In this project you will create a solution designed to play Blackjack. We will use simple rules of the game. Assume there exist a player and a dealer. The winner of the game is the player who gets closest to 21 without exceeding 21 or “going bust”. I f both players have the same points, then neither players wins the hand. An ace has a value of 1 or 11. Face cards have a value of 10. The dealer’s hand is worth 16 or less. The dealer my not take a card if the hand is worth 17 or more. As the player, you may take a card at any time.

Allow the user to play as many hands as he/she wants.

Keep track of how many hands the dealer wins and how many hands the player wins.

**Create a class called Card.** The Card has three fields, value and suit, face.

The Card class has a constructor that takes three values of the three fields.

Do not create a no-args constructor.

The Card class has three get methods to return the values of each of the fields. The card class has toString() method.

The Card class has a compareTo() method that uses the value of the card to determine the ordering. For this implementation the ordering is as follows: (high to low) A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2

**Create a class called Deck.** A deck has two fields- an array of Cards and an int which holds the subscript of the Card at the top of the deck.

The Deck class will have only one constructor, which will create the array. Try to think of a way to code this without writing 52 statements.

The Deck class also has a method getTopCard() which returns a **copy** of the Card that is at the top of the array.

The Deck class has a method called shuffle() which shuffles the array of Cards. To shuffle, generate random numbers and use these to exchange array elements. Exchange 1000 times in this method. If you are unsure of how to generate a random number after reading the Java API documentation on the Random class, please ask in class.

**Create the driver class-call it Game21**

This is where your game logic will go. As a hint, you will need to create one Deck instance and then call the shuffle method() for each. When you are texting as you write, before you shuffle() you may want to print out the entire Deck to make sure you have created the array Cards correctly.

Each player is dealt two cards to start. Then the rules stated above are followed. Only the user has the choice whether to deal another card or quit. The dealer (computer) must be dealt another card from its deck if its total is 16 or under and must quit if its hand is 17 or over. Once a hand is won, either by the computer or the player, allow the player to decide if he/she wants to play another hand. Keep a running total of the number of hands won by the computer and the number of hands won by the player.

**BlackjackApp.java**

package com.almasb.blackjack;

import javafx.application.Application;  
import javafx.beans.property.SimpleBooleanProperty;  
import javafx.beans.property.SimpleStringProperty;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.layout.Region;  
import javafx.scene.layout.StackPane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Rectangle;  
import javafx.scene.text.Text;  
import javafx.stage.Stage;  
public class BlackjackApp extends Application {

private Decks dk = new Decks();  
private Hands deals, player;  
private Text mssg = new Text();

private SimpleBooleanProperty playin = new SimpleBooleanProperty(false);

private HBox dealCrd = new HBox(20);  
private HBox palyCrd = new HBox(20);

private Parent drawingHands() {  
deals = new Hands(dealCrd.getChildren());  
player = new Hands(palyCrd.getChildren());

Pane rt = new Pane();  
rt.setPrefSize(800, 600);

Region reg = new Region();  
reg.setPrefSize(800, 600);  
reg.setStyle("-fx-reg-color: rgba(0, 0, 0, 1)");

HBox rtLaying = new HBox(5);  
rtLaying.setPadding(new Insets(5, 5, 5, 5));  
Rectangle ltBG = new Rectangle(550, 560);  
ltBG.setArcWidth(50);  
ltBG.setArcHeight(50);  
ltBG.setFill(Color.GREEN);  
Rectangle rtBG = new Rectangle(230, 560);  
rtBG.setArcWidth(50);  
rtBG.setArcHeight(50);  
rtBG.setFill(Color.ORANGE);

VBox ltVBox = new VBox(50);  
ltVBox.setAlignment(Pos.TOP\_CENTER);

Text dlrSc = new Text("Dealer: ");  
Text plyrScre = new Text("Player: ");

ltVBox.getChildren().addAll(dlrSc, dealCrd, mssg, palyCrd, plyrScre);  
VBox rtVBox = new VBox(20);  
rtVBox.setAlignment(Pos.CENTER);

final TextField tf = new TextField("BET");  
tf.setDisable(true);  
tf.setMaxWidth(50);  
Text price = new Text("MONEY");

Button btnPlay = new Button("PLAY");  
Button btnHit = new Button("HIT");  
Button btnStand = new Button("STAND");

HBox btnHBox = new HBox(15, btnHit, btnStand);  
btnHBox.setAlignment(Pos.CENTER);

rtVBox.getChildren().addAll(tf, btnPlay, price, btnHBox);  
rtLaying.getChildren().addAll(new StackPane(ltBG, ltVBox), new StackPane(rtBG, rtVBox));  
rt.getChildren().addAll(reg, rtLaying);  
btnPlay.disableProperty().bind(playin);  
btnHit.disableProperty().bind(playin.not());  
btnStand.disableProperty().bind(playin.not());

plyrScre.textProperty().bind(new SimpleStringProperty("Player: ").concat(player.valProps().asString()));  
dlrSc.textProperty().bind(new SimpleStringProperty("Dealer: ").concat(deals.valProps().asString()));

player.valProps().addListener((obs, old, newValue) -> {  
if (newValue.intValue() >= 21) {  
theEnd();  
}  
});

deals.valProps().addListener((obs, old, newValue) -> {  
if (newValue.intValue() >= 21) {  
theEnd();  
}  
});

btnPlay.setOnAction(event -> {  
newGame();  
});

btnHit.setOnAction(event -> {  
player.takeCard(dk.withdrawCrd());  
});

