**2015-2016**

SRINJOY SANTRA

CLASS XII

2015-2016



Annual Computer Project:

Game of UNO

**Design:**

This is an attempt to emulate the classic card game of UNO.In order to implement the game through programming in bluej, the fundamental concepts of Java language have been used.

The program has been divided into three classes;

1) Playcards2: (base class) manages cards beginning from card allocation, cards matching, display of discarded card and cards held by each player, deciding nature of power cards, keeping score of winner to sort cards once one card is discarded.

2) Game\_playUlt2: (derived class) starts the game, considers number of players (human and computer), effecting power cards, informing the required player the action of power card, selecting the appropriate card, drawing a card, passing a turn, deciding action of computer players.

3) File\_Display: displays wallpaper, rules of Uno, feedback details.

\*\* An added function of cheat code has also been included.When options of seeing and exiting is screened just enter “XLR8”. Enter cards in Block letters. e.g.WILD\_DRAW\_FOUR ; REVERSE BLUE etc. (This was made to check the proper functioning of the program at the micro level).

**System requirements:**

**Operating System: Windows XP/2007/2010**

**Java :Bluej environment (1.2 or higher version) compatible JDK(1.3 or higher version)**

**Contents:**

|  |  |  |
| --- | --- | --- |
| **Topics:** | **Page no:** | |
| **Java source code** | | **3** |
| **Method Description Table** | | **35** |
| **Variable Description Table** | | **38** |
| **Sample Output** | | **48** |
| **Acknowledgement-Bibliography** | | **49** |
| **Certificate** | | **50** |

**Java Source Code:**

//class Playcards2

import java.util.\*;

import java.io.\*;

public class Playcards2

{

public String cardselect() //

{ /\*to randomly give a card \*/

String card="";

int number=random(14);

switch(number)

{

case 0 :

case 1 :

case 2 :

case 3 :

case 4 :

case 5 :

case 6 :

case 7 :

case 8 :

case 9 :

card =Integer.toString(number);

break;

case 10 : card = "DRAW\_TWO"; break;

case 11 : card = "REVERSE"; break;

case 12 : card = "SKIP"; break;

case 13 : card = "WILD"; break;

case 14 : card = "WILD\_DRAW\_FOUR"; break;

}

if((number!=13)&&(number!=14))

{ int colour=random(3);

switch(colour)

{

case 0 : card =card+" BLUE" ; break;

case 1 : card =card+" GREEN" ; break;

case 2 : card =card+" RED" ; break;

case 3 : card =card+" YELLOW" ; break;

}

}

return card;

}

public int random(int n)

{

return (int)Math.round(Math.random()\*n);

}

public String split(String g,int opt)

{

String cardval=g+" ";//cardval--> info stored in each card

String num="";//num--> to store the number part

String col="";//col--> to store the colour part

String w="";

int count=0;

for(int x=0;x<cardval.length();x++)

{

if (cardval.charAt(x)==' ')

{

if ((count==0)/\*&&(w.length()==1)\*/)num=w;

else col=w;

count++;

w="";

}

else

{

w=w+cardval.charAt(x);

}

}

if(opt==1)

return num;

else

return col;

}

public boolean checkcom(String g,String d)

{

if(split(g,1).equals(split(d,1)))

{

return true;

}

else if(split(g,2).equals(split(d,2)))

{

return true;

}

else if((split(g,1).equals("WILD"))||(split(g,1).equals("WILD\_DRAW\_FOUR")))

return true;

else return false;

}

public void display(String g[][],int cp[],int i)

{

System.out.println("\nYOUR CARDS ARE :\n");

int j=0,q=0;

int pres=cp[i];//hold cards present for current player

while(j<=pres)

{

int cj=j;

System.out.println("-----------------------------------------------------------------");

System.out.println("| | | | |");

while(j-q<4) //1st line

{

try{ show(g[i][j++],0);}//}to show card num if present

catch(Exception e){show("",4);}//ArrayIndexOutOfBounds

}

j=cj;//restoring starting address

System.out.println("|");//2nd lin

while(j-q<4) //2nd line

{

try{show(g[i][j++],1); }//to show card col if present

catch(Exception e){show("",4);}

}

j=cj;

System.out.println("|");//3rd line

while(j-q<4)

{

try

{

String c="("+(int)(j+1)+")";

show(c,3);

j++;

}

catch(Exception e){show("",4);}

}

System.out.println("|");

System.out.println("| | | | |");

System.out.println("-----------------------------------------------------------------");

q+=4;

if(pres==q)break;

}

}

public void show(String g,int p)

{

String num="",col="";

if(g.length()!=1)

{

num=split(g,1);

col=split(g,2);

}

if(p==0)

{

System.out.print("|"+(num));

for(int y=0;y<15-num.length();y++)

System.out.print(" ");

}

else if(p==1)

{

System.out.print("|"+(col));

for(int y=0;y<15-col.length();y++)

System.out.print(" ");

}

else if(p==3)

{

System.out.print("|"+(g));

for(int y=0;y<15-g.length();y++)

System.out.print(" ");

}

else if(p==4)

{

System.out.print("|");

for(int y=0;y<15;y++)

System.out.print(" ");

}

}

public void display(String g)

{

String num=split(g,1);

String col=split(g,2);

System.out.println("\nDISCARD PILE:");

System.out.println("------------------");

System.out.println("| |");

System.out.print("|"+(num));

for(int y=0;y<14-num.length();y++)

System.out.print(" ");

System.out.println(" |");

System.out.print("|"+(col));

for(int y=0;y<12-col.length();y++)

System.out.print(" ");

System.out.println(" |");

System.out.println("| |");

System.out.println("------------------");

}

public int spclcard(String scard)

{

scard+=" ";

int p=scard.indexOf(' ');

if(scard.substring(0,p).equals("DRAW\_TWO")) return 1;

else if(scard.substring(0,p).equals("REVERSE")) return 2;

else if(scard.substring(0,p).equals("SKIP"))return 3;

else if(scard.substring(0,p).equals("WILD"))return 4;

else if(scard.substring(0,p).equals("WILD\_DRAW\_FOUR") ) return 5;

else return 0;

}

public void score(String name,String g[][],int cp[],int posi)

{

int total=0;

int c;

for(int a=0;a<g.length;a++)

{

if(a!=posi)

{

for(int b=0;b<cp[a];b++)

{

c=spclcard(g[a][b]);

switch (c)

{

case 0:total+=(int)g[a][b].charAt(0);

break;

case 1:total+=20;

break;

case 2:total+=20;

break;

case 3:total+=20;

break;

case 4:total+=50;

break;

case 5:total+=50;

}

}

}

}

System.out.println(name+"'s score= "+total);

}

public void exSelSort(String gamers[][],String discard,int cardfunc[],int cardpresent [],int playerpos)

{

boolean flag=false;

for(int i=0;i<cardpresent[playerpos];i++)

{

StringTokenizer st=new StringTokenizer(discard);

String dis=st.nextToken();

if(discard.substring(0,4).equals("WILD")&&discard.substring(0,4).equals(gamers[playerpos][i]))flag=true;

else if(dis.equals("WILD\_DRAW\_FOUR")&&dis.equals(gamers[playerpos][i]))flag=true;

else if(gamers[playerpos][i].equals(discard))

flag=true;

if(flag==true)

gamers[playerpos][i]=gamers[playerpos][i+1];

}

}

}

//class Game\_playUlt2

import java.io.\*;

import java.util.\*;

/\*\* @author (SRINJOY SANTRA)

\* @version (20/10/2015)\*/

public class Game\_playult2 extends Playcards2

{

BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

String discard,drawcard;

public void main()throws IOException

{

File\_Display ob=new File\_Display();//displaying all options

outer:

while(1>0)

{

System.out.println("\f");

ob.masterprint(0);

int ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("\f\nSelect the type of the game:");

System.out.println("Press 1 for player vs player");

System.out.println("Press 2 for player vs computer");

System.out.println("Press 3 for classic mode (1player vs 3computers)");

int opt=0,hp=0,co=0;

try{opt=Integer.parseInt(br.readLine());}

catch(Exception e)

{System.out.println("Invalid Input!");

System.out.println("\nTo continue press any key");

String s=br.readLine();

continue;};

switch (opt)

{

case 1: System.out.println("Enter number of human players:");

hp=Integer.parseInt(br.readLine());co=0;

if(hp<=1)

{ System.out.println("Number of players has to be more than one\n");

System.out.println("\nTo continue press any key");

String s=br.readLine();

continue;

}

if(hp>5)

{ System.out.println("Number of players has to be less than 5\n");

System.out.println("\nTo continue press any key");

String s=br.readLine();

continue;

}

break;

case 2: System.out.println("\nEnter number of human players :");

hp=Integer.parseInt(br.readLine());

System.out.println("\nEnter number of computer players :");

co=Integer.parseInt(br.readLine());

if(hp<1||co<1)

{ System.out.println("Number of players can not be less than one");

break outer;

}

if(hp+co>5)

{ System.out.println("Number of players can not be more than 5");

break outer;

}

break;

case 3: hp=1;co=3;

break;

default:System.out.println("Invalid Input!");

continue;

}

players(hp,co);

break;

case 2 :ob.masterprint(2);

System.out.println("\nPress 1 to see furhter details :");

int i=Integer.parseInt(br.readLine());

if(i==1)ob.masterprint(i);

System.out.println("\nTo continue press any key");

String s=br.readLine();

break;

case 3 :ob.masterprint(ch);

System.out.println("\nDo you really want to exit? Y or N");

s=br.readLine();

if(s.equalsIgnoreCase("Y"))System.exit(0);

break;

}

}

}

public void players(int hp,int co)throws IOException

{

Game\_playult2 ob=new Game\_playult2();

int playerdata=hp+co;

String gamers [][]=new String[playerdata][50];

int cardpresent []=new int[playerdata];

for(int i=0;i<playerdata;i++)//to initialize each gamer with 7 cards

cardpresent [i]=7;

String name []=new String[playerdata];

for(int i=0;i<hp;i++)//to initialize each gamer with a name

{

System.out.println("\nEnter your name player "+(i+1));

name [i]=br.readLine();

}

System.out.println("\nTo continue press 1");

String s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

for(int i=hp;i<playerdata;i++)//to initialize each computr player with a name

{

name [i]="Autobot "+i;

}

int cardfunc []=new int[playerdata];

for(int i=0;i<playerdata;i++)//to initialize each gamer's turn without skip or reverse from before

cardfunc [i]=0;

for(int i=0;i<playerdata;i++)// to initialize each gamer with randomly selected cards

{

for(int j=0;j<cardpresent[i];j++)

{

gamers [i][j]=super.cardselect();

}

}

String discard="";

while(1>0)//to decide the beginning card

{

discard=super.cardselect().trim();

StringTokenizer st=new StringTokenizer(discard);

String cad=st.nextToken();

try

{

if(cad.equals("WILD\_DRAW\_FOUR")||cad.equals("WILD"))

continue;

if(1>0) cardfunc[0]=0;

if(cad.equals("DRAW\_TWO")){cardfunc[0]=1;DRAWTWO(cardpresent,0,gamers);}

if(cad.equals("REVERSE")){cardfunc[0]=2;}

if(cad.equals("SKIP"))cardfunc[0]=3;

super.display(discard);

this.discard=discard;

break;

}

catch(Exception e)

{

continue;

}

}

gamerep(gamers,ob,cardfunc,cardpresent,name,hp);

}

/\*reverses the execution of control in a an array from ascending to descending and vice versa

\* If the control runs in ascending order rev=true ; vice versa \*/

public void gamerep(String gamers[][],Game\_playult2 ob,int cardfunc[],int cardpresent [],String name[],int hp)throws IOException

{

int position;//for indicating the position of next expected player

int begin=1;

int limit;

boolean rev,tmp;

//rev=denotes whether ascen or descen

//tmp to store the return value when the player misses chance to play

if (cardfunc[0]==2) rev=false;//cardfunc[0] indicates reversal

else rev=true;

int i=0,j=-9;

String s;

inwh:

while(begin>0)//to repeat the game till 1 player wins

{

inif:

if(rev)

{

//ascending case

tmp=rev;

limit=name.length;//in ascending loop continued till last

if(j!=-9) i=j;//j==-9 indicates end of array has been reached

else i=0;

inf1:

for(;i<limit;i++)

{

try

{

if(cardpresent[i]==0)//

{

continue inf1;

}

}

catch(Exception e){;}

//separating destinations for human and computer

if(i<hp)rev=shortcut(gamers,ob,cardfunc,cardpresent,rev,i,name);

else if(i<cardfunc.length) rev=autocomp(gamers,ob,cardfunc,cardpresent,rev,i,name);

cardfunc[i]=0;

if(i==limit-1)j=-9;

if(rev==false)

{

//this block infroms the player who was supposed to be playing if the order wasn't reversed but it has been done

if(i+1==limit) position=0;//

else position=i+1;

if (cardfunc.length!=2)

if(position<hp)tmp=shortcut(gamers,ob,cardfunc,cardpresent,rev,position,name);

else if(position<cardfunc.length) tmp=autocomp(gamers,ob,cardfunc,cardpresent,rev,position,name);

if(i-1<0) j=name.length-1;//

else j=i-1;

if(cardfunc[i]==-5)begin=-1;

cardfunc[i]=0;

break inf1;

}

System.out.println("\nTo continue press 1");

s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

}

}

else

{

//descending case

tmp=rev;

limit=0;

if(j!=-9) i=j;

else i=name.length-1;

inf2:

for(;i>=limit;i--)

{

try

{

if(cardpresent[i]==0)

{

continue inf2;

}

}

catch(Exception e){;}

if(i<hp)rev=shortcut(gamers,ob,cardfunc,cardpresent,rev,i,name);

else if(i<cardfunc.length) rev=autocomp(gamers,ob,cardfunc,cardpresent,rev,i,name);

cardfunc[i]=0;

if(i==0)j=-9;

if(rev==true)

{

//this block infroms the player who was supposed to be playing if the order wasnt reversed but it has been done

if(i-1<0) position=name.length-1;//

else position=i-1;

if (cardfunc.length!=2)

if(position<hp)tmp=shortcut(gamers,ob,cardfunc,cardpresent,rev,position,name);

else if(position<cardfunc.length) tmp=autocomp(gamers,ob,cardfunc,cardpresent,rev,position,name);

if(cardfunc[i]==-5)begin=-1;

cardfunc[i]=0;

if(i+1>=name.length) j=0;//

else j=i+1;

break inf2;

}

System.out.println("\nTo continue press 1");

s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

}

}

begin++;

}

}

public boolean shortcut(String gamers[][],Game\_playult2 ob,int cardfunc[],int cardpresent[],boolean rev,int i,String name[])throws IOException

{

boolean drawx=true;//to count wheteher a player has already drawn 1 card

System.out.println("Hi, player "+(name[i])+" it's your turn ;");

int limit=name.length;

if (rev)

{

if(i==0)System.out.println("the player BEFORE you is "+name[limit-1]);

else System.out.println("the player BEFORE you is "+name[i-1]);

if (i==limit-1)System.out.println("the player AFTER you is "+name[0]);

else System.out.println("the player AFTER you is "+name[i+1]);

}

else

{

if(i==0)System.out.println("the player AFTER you is "+name[limit-1]);

else System.out.println("the player after you is "+name[i-1]);

if (i==limit-1)System.out.println("the player BEFORE you is "+name[0]);

else System.out.println("the player BEFORE you is "+name[i+1]);

}

if(cardfunc[i]==1)

{

System.out.println("\nSorry your turn is SKIPPED!");

System.out.println(" You have got a DRAW TWO card!!\n");

super.display(gamers,cardpresent,i);

return rev;

}

else if(cardfunc[i]==2)

{

System.out.println("\nSorry the direction of play is REVERSED!\n");

super.display(gamers,cardpresent,i);

System.out.println("\nTo continue press 1");

String s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

return rev;

}

else if(cardfunc[i]==3)

{

System.out.println("\nSorry your turn is SKIPPED!\n");

super.display(gamers,cardpresent,i);

return rev;

}

else if(cardfunc[i]==5)

{

System.out.println("\nSorry your turn is SKIPPED!");

System.out.println(" You have got a DRAW FOUR card!!!!\n");

super.display(gamers,cardpresent,i);

return rev;

}

else //if((cardfunc[i]==0)||(cardfunc[i]==4))

{ inner :

while(1>0)//to repeat the same iteraration to allow player to play a matching card

{

//String discard=this.discard;

try

{

super.display(this.discard);

System.out.println("To SEE your cards press 1.");

System.out.println("To EXIT the game press 2.");

String s=br.readLine();

if(s.charAt(0)=='1')

{

super.display(gamers,cardpresent,i);

System.out.println("Please see your cards carefully.");

}

else if(s.equals("XLR8"))

cheatcode(gamers,cardpresent,i);

else if(s.charAt(0)=='2')

{

System.out.println("END of the present game.......");

main();

}

else continue;

System.out.println("To SELECT any card press 1");

if (drawx)

System.out.println("To DRAW any card press 2");

else

System.out.println("To PASS press 3");

s=br.readLine();

if(s.charAt(0)=='1')

