For: Alexander Hansen

Assignment: BlackJack

GitHub URL: https://github.com/ah0912784/CIS171HansenA.git

Student: Please answer the questions, then use the Insert, Screenshot option in Word to snip an appropriate sample of your executing program’s output.

Copy the code from your .java file(s) into the code section below. Your code should match the code submitted in GitHub.

Be sure to review your graded assignment for instructor comments!

|  |
| --- |
| **Analysis** |
| The goal was to create a game of black jack with a fully functioning GUI |
|  |

|  |
| --- |
| **Design** |
| I used two classes. The first class, BlackJackGUI, and the second BlackJackFunctions. One of them basically contained all of my GUI elements and the other the methods. I ran most of these through action events on buttons. Otherwise I used various get functions to get what variables I needed. I used Arrays and Strings like it was no ones business and that helped create the monster I have now. |
|  |

|  |
| --- |
| **Testing** |
| After I got a rough GUI in place. I started working on my methods in the BJF class. I decided I wants most of my logic happening there so I could find errors more easily, and keep the code neatish.  I for each function I went through and my variables print to the console so I could see what they were doing. The hardest part was trying to tune the CheckForWin() method so that it was fair. |
|  |

|  |
| --- |
| **Screenshot(s)** |
|  |
|  |

|  |
| --- |
| **Code** |
| import java.util.ArrayList;  import javafx.application.Application;  import javafx.event.ActionEvent;  import javafx.event.EventHandler;  import javafx.scene.Scene;  import javafx.scene.control.Button;  import javafx.scene.control.Label;  import javafx.scene.control.TextField;  import javafx.scene.image.Image;  import javafx.scene.image.ImageView;  import javafx.scene.layout.GridPane;  import javafx.scene.layout.HBox;  import javafx.scene.layout.Pane;  import javafx.stage.Stage;  /\*\*  \*  \* @author rocke  \*/  public class BlackJackGUI extends Application{  BlackJackFunctions func = new BlackJackFunctions();    @Override  public void start(Stage primaryStage) throws Exception {  BlackJackFunctions func = new BlackJackFunctions();  func.shuffle();  primaryStage.setTitle("Black Jack");  GridPane pane = new GridPane();      //creating buttons  Button stayB = new Button("Stay");  Button hitB = new Button("Hit");  Button dealB = new Button("Deal");  Button resetB = new Button("Reset");  Button newRound = new Button("New Round");  //actions  dealB.setOnAction((ActionEvent event) -> {  func.Deal();  createImg(4,func.getPlayerHand(),pane);  createImgDealer(pane, func.getDealerHand());  func.calculatePoints(func.getDealerHand(), func.dPoints);  });    hitB.setOnAction((ActionEvent event) -> {  func.Hit();  createImg(4,func.getPlayerHand(),pane);  });    stayB.setOnAction((ActionEvent event) -> {  func.Stay();  createImg(0,func.getDealerHand(),pane);  textBoxes(pane, func.getWins(),func.getLosses());  });    resetB.setOnAction((ActionEvent event) -> {  removeImg(pane, func.getPlayerHand(), func.getDealerHand());  func.Reset();  func.shuffle();  //pane.getChildren(ImageView);  });  newRound.setOnAction((ActionEvent event)-> {  removeImg(pane, func.getPlayerHand(), func.getDealerHand());  func.newRound();  func.shuffle();      });  //adding to pane  pane.add(dealB, 0, 5);  pane.add(hitB, 1,5);  pane.add(stayB, 2, 5);  pane.add(resetB, 3, 5);  pane.add(newRound, 4, 5);  //text fields  textBoxes(pane, func.getWins(),func.getLosses());            //setting the scene  // pane.setHgap(71);  // pane.setVgap(96);  Scene scene = new Scene(pane, 600, 600);  primaryStage.setScene(scene);  primaryStage.show();          }  public static void main(String[] args) {  launch(args);  }    public void createImg(int integer,ArrayList<String> ary, GridPane pane) {  BlackJackFunctions func = new BlackJackFunctions();    for (int i = 0; i < ary.size(); i++) {  String img = func.getCardImage(i, ary);  pane.add(new ImageView(new Image(img)),i, integer);  }    }  public void createImgDealer(GridPane pane, ArrayList<String> ary) {  BlackJackFunctions func = new BlackJackFunctions();    String img = func.getCardImage(0, ary);  pane.add(new ImageView(new Image(img)),0,0);  pane.add(new ImageView(new Image("cards\_png/b1fv.png")),1,0);  }  public void removeImg(GridPane pane, ArrayList<String> ph, ArrayList<String> dh) {  BlackJackFunctions func = new BlackJackFunctions();    for(int i = 0; i < ph.size(); i++) {  pane.add(new ImageView(new Image("cards\_png/b1fv.png")),i,4);  }  for(int i = 0; i < dh.size(); i++) {  pane.add(new ImageView(new Image("cards\_png/b1fv.png")),i,0);  }  }  public void textBoxes(GridPane pane, String w, String L) {  TextField winsT = new TextField(w);  TextField lossesT = new TextField(L);    Label winsL = new Label("Wins: ");  Label lossesL = new Label("Losses: ");  HBox hbox = new HBox();    hbox.getChildren().addAll(winsL, winsT,lossesL,lossesT);  winsT.setPrefWidth(25);  lossesT.setPrefWidth(25);  pane.add(hbox,0,3 );  }    }  import java.util.