```
package uno;

import java.awt.*;
import java.net.URL;
import java.util.ArrayList;
import java.util.Random;
import java.util.Stack;

import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;

import javax.swing.ImageIcon;

/*-------------------------------------------------------------------------------
-------------*
 * ----------------------------------UNO : GG Edition--------------------------
-------------*
 *                            Presented to you by : Ahmed Abdultawab & Mohammed
Zakaria                          *
 * -------------------------------------------------------------------------------
-------------*
 * (Game Description goes here)
                                                *
 *-------------------------------------------------------------------------------
-------------*/
/*-------------------------------------------------------------------------------
-------------*
 * WARNING : You're not playing UNO. You're playing UNO : GG Edition !
                          *
 * with all new updated HD graphics, and silky smooth 60 fps gameplay.
                          *
 *-------------------------------------------------------------------------------
-------------*/

/*TO DO List :
 *
 * -
 * -
 * -
 * - clean up
 * -
 * -  cant pass on first turn
 *
 */

public class Game {


      public static boolean cardDrawn;
      public static boolean wildCardPlayed;
      public static boolean winAnimation ;
      public static String topColor;
      int cardx = 1264;
      int cardy = 364;
      double counter =0;
      ArrayList<Integer> temp ;
      Player p1;
      Player cpu1;
```

```
Player cpu2;
Player cpu3;
Deck deck;
Stack<Card> playStack;
Queue<Player> playerQ;
static DrawWindow display;
/*
 *
 */
private boolean messageShown;

private boolean isEligibleSaving;

public static void main(String[] args) {
      // sample code to run the game
      DrawWindow display = new DrawWindow(1600,900);
      display.setVisible(true);
      display.setTitle("UNO : GG Edition");
      Game game = new Game(display);
      game.play();
}

// Game Constructor

public Game(DrawWindow display) {

      this.display = display;
      wildCardPlayed = false;
      winAnimation = false;
      cardDrawn = false;
      p1 = new Player(true,"P1");
      cpu1 = new Player(false,"CPU1");
      cpu2 = new Player(false,"CPU2");
      cpu3 = new Player(false,"CPU3");
      deck = new Deck();
      playStack = new Stack<Card>();
      playerQ = new Queue<Player>();
      playerQ.push(p1);
      playerQ.push(cpu1);
      playerQ.push(cpu2);
      playerQ.push(cpu3);

      p1.drawCards(deck, 5, playStack);
      cpu1.drawCards(deck, 5, playStack);
      cpu2.drawCards(deck, 5, playStack);
      cpu3.drawCards(deck, 5, playStack);



}

// plays the game
private void play() {
      // create display
      if (!winAnimation){

      // DRAWING GAME ASSETS
```

```
            display.drawImage(0, 0, 1600, 900, "poker-table-background.jpg");

            // helper

            //drawing player queue

            display.drawImage(35, 25, 458, 125, "PlayerQueue.png");
            for (int i=0; i<playerQ.size(); i++){
                  if (playerQ.getElement(i).equals(p1)){
                        display.drawText(435 - (i*80), 85, 33, "white", "P1");
                  }else if(playerQ.getElement(i).equals(cpu1)){
                        display.drawText(435- (i*80), 85, 27, "white", "CPU1");
                  } else if(playerQ.getElement(i).equals(cpu2)) {
                        display.drawText(435 - (i*80), 85, 27, "white", "CPU2");
                  }else if(playerQ.getElement(i).equals(cpu3)) {
                        display.drawText(435- (i*80), 85, 27, "white", "CPU3");
                  }
            }
            if (playerQ.getElement(1).equals(cpu1)){
                  display.drawText(220, 150, 22, "white", "Turns Are Moving
Clockwise");
            }else{
                  display.drawText(260, 150, 22, "white", "Turns Are Moving
Counter-Clockwise");
            }

            //drawing the deck
            for (int i=1; i <35; i++){
                  display.drawImage(1400-(i+1), 625-(i+1), 172, 264,
"CardBack.png");
            }
            display.drawText(1500, 805,30,"white",
String.valueOf(deck.getcardCount()));

            //drawing the playStack

            display.drawImage(650, 240, 308, 326, "CardStack.png");
            if (!playStack.isEmpty()){
                  display.drawImage(700, 270, 170, 260,
playStack.peek().imgString());//change to playstack.peek.imgString
            }

            // drawing player hand
            for (int i=0; i <p1.handSize(); i++){
                  if (i >=14){
                        display.drawImage(230+(150*(i-14)), 750, 215,
330,p1.getCard(i).imgString());}
                  else if (i>=7){
                        display.drawImage(215+(150*(i-7)), 670, 215, 330,
p1.getCard(i).imgString());}
                  else{
                        display.drawImage(200+(150*i), 570, 215, 330,
p1.getCard(i).imgString());}}

            // drawing the pass button

            display.drawImage(35,730, 128, 120, "Pass.png");
```

```
            //drawing opponents


            display.drawImage(-125,300, 325, 275, "Opponent.png");
            display.drawImage(625,-125, 275, 325, "OpponentTop.png");
            display.drawImage(1725,300, -325, 275, "Opponent.png");


            display.drawText(15, 435,35,"black",
String.valueOf(cpu1.handSize()));
            display.drawText(765, 15,35,"black",
String.valueOf(cpu2.handSize()));
            display.drawText(1575, 435,35,"black",
String.valueOf(cpu3.handSize()));

            // special cases

            if (wildCardPlayed){
                  wildCardPlayed = false;
            }

            if (!playStack.isEmpty()){
            if (playStack.peek().getColor().equals("wild")){
                  display.drawText(1300,15,25,"white","Wild Card enforced the
color: " +  Game.topColor );
            }}

      }
            // main game loop
            while (true) {
                  // pause 100 milliseconds to wait for user click
                  try {
                        Thread.sleep(100);
                  } catch (Exception e) {
                  }

                  // check if the user clicked
                  int[] click = display.checkMouse();
                  if (click != null) {
                        // the user clicked
                        int Clickx = click[0];
                        int Clicky = click[1];

                        // process click on (row, col)
                        System.out.println("mouse coordinates : " +Clickx + " " +
Clicky);
                        // for (int i; i<hand.size), if between i*value --> play
card at index i


                        //draw card
                        if (!winAnimation ){

                              if (Clickx > 1370 && Clickx <1529 && Clicky < 848
&& Clicky>594){
                                    if (!cardDrawn){
                                          p1.drawCards(deck, 1, playStack);
                                          cardDrawn= true;
```

```
                                    play();

                              } else{
                                    System.out.println("You've already drawn
a card this turn!");
                              }

                        }

                        // pass
                        if (Clickx > 35 && Clickx <163 && Clicky < 850 &&
Clicky>730){
                              System.out.println("PASS!");
                              playerQ.push(playerQ.pop());
                              playerQ.peek().playTurn(playStack, deck,
playerQ);
                              play();

                        }

                        // Playing a card
                        for (int i=0; i<p1.handSize(); i++){

                              if (p1.handSize() >= 14){
                                    if (Clickx>(230+(150*(i-14))) &&
Clickx<(380+(150*(i-14))) && Clicky>750 && Clicky<900){
                                          p1.playCard(playStack, i, deck,
playerQ);
                                          if (p1.handSize()== 0){
                                                win(p1);
                                          }else{
                                          play();}
                                          break;
                                    }

                                    if (Clickx>(215+(150*(i-7))) &&
Clickx<365+(150*(i-7)) && Clicky>670 && Clicky<750){
                                          p1.playCard(playStack, i, deck,
playerQ);
                                          if (p1.handSize()== 0){
                                                win(p1);
                                          }else{
                                          play();}
                                          break;
                                    }
                                    if (Clickx>200+(i*150) &&
Clickx<350+(i*150) && Clicky>572 && Clicky<670){
                                          p1.playCard(playStack, i, deck,
playerQ);
                                          if (p1.handSize()== 0){
                                                win(p1);
                                          }else{
                                          play();}
                                          break;
                                    }

                              }
                              else if (p1.handSize()>7){
```

```
                                        if (Clickx>(215+(150*(i-7))) &&
Clickx<365+(150*(i-7)) && Clicky>670 && Clicky<900){
                                              p1.playCard(playStack, i, deck,
playerQ);
                                              if (p1.handSize()== 0){
                                                    win(p1);
                                              }else{
                                              play();}
                                              break;
                                        }
                                        if (Clickx>200+(i*150) &&
Clickx<350+(i*150) && Clicky>572 && Clicky<670){
                                              p1.playCard(playStack, i, deck,
playerQ);
                                              if (p1.handSize()== 0){
                                                    win(p1);
                                              }else{
                                              play();}
                                              break;
                                        }
                                  }
                                  else if (p1.handSize()<=7){

                                  if (Clickx>200+(i*150) && Clickx<350+(i*150)
&& Clicky>572 && Clicky<900){
                                        p1.playCard(playStack, i, deck,
playerQ);
                                        if (p1.handSize()== 0){
                                              win(p1);
                                        }else{
                                        play();}
                                        break;
                                  }
                                  }
                                  }

                        }

                        else{

                        }
                  }
            }
      }

      public static void updateStack(Stack<Card> playStack){

            if (!playStack.isEmpty()){
                  display.drawImage(700, 270, 170, 260,
playStack.peek().imgString());//change to playstack.peek.imgString
            }
      }


      public static void playWildCard(){

            Random r = new Random();
```

```
            int randomint = r.nextInt(4);
            if (randomint == 0){
                  Game.topColor = "blue";}
            else if (randomint == 1){
                  Game.topColor = "red";}
            else if (randomint == 2){
                  Game.topColor = "yellow";}
            else if (randomint == 3){
                  Game.topColor = "green";}
            display.drawText(1300,15,25,"white","Wild Card enforced the color: "
+  Game.topColor );
            Game.wildCardPlayed = true;
      }

      public static void win(Player player){
            display.drawImage(0, 0, 1600, 900, "poker-table-background.jpg");
            display.drawText(800, 350, 50, "white", player.getName() + " HAS WON
!");

            Game.winAnimation = true;
      }

      public static void updateDisplay(Player player, Queue<Player> playerQ) {

            Player P1 = null;
            Player CPU1 = null ;
            Player CPU2 = null;
            Player CPU3 = null;

            for (int i=0; i<playerQ.size();i++ ){
                  if (playerQ.getElement(i).getName().equals("P1")){
                        P1 = playerQ.getElement(i);
                  }else if (playerQ.getElement(i).getName().equals("CPU1") ){
                        CPU1 = playerQ.getElement(i);
                  }else if (playerQ.getElement(i).getName().equals("CPU2")){
                        CPU2 = playerQ.getElement(i);
                  }else {
                        CPU3 = playerQ.getElement(i);
                  }
            }

            if (player.equals(CPU1)){
                  display.drawImage(-125,300, 325, 275, "Opponent.png");
                  try {
                        Thread.sleep(500);
                  } catch (Exception e) {}
                  display.drawText(15, 435,35,"black",
String.valueOf(player.handSize()));
            }

            else if (player.equals(CPU2)){
                  display.drawImage(625,-125, 275, 325, "OpponentTop.png");
                  try {
                        Thread.sleep(500);
                  } catch (Exception e) {
                  }
                  display.drawText(765, 15,35,"black",
String.valueOf(player.handSize()));}
```

```
            else{
                  display.drawImage(1725,300, -325, 275, "Opponent.png");
                  try {
                        Thread.sleep(500);
                  } catch (Exception e) {
                  }
                  display.drawText(1575, 435,35,"black",
String.valueOf(player.handSize()));}


            display.drawImage(35, 25, 458, 125, "PlayerQueue.png");


            for (int i=0; i<playerQ.size(); i++){
                  if (playerQ.getElement(i).equals(P1)){
                        display.drawText(435 - (i*80), 85, 33, "white", "P1");
                  }else if(playerQ.getElement(i).equals(CPU1)){
                        display.drawText(435- (i*80), 85, 27, "white", "CPU1");
                  } else if(playerQ.getElement(i).equals(CPU2)) {
                        display.drawText(435 - (i*80), 85, 27, "white", "CPU2");
                  }else if(playerQ.getElement(i).equals(CPU3)) {
                        display.drawText(435- (i*80), 85, 27, "white", "CPU3");
                  }
            }
            if (playerQ.getElement(1).equals(CPU1)){
                  display.drawText(220, 150, 22, "white", "Turns Are Moving
Clockwise");
            }else{
                  display.drawText(260, 150, 22, "white", "Turns Are Moving
Counter-Clockwise");
            }

            display.drawImage(175, 569,1181,331, "PlayerHandUpdate.jpg");

            for (int i=0; i <P1.handSize(); i++){
                  if (i >=14){
                        display.drawImage(230+(150*(i-14)), 750, 215,
330,P1.getCard(i).imgString());}
                  else if (i>=7){
                        display.drawImage(215+(150*(i-7)), 670, 215, 330,
P1.getCard(i).imgString());}
                  else{
                        display.drawImage(200+(150*i), 570, 215, 330,
P1.getCard(i).imgString());}}


      }

}
```