{

System.out.println("Which card do you want to select between (1) to ("+(cardpresent [i])+")?");

int option=Integer.parseInt(br.readLine())-1;

if (super.checkcom(gamers[i][option],this.discard))

{

this.discard=gamers[i][option];

int playerpos;

int u=super.spclcard(this.discard);

if (rev)

{

if(i+1==gamers.length) playerpos=0;//

else playerpos=i+1;

}

else

{

if(i-1<0) playerpos=gamers.length-1;//

else playerpos=i-1;//

}

cardfunc[playerpos]=u;

switch (u)

{

case 1:DRAWTWO(cardpresent,playerpos,gamers);

break;

case 2: if (rev==false) rev=true;

else if(rev==true) rev=false;

System.out.println("\nTo continue press 1");

s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

break;

case 3:break;

case 4:WILD(ob,1);

break;

case 5:DRAWFOUR(cardpresent,playerpos,gamers);

WILD(ob,1);

break;

default : break;

}

cardfunc[i]=0;

super.exSelSort(gamers,this.discard,cardfunc,cardpresent,i);

cardpresent[i]--;

if(cardpresent[i]==1)

{

System.out.println("!");

String c=br.readLine();

if(c.equalsIgnoreCase("UNO"));

else

{

System.out.println("\nYou didn't say 'UNO' so you have to DRAW two cards!");

DRAWTWO(cardpresent,i,gamers);

}

}

if(cardpresent[i]==0)

{

System.out.println("\fCONGRATULATIONS!! "+name[i]+" YOU WON!!!");

super.score(name[i],gamers,cardpresent,i);

System.out.println("The game is over!\n Return to main menu");

System.out.println("\nTo continue press 1");

s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

main();

}

return rev;

}

else

{

System.out.println("Unmatched card");

continue inner;

}

}

else if((s.charAt(0)=='2')&&(drawx))

{

cardpresent[i]++;

drawx=false;

int flag=0;

while(flag==0)

{

this.drawcard=super.cardselect();

if( this.drawcard.equals("null"))

flag=0;

else

{

flag=1;

}

}

gamers[i][cardpresent[i]-1]=this.drawcard;

continue inner;

}

else if(s.charAt(0)=='3'&&drawx==false)

{

return rev;

}

}

catch(Exception e) {;}

}

}

}

public boolean autocomp(String gamers[][],Game\_playult2 ob,int cardfunc[],int cardpresent[],boolean rev,int i,String name[])throws IOException

{

boolean drawx=true;//to count wheteher a player has already drawn 1 card

System.out.println("Player "+(name[i])+" making its decision in ");

for(int t=0;t<1000000;t++)

{

if(t%10000==0)

System.out.print(" ");

if(t%100000==0)

System.out.print(10-(t/100000));

}

if(cardfunc[i]==1)

{

System.out.println("\n"+name[i]+" turn was SKIPPED!");

System.out.println("He has got a DRAW TWO card!");

super.display(this.discard);

return rev;

}

else if(cardfunc[i]==2)

{

System.out.println("\n the direction of play is REVERSED!");

super.display(this.discard);

System.out.println("\nTo continue press 1");

String s=br.readLine();

if(s.charAt(0)=='1')

System.out.println("\f");

return rev;

}

else if(cardfunc[i]==3)

{

System.out.println("\nHis turn was SKIPPED!");

super.display(this.discard);

return rev;

}

else if(cardfunc[i]==5)

{

System.out.println("\n"+name[i]+" turn was SKIPPED!");

System.out.println("He has got a DRAW FOUR card!");

super.display(this.discard);

return rev;

}

else

{ inner :

while(1>0)//to repeat the same iteraration to allow player to play a matching card

{

super.display(this.discard);

System.out.println("\*"+name[i]+" holds "+cardpresent[i]+" cards");

int combination[]=new int[cardpresent[i]];//selecting a card FROM VALID OPTIONS

int y=0;

for(int x=0;x<cardpresent[i];x++)

{

if (super.checkcom(gamers[i][x],this.discard))

{

combination[y]=x;

y++;

}

}

if((y==0)&&(drawx))//drawing

{

cardpresent[i]++;

drawx=false;

int flag=0;

while(flag==0)

{

this.drawcard=super.cardselect();

if( this.drawcard.equals("null"))

flag=0;

else

{

flag=1;