btnStand.setOnAction(event -> {  
while (deals.valProps().get() < 17) {  
deals.takeCard(dk.withdrawCrd());  
}

theEnd();  
});

return rt;  
}

private void newGame() {  
playin.set(true);  
mssg.setText("");

dk.refill();

deals.newOne();  
player.newOne();

deals.takeCard(dk.withdrawCrd());  
deals.takeCard(dk.withdrawCrd());  
player.takeCard(dk.withdrawCrd());  
player.takeCard(dk.withdrawCrd());  
}

private void theEnd() {  
playin.set(false);

int dlrVal = deals.valProps().get();  
int plyrVal = player.valProps().get();  
String winning = "Exceptional case: d: " + dlrVal + " p: " + plyrVal;

if (dlrVal == 21 || plyrVal > 21 || dlrVal == plyrVal  
|| (dlrVal < 21 && dlrVal > plyrVal)) {  
winning = "DEALER";  
}  
else if (plyrVal == 21 || dlrVal > 21 || plyrVal > dlrVal) {  
winning = "PLAYER";  
}

mssg.setText(winning + " WON");  
}

@Override  
public void start(Stage primaryStage) throws Exception {  
primaryStage.setScene(new Scene(drawingHands()));  
primaryStage.setWidth(800);  
primaryStage.setHeight(600);  
primaryStage.setResizable(false);  
primaryStage.setTitle("BlackJack");  
primaryStage.show();  
}

public static void main(String[] args) {  
launch(args);  
}  
}

**Cards.java**

package com.almasb.blackjack;

import javafx.scene.Parent;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.StackPane;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Rectangle;  
import javafx.scene.text.Font;  
import javafx.scene.text.Text;

public class Cards extends Parent {

private static final int CRD\_WTH = 100;  
private static final int CRD\_HT = 140;

enum Suites {  
HEART, DIAMONDZ, CLUBZ, SPADEZZ;

final Image img;

Suites() {  
this.img = new Image(Cards.class.getResourceAsStream("images/".concat(name().toLowerCase()).concat(".png")),  
32, 32, true, true);  
}  
}

enum Grades {  
II(2), III(3), IV(4), V(5), VI(6), VII(7), VIII(8), IX(9), X(10),  
JJJ(10), QQQ(10), KKK(10), AAA(11);

final int val;  
Grades(int val) {  
this.val = val;  
}

String printName() {  
return ordinal() < 9 ? String.valueOf(val) : name().substring(0, 1);  
}  
}

public final Suites stui;  
public final Grades grades;  
public final int val;

public Cards(Suites stui, Grades grades) {  
this.stui = stui;  
this.grades = grades;  
this.val = grades.val;

Rectangle bgrund = new Rectangle(CRD\_WTH, CRD\_HT);  
bgrund.setArcWidth(20);  
bgrund.setArcHeight(20);  
bgrund.setFill(Color.WHITE);

Text txt = new Text(grades.printName());  
txt.setFont(Font.font(18));  
txt.setX(CRD\_WTH - txt.getLayoutBounds().getWidth() - 10);  
txt.setY(txt.getLayoutBounds().getHeight());

Text txt1 = new Text(txt.getText());  
txt1.setFont(Font.font(18));  
txt1.setX(10);  
txt1.setY(CRD\_HT - 10);

ImageView vw = new ImageView(stui.img);  
vw.setRotate(180);  
vw.setX(CRD\_WTH - 32);  
vw.setY(CRD\_HT - 32);

getChildren().addAll(bgrund, new ImageView(stui.img), vw, txt, txt1);  
}

@Override  
public String toString() {  
return grades.toString() + " of " + stui.toString();  
}  
}

**Decks.java**

package com.almasb.blackjack;

import com.almasb.blackjack.Cards.Grades;  
import com.almasb.blackjack.Cards.Suites;

public class Decks {

private Cards[] crdzz = new Cards[52];

public Decks() {  
refill();  
}

public final void refill() {  
int uu = 0;  
for (Suites stui : Suites.values()) {  
for (Grades grades : Grades.values()) {  
crdzz[uu++] = new Cards(stui, grades);  
}  
}  
}

public Cards withdrawCrd() {  
Cards cdrr = null;  
while (cdrr == null) {  
int indices = (int)(Math.random()\*crdzz.length);  
cdrr = crdzz[indices];  
crdzz[indices] = null;  
}  
return cdrr;  
}  
}

**Hands.java**

package com.almasb.blackjack;

import javafx.beans.property.SimpleIntegerProperty;  
import javafx.collections.ObservableList;  
import javafx.scene.Node;

import com.almasb.blackjack.Cards.Grades;

public class Hands {

private ObservableList<Node> crdzz;  
private SimpleIntegerProperty val = new SimpleIntegerProperty(0);

private int azz = 0;

public Hands(ObservableList<Node> crdzz) {  
this.crdzz = crdzz;  
}

public void takeCard(Cards cdrr) {  
crdzz.add(cdrr);

if (cdrr.grades == Grades.AAA) {  
azz++;  
}

if (val.get() + cdrr.val > 21 && azz > 0) {  
val.set(val.get() + cdrr.val - 10); //then count ace as '1' not '11'  
azz--;  
}  
else {  
val.set(val.get() + cdrr.val);  
}  
}

public void newOne() {  
crdzz.clear();  
val.set(0);  
azz = 0;  
}

public SimpleIntegerProperty valProps() {  
return val;  
}  
}

**OUTPUT AS**