\*;  import static java.lang.Integer.parseInt;  public class BlackJackFunctions {    private final String[] baseDeck = {"c1","c2","c3","c4","c5","c6","c7","c8","c9","c10","cj",  "cq","ck","s1","s2","s3","s4","s5","s6","s7","s8","s9","s10","sj","sq","sk",  "d1","d2","d3","d4","d5","d6","d7","d8","d9","d10","dj","dq","dk","h1","h2",  "h3","h4","h5","h6","h7","h8","h9","h10","hj","hq","hk"};    private ArrayList<String> shuffledDeck = new ArrayList<>();  private ArrayList<String> playerHand = new ArrayList<>();  private ArrayList<String> dealerHand = new ArrayList<>();  int baseDeckLength = baseDeck.length;  int pPoints = 0;  int dPoints = 0;  int wins = 0;  int losses = 0;      public void Deal() {    for(int i = 0; i <= 1; i++) {  String t = addCard();  dealerHand.add(t);  System.out.println("this is t "+t);  }  for(int i = 0; i <= 1; i++) {    String s = addCard();  System.out.println("this is s "+s);  playerHand.add(s);    }  }  public int calculatePoints(ArrayList<String> array, int points)  {    ArrayList ary = array;    for (int i = 0; i < ary.size(); i++ ){  System.out.print(ary);  String s = (String) ary.get(i);  char a = s.charAt(s.length()-1);  String b = Character.toString(a);  switch (a) {  case '1':  points += 11;  break;  case '0':  case 'j':  case 'q':  case 'k':  points += 10;  break;  default:  points += parseInt(b);  break;  }  }  points = oneOrEleven(ary, points);  System.out.println(points);  return points;  }  public void newRound() {  shuffledDeck.clear();  playerHand.clear();  dealerHand.clear();  System.out.println(shuffledDeck);  pPoints = 0;  dPoints = 0;  }  public void Reset() {  shuffledDeck.clear();  playerHand.clear();  dealerHand.clear();  wins = 0;  losses = 0;  }  public void Stay(){  int cond;  dPoints = calculatePoints(dealerHand, dPoints);  CheckForWin();  while (pPoints > dPoints ) {    CheckForWin();  String a = addCard();  dealerHand.add(a);  dPoints = calculatePoints(dealerHand, dPoints);    if (dPoints > pPoints) {  break;  }  else if (dPoints > 21) {  break;  }      }  }    public void Hit() {  String s = addCard();  playerHand.add(s);  pPoints = calculatePoints(playerHand, pPoints);    }  public String addCard() {  String s = new String();  Random r = new Random();    int x = r.nextInt(shuffledDeck.size());  //System.out.print(x);  s = shuffledDeck.get(x);  //System.out.println(shuffledDeck);  shuffledDeck.remove(x);  //System.out.println(shuffledDeck);  //System.out.println(s);  return s;  }  public void CheckForWin() {    int a = 0;  if (dPoints > 21) {  wins++;  System.out.println("You Win"+wins);  a = 0;  }  else if (pPoints == 21) {  wins++;  System.out.println("You win "+wins);  a = 1;  }  else if ( pPoints > 21) {  losses++;  System.out.println("Busted");    }  else if (dPoints == pPoints) {  System.out.println("Draw");    }  else if (dPoints > pPoints) {  if(dPoints > 21) {  wins++;  }  else{  losses++;  System.out.println("Beat by the dealer");  }  }            }  public ArrayList loadDeck(){  shuffledDeck.clear();  for (int i = 0; i <= baseDeck.length-1; i++) {  String s = baseDeck[i];  shuffledDeck.add(s);  }  //System.out.println(shuffledDeck);  return shuffledDeck;  }  public ArrayList shuffle() {  loadDeck();    Collections.shuffle(shuffledDeck);  return shuffledDeck;  }  public int hasAce(ArrayList ary) {  int aces = 0;  for (int i = 0; i < ary.size(); i++) {  String s = (String) ary.get(i);  char a = s.charAt(s.length()-1);  if (a == '1') {  aces++;  }  }  return aces;  }    public int oneOrEleven(ArrayList<String> array, int points) {    int aceNum = hasAce(array);  while (aceNum > 0 && points > 21) {  points -= 10;  }        return points;  }    public String getCardImage(int x, ArrayList<String> ary) {  //System.out.println(ary);  String s = ary.get(x);  String line = "cards\_png/"+s+".png";  return line;  }  public ArrayList<String> getDealerHand() {  return dealerHand;  }    public ArrayList<String> getPlayerHand() {  return playerHand;  }  public ArrayList<String> getshuffledDeck() {  return shuffledDeck;  }  public String getWins() {  String winS = Integer.toString(wins);  return winS;  }  public String getLosses() {  String losseS = Integer.toString(losses);  return losseS;  }  } |
|  |