}

}

gamers[i][cardpresent[i]-1]=this.drawcard;

System.out.println("\*"+name[i]+" has DRAWN a card.");

continue inner;

}

else if(drawx==false)//passing

{

System.out.println("\*"+name[i]+" has PASSED.");

return rev;

}

else

{

int n=super.random(y-1);//n=storing the random postion

int o=combination[n];//o=storing the position from combination[]

if(super.checkcom(gamers[i][o],this.discard)) //checking again wheder the chosen card is valid or not

this.discard=gamers[i][o];

int playerpos;

if (rev)

{

if(i+1==gamers.length) playerpos=0;//

else playerpos=i+1;

}

else

{

if(i-1<0) playerpos=gamers.length-1;//

else playerpos=i-1;//

}

int u=super.spclcard(this.discard);

cardfunc[playerpos]=u;

switch (u)

{

case 1:DRAWTWO(cardpresent,playerpos,gamers);

break;

case 2:if (rev==false) rev=true;

else if(rev==true) rev=false;

break;

case 4: WILD(ob,0);

break;

case 5:DRAWFOUR(cardpresent,playerpos,gamers);

WILD(ob,0);

break;

default : break;

}

cardfunc[i]=0;

System.out.println("\*"+name[i]+" has played a card");

super.exSelSort(gamers,this.discard,cardfunc,cardpresent,i);

cardpresent[i]--;

super.display(this.discard);

System.out.println("\*"+name[i]+" holds "+cardpresent[i]+" cards");

if(cardpresent[i]==1)

{

int c=(int)Math.round(Math.random()\*1);

System.out.println("!");

if(c==0)

{

//checking for not saying uno by the computer player

DRAWTWO(cardpresent,i,gamers);

}

}

if(cardpresent[i]==0)

{

System.out.println("\f\*"+name[i]+" has WON");

super.score(name[i],gamers,cardpresent,i);

for(int g=0;g<cardpresent.length;g++)

cardfunc[g]=-7;

cardfunc[i]=-5;

}

return rev;

}

}

}

}

public void cheatcode(String g[][],int cp[],int i)throws IOException

{

System.out.println("Enter the cards");

for(int j=0;j<cp[i];j++)

{

String user=br.readLine();

if(user.charAt(0)==' ')break;

else

g[i][j]=user;

}

}

void DRAWTWO(int cardpresent[],int playerpos,String gamers[][])

{

int count=0;

while(count<2)

{

cardpresent[playerpos]++;//

gamers[playerpos][cardpresent[playerpos]-1]=super.cardselect();//DRAW TWO

count++;

}

}

void DRAWFOUR(int cardpresent[],int playerpos,String gamers[][])throws IOException

{

int count=0;

while(count<4) //wd4

{

cardpresent[playerpos]++;

gamers[playerpos][cardpresent[playerpos]-1]=super.cardselect();//DRAW fOUR

count++;

}

}

void WILD(Game\_playult2 ob,int x)throws IOException

{

int n=9;

if(x==0)

n=super.random(3);

else if(x==1)

{

System.out.println("Enter the colour:");

System.out.println("1->BLUE");

System.out.println("2->GREEN");

System.out.println("3->RED");

System.out.println("4->YELLOW");

n=Integer.parseInt(br.readLine());

n-=1;

}

String col="";

switch (n)

{

case 0 : col =" BLUE" ; break;

case 1 : col =" GREEN" ; break;

case 2 : col =" RED" ; break;

case 3 : col =" YELLOW" ; break;

}

this.discard+=col;

}

}

//class Game\_playUlt2

import java.util.\*;

import java.io.\*;

public class File\_Display

{

void masterprint(int p)throws IOException

{

Scanner sn=new Scanner(System.in);

FileReader fr;

if(p==0)

fr=new FileReader("wallpaper.txt");

else if(p==2)

fr =new FileReader("unorules1.txt");

else if(p==1)

fr =new FileReader("unorules2.txt");

else if(p==3)

{

fr =new FileReader("feedbac.txt");

}

else fr=new FileReader("unofile.txt");;

BufferedReader br=new BufferedReader(fr);

String txxt;

while((txxt=br.readLine())!=null)

{System.out.println(txxt);}

br.close();

fr.close();

if(p==3){ System.out.print("Your choice? :");

String ho=sn.next();

System.out.println("Thank you. We will try our best to improve in the next version.");

}

}

}

**Method description table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Identifier*** | ***Access Specifier*** | ***Return Type*** | ***Purpose*** | ***Class*** |
| main() | public | void | to ask the user his choice; his choice of game; number of human players and computer players | Game\_playUlt2() |
| players() | public | void | to create cards for players; asks their name; decides the beginning card |
| gamerep() | public | void | to reverse the execution of control in a an array from ascending to descending and vice versa; make the next expected player that course of game has reversed |
| shortcut() | public | boolean | to carry the turn of a human player; to accept, draw a card, pass the chance; to check for “uno” calling from user at the end; to congratulate the winner |
| ***Identifier*** | ***Access Specifier*** | ***Return Type*** | ***Purpose*** | ***Class*** |
| shortcut() | public | boolean | to carry the turn of a computer player; to accept, draw a card, pass the chance(all by making random decisions) ; to check for “uno” calling; to declare the winner | Game\_playUlt2() |
| cheatcode() | public | void | to enable programmer to check the efficiency of the code; to increase chances of win |
| DRAWTWO  () | default | void | to implement action of draw two card |
| DRAWFOUR() | default | void | to implement action of draw four card |
| WILD  (Game\_playult2 ob, int x) | default | void | to implement action of colour changing by computer player or by user player |
| cardselect() | public | String | to generate a random card | Playcards2 |
| random() | public | int | to return a random number within range |
| ***Identifier*** | ***Access***  ***Specifier*** | ***Return***  ***Type*** | ***Purpose*** | ***Class*** |
| checkcom() | public | boolean | to check for matching cards | Playcards2 |
| display(String g[][],int cp[],int i) | public | void | to show cards a player holds |
| show() | public | void | to show required data of card |
| spclcard() | public | int | to effect proper functioning of power cards |
| score() | public | void | to calculate the score of the winner |
| exSelSort() | public | void | to selection sort the remaining cards after one card is played |
| masterprint() | default | void | to show wallpaper, uno rules, feedback | File\_Display |

**Variable description table:**

|  |  |  |  |
| --- | --- | --- | --- |
| class Game\_playUlt2() | | | |
| Identifier | Datatype | Purpose | Scope |
| ch | int | to accept a choice from user among ( new game ,…, exit) | main() |
| opt | int | to accept the type of game  (Player vs. Player ,…., classic mode ) |
| hp | int | to accept the no. of human players |
| co | int | to accept the no. of computer players |
| i | int | to ensure further details of rules are displayed |
| hp | int | to accept the no. of human players | players()  players() |
| co | int | to accept the no. of computer players |
| playerdata | int | to hold total no. of players |
| gamers [][] | String | array to store the cards each player holds during game |
| cardpresent [] | int | array to store no. of cards each player holds |
| name[] | String | array to hold names of each player |
| i | int | for-loop variable |
| s | String | to affirm continuation |
| j | int | for-loop variable |
| discard | String | to hold the initiating card |
| cad | int | to extract each token of a card |
| cardfunc[] | int | to store the power cards that are to be effected |
| position | int | to indicate the position of next expected player | gamerep()  gamerep() |
| begin | int | to continue till a player wins |
| limit | int | to pass positions  from one end to other |
| rev | boolean | denotes ascending or descending traversals |
| temp | boolean | to store the return value when the player misses chance to play |
| i | int | to store the adjacent position |
| j | int | to store the position of the other end |
| gamers [][] | String | array to store the cards each player holds during game |
| cardpresent [] | int | array to store no. of cards each player holds |
| name[] | String | array to hold names of each player |
| hp | int | to accept the no. of human players |
| cardfunc[] | int | to store the power cards that are to be effected |
| rev | boolean | denotes ascending or descending traversals | shortcut()  shortcut() |
| gamers [][] | String | array to store the cards each player holds during game |
| cardpresent [] | int | array to store no. of cards each player holds |
| name[] | String | array to hold names of each player |
| hp | int | to accept the no. of human players |
| i | int | to store the adjacent position |
| cardfunc[] | int | to store the power cards that are to be effected |
| drawx | boolean | to count whether theplayer has already drawn one card |
| limit | int | to store length of array name |
| s | String | to affirm continuation |
| option | int | to select a card from players’ pile |
| playerpos | int | to store the position of next player |
| u | int | to store the power card for next player |
| c | String | to affirm player’s uno calling |
| t | int | for loop variable | autocomp()  autocomp()  autocomp() |
| rev | boolean | denotes ascending or descending traversals |
| gamers [][] | String | array to store the cards each player holds during game |
| cardpresent [] | int | array to store no. of cards each player holds |
| name[] | String | array to hold names of each player |
| hp | int | to accept the no. of human players |
| i | int | to store the adjacent position |
| cardfunc[] | int | to store the power cards that are to be  effected |
| drawx | boolean | to count whether theplayer has already drawn one card |
| combination[] | int | to select the valid card options |
| y | int | to store a cardone at a time |
| x | int | to count number of valid cxards |
| flag | int | to find whether a valid random card is created |
| n | int | to store a random position |
| o | int | to store the element at the random position |
| playerpos | int | to store the position of next player |
| u | int | to store the power card for next player |
| c | String | to randomly decide computer’s uno calling |
| g | int | for loop variable |
| g [][] | String | array to store the cards each player holds during game | cheatcode() |
| cp [] | int | array to store no. of cards each player holds |
| j | int | for loop variable |
| user | String | to store user given string |
| gamers [][] | String | array to store the cards each player holds during game | DRAWTWO() |
| cardpresent [] | int | array to store no. of cards each player holds |
| playerpos | int | to store the position of next player |
| count | int | to count and give the next player two cards |
| gamers [][] | String | array to store the cards each player holds during game | DRAWFOUR()  DRAWFOUR() |
| cardpresent [] | int | array to store no. of cards each player holds |
| playerpos | int | to store the position of next player |
| count | int | to count and give the next player two cards |
| n | int | to randomly generate a colour or accept a colour | WILD(Game\_playult2 ob,int x) |
| col | String | to store the colour |
| x | int | to make the method act differently for computer or human player call |
| class Playcards2 | | | |
| Identifier | Datatype | Purpose | Scope |
| card | String | to create a card | cardselect() |
| number | int | to pick a random  number |
| n | int | to accept the upper limit to create a random number | random() |
| g | String | to accept a card information | split()  split() |
| opt | int | to return as per requirement |
| num | String | to store initial part |
| col | String | to store colour |
| cardval | String | to copy g |
| count | int | to keep track of number or colour extraction |
| w | String | to temporarily hold colour or number |
| x | int | for-loop variable |
| g | String | to accept a card information | checkcom() |
| d | String | to accept discarded card details |
| g[][] | String | to accept all cards of the players | display(String g[][],int cp[],int i) |
| cp[] | int | to accept the number of cards held by players |
| i | int | the player number |
| j | int | while loop variable |
| q | int | to count number of cards showed in previous rows |
| pres | int | to hold cards present for current player |
| cj | int | to copy value of j |
| g | String | to accept a card information | show()  show() |
| p | int | to ensure return of correct data |
| num | String | to store initial part |
| col | String | to store colour |
| y | int | for-loop variable |
| num | String | to store initial part | display(String g) |
| col | String | to store colour |
| y | int | for-loop variable |
| scard | String | to accept the card information | spclcard() |
| p | int | to select effective part of power card |
| name | String | to hold the name of winner | score() |
| g[][] | String | to accept all cards of the players |
| cp[] | int | to accept the number of cards held by players |
| posi | int | to store position of winner in the array |
| a | int | for-loop variable |
| b | int | for-loop variable |
| total | int | to calculate total score |
| c | int | to find category of power card |
| flag | boolean | to indicate card match | exSelSort()  exSelSort() |
| i | int | for-loop variable |
| dis | String | to store initial part of card |
| gamers [][] | String | array to store the cards each player holds during game |
| cardpresent [] | int | array to store no. of cards each player holds |
| discard | String | to hold the initiating card |
| cardfunc[] | int | to store the power cards that are to be effected |
| playerpos | int | to store the position of next player |
| Class File\_Display | | | |
| p | int | to store user option | masterprint() |
| txxt | String | to store each line of file |
| ho | String | to store choice |

**Sample Output:**

**Acknowledgement:**

I would like to pay my gratitude to my parents, Mr Kasinath Santra and Mrs Saswati Santra for their support.

I am indebted to my school computer Science teacher, Mr Abhishek Das for approving the topic and advising on improvement possibilities.

**Bibliography:**

Computer Science with Java, Class XII, Sumita Arora.

**Certificate:**

This project has been successfully completed within .

Signature of School Teacher:

Signature of Visiting Examiner: