Izzy Biel

Period 8

**Elevens Lab Activity Questions**

***Activity 2***

1. Relationship between Deck and Card: The Deck class contains instances of individual card objects. While the card class contains specific instance variables of the rank, suit, and point value, the Deck class contains instance variables of the size of **how many cards** there are in the deck, which is referenced via List.
2. The initialized deck contains 3 cards because there are three different ranks and point values, with 2 different suits. This means that of the three cards, two of them have the same suit.
3. String[] ranks = {“2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “J”, “Q”, “K”, “A”};

String[] suits = {“Spades”, “Hearts”, “Diamonds”, “Clubs”};

int[] pointValues = {2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11};

1. The only thing that matters in terms of keeping elements in order is making sure that the ranks array and the pointValues array are parallel. If they are not, the card will not be assigned the correct pointValue according to the rank. Otherwise, the numbers do not need to be in order.

***Activity 3***

1. public static String flip()

{

Random generator = new Random();

int coin = generator.nextInt(3);

if( coin == 0 || coin == 1)

{

return “heads”;

}

return “tails”;

}

1. public static Boolean arePermutations(int[] array1, int[] array2)

{

Boolean[] return = new boolean[array1.length];

for( int i1 = 0; i1<array1.length; i1++ )

{

for( int i2 = 0; i2<array1.length; i2++ )

{

if( array2[i2] == array1[i1] )

{

return[i1] = true;

}

else

{

Return[i1] = false;

}

}

}

for(int i = 0; i<return.length; i++)

{

If( return[i] == false)

{

return false;

}

}

return true;

}

1. 0, 1, 1

***Activity 5***

**Buggy 1:**

* Method name: isEmpty()
* Error: The code checks the incorrect number for size – it checks for a value greater than zero (instead of zero)

**Buggy 2:**

* Method name: deal()
* Error: The code subtracts one too many times; it subtracts one from size inside the if loop, as well as once outside the loop.

**Buggy 3:**

* Method name: shuffle()
* Error: Shuffle method is dysfunctional because it doesn’t change the order of the cards – doesn’t change the value of values[]

**Buggy 4:**

* Method name: deal()
* Error: The deal method doesn’t check if the deck is empty – it just returns null.

**Buggy 5:**

* Method name: shuffle()
* Error: The code sets the size variable wrong. Instead of cards.size()-1, it just assigns the size variable cards.size() – creates an out of bounds error.

***Activity 6***

1. All possible plays:

5spades+6clubs or 5clubs + 6clubs

1. Yes – if the game is played correctly, there all pairs and triplets should be made. If there are three cards left and all other cards are used, those three cards must be a triplet of 11.
2. I don’t think it involves strategy – when there is more than one play possible, it doesn’t matter how the pairs are made because the amount of pairs that is made is still going to be the same. This game is based on chance.

***Activity 7***

1. Necessary items: deck, currentCards
   1. Private Deck deck;
   2. Private Card[] currentCards
2. Actions necessary for game:

* Deal out initial nine cards
* Make pairs until can’t anymore
* Win or game over

Public void playElevens()

deck.shuffle();

for(int i = 0; I < 10; i++)

deck.deal();

for(int I = 0; i<currentCards.length; i++)

int i1 = currentCards[i];

if( currentCards[i+1] + i1 == 11

|| // J + Q + K )

// make pair

// replace currentCards that made pair

if no more pairs available

you lose

else

you win

1. Yes – class contains necessary behaviors (except those methods that will be implemented in activity 9
2. **A.** dealMyCards() is inside newGame()

**B.** methods: isLegal(List<Integer> selectedCards) + anotherPlayIsPossible()

**C.** returned list: Jhearts, 6clubs, 2spades, Aspades, 4hearts

**D.** printCards method code:

For(int I = 0; I < cIndexes.size(); i++)

For(int j = 0; j <cards.size(); j++)

System.out.print(cards[i].toString();)

**E.** which method?

***Activity 8***

1. Similarities:

* \*state = deck of cards and cards on the board
* \*behavior = methods: dealing cards, removing/replacing selected cards, checking for win, check if satisfying rules, more selections?

Differences:

* pairs of cards add up to a different number (11, 13, 10)
* All have different set number of board cards (9, 10, 11)

1. ElevensBoard must inherit from Board. This means the instance variables are inherited as well
2. Abstract methods of Board.java:

***Activity 9***