BLACKJACK GAME PART 1 (Card , Deck and Dealer)

Цель этого задания - создать игру BLACKJACK, в которую можно будет играть в выходном контейнере (терминале) IDE.

Примечание. Вы можете изменить любые детали реализации, если программа будет работать в соответствии с требованиями.

1. Создайте класс с именем Card, у этого класса будут следующие поля: face тип String, suit тип String, cardValue тип int и IsVisible тип boolean.

1.1 Добавьте методы getter и setter для полей, упомянутых выше.

1.2 Добавить метод toString

1.3 Добавьте конструктор, содержащий все поля

2. Создайте класс под названием DeckFactory (этот класс представляет собой колоду игральных карт).

2.1 Добавьте следующие статические поля:suits типа String [] и faceValues для типа String[]

и инициируйте переменные с соответствующими значениями:

private static String [] suits = {"Diamonds", "Clubs", "Hearts", "Spades"};

private static String [] faceValues = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",

"Queen", "King", "Ace"}

2.2 Добавьте статический метод с именем generateRandomDeck (), который создает объекты Card на основе значений в приведенных выше статических массивах. Внутри метода создайте ArrayList типа Card и, используя 2 цикла for, заполните ArrayList. Карты внутри ArrayList следует перемешать, и только после этого должен быть возвращен ArrayList. Значