ITP277: Milestone 4

The goal for this milestone is to revise and refine what we have developed thus far. The main objective is to decompose a complex program into modular, specific tasks. You want to think of your driver program as a coordinator of actions between objects. To develop a correct, logical and easy to debug solution, you will need to use *abstraction*.

1. Activity Directions: Use a flowcharting tool to refine your game logic flowchart (from Milestone 1). Use the functionality in your newly developed classes as a guide and write the flowchart using objects and object methods. Test your flowchart carefully by trying to play a game of UNO using the flowchart as your rule set. Revise as needed, adding functionality to your classes if needed.
2. Rewrite your driver program so that it simulates a game of UNO. Enforce as many of the standard UNO rules as you can. At a minimum, the game should force users to play legal matching cards. The game should detect a winner and stop. Create a video in which you walk through your flowchart and explain the role of the different objects in your game. Also, run your program and discuss any issues you plan to address in before your final version is complete.
3. Submit the following:
   1. Revised flowchart
   2. Revised classes (submit .java files in a single zip file)
   3. Revised driver program
   4. Link to your video

Ask your instructor for specific submissions details.

Text

Description automatically generated

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Scanner;  
  
public class Test {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 Deck d = new Deck();  
 Player p1 = new Player();  
 Player p2 = new Player();  
 Player p3 = new Player();  
 ArrayList<Player> players = new ArrayList<>();  
 players.add(p1);  
 players.add(p2);  
 players.add(p3);  
 Card currentCard;  
 d.newDeck();  
 int startVal =0;  
 int pickCard;  
 boolean winLose = false;  
  
 //Dealing Hand  
  
 for(int x=0;x<7;x++) {  
 p1.hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 p2.hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 p3.hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
  
 //Gets top card  
 currentCard = d.cards.get(0);  
  
 do {  
 //Begins Turns  
 for (int x = startVal; x < 3; x++) {  
 System.*out*.println(players.get(x).getHand());  
 System.*out*.println(currentCard);  
 System.*out*.println("Pick a card to play or draw a card >>> 1.Play Card 2.Draw Card");  
 int pd = sc.nextInt();  
  
 //If Play card  
 if (pd == 1) {  
 if (*hasEqual*(players.get(x), currentCard)) {  
 do {  
 System.*out*.println("Pick the card to play- enter the number of the card in your hand");  
 pickCard = sc.nextInt() - 1;  
  
 } while (pickCard >= players.get(x).hand.size());  
  
 currentCard = players.get(x).hand.get(pickCard);  
 players.get(x).hand.remove(pickCard);  
  
 if (currentCard.getValue() == Card.Value.*Skip*) {  
 if (x == 2)  
 x = 0;  
 else  
 x++;  
 } else if (currentCard.getValue() == Card.Value.*Reverse*) {  
 Collections.*reverse*(players);  
 } else if (currentCard.getValue() == Card.Value.*Wild* || currentCard.getValue() == Card.Value.*W4*) {  
 System.*out*.println("Enter a color to change to >>> 1.Blue 2.Red 3.Green 4.Yellow");  
 int colorChange = sc.nextInt();  
  
 switch (colorChange) {  
 case 1:  
 currentCard.setColor(Card.Color.*Blue*);  
 break;  
 case 2:  
 currentCard.setColor(Card.Color.*Red*);  
 break;  
 case 3:  
 currentCard.setColor(Card.Color.*Green*);  
 break;  
 case 4:  
 currentCard.setColor(Card.Color.*Yellow*);  
 break;  
 default:  
 currentCard.setColor(Card.Color.*Blue*);  
 break;  
 }  
  
 if (currentCard.getValue() == Card.Value.*W4*) {  
 if (x == 2) {  
 for (int y = 0; y < 4; y++) {  
 players.get(0).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
 } else {  
 for (int y = 0; y < 4; y++) {  
 players.get(x + 1).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
 }  
 }  
 } else if (currentCard.getValue() == Card.Value.*D2*) {  
 if (x == 2) {  
 for (int y = 0; y < 2; y++) {  
 players.get(0).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
 } else {  
 for (int y = 0; y < 2; y++) {  
 players.get(x + 1).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
 }  
 }  
  
 }  
 //If no cards to play  
 else {  
 System.*out*.println("You don't have a card to play");  
 System.*out*.println("Drawing Card");  
 players.get(x).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
 }  
 //If Draw Card  
 else if (pd == 2) {  
 System.*out*.println("Drawing Card");  
 players.get(x).hand.add(d.cards.get(0));  
 d.cards.remove(0);  
 }  
  
 if(players.get(x).hand.isEmpty())  
 {  
 winLose=true;  
 }  
  
 }//end of for loop for turns  
  
 }while(!winLose);  
 }  
  
 public static boolean hasEqual(Player p,Card current)  
 {  
 boolean check = false;  
 for(int x=0;x<p.hand.size();x++)  
 {  
 if(current.getColor()==p.hand.get(x).getColor() || current.getValue()==p.hand.get(x).getValue())  
 {  
 check =true;  
 break;  
 }  
 }  
 return check;  
 }  
}