
}

Main Class에서 computer가 게임을 진행하는 부분이다. 순서대로 진행하며 오버라이딩한 playCard메소드를 통해 카드를 뽑을지 낼지 판단하여 진행한다.

if(game\_end == true) {  
 //game.ranking();  
 System.*out*.println("Winner is "+game.players.get(win).getClass().getName());  
 game.players.remove(win);  
 if(game.players.get(0).hand.size()<game.players.get(1).hand.size())  
 System.*out*.println("Second Rank is "+game.players.get(0).getClass().getName());  
 else if(game.players.get(0).hand.size()>game.players.get(1).hand.size())  
 System.*out*.println("Second Rank is "+game.players.get(1).getClass().getName());  
 else if(game.players.get(0).hand.size()==game.players.get(1).hand.size())  
 System.*out*.println("Second Rank is "+game.players.get(0).getClass().getName()+" and "+game.players.get(1).getClass().getName());  
 String restart;  
  
 System.*out*.print("restart game? (y/n) :");  
 restart = *sc*.nextLine();  
 switch(restart)  
 {  
 case "y":  
 game.deck.clear();  
 game.initializeCard();  
 System.*out*.println("Start new Game");  
 break;  
 case "n":  
 System.*out*.println("exit game");  
 System.*exit*(0);  
 }  
  
 }  
 else  
 {  
 System.*out*.println(game.players.get(1).getClass().getName()+"has "+game.players.get(1).hand.size()+"cards in hand");  
 System.*out*.println(game.players.get(2).getClass().getName()+"has "+game.players.get(2).hand.size()+"cards in hand");  
 }  
 }  
}

게임에 참여한 인원 중 승리자가 있을 경우 승리자 이름을 출력하고, 이등 이름도 출력한다. 이후 키보드 입력을 통해 게임을 새로 진행할지 프로그램을 종료할지 정한다.

승리자가 없을 경우에는 각 컴퓨터의 남은 카드수를 출력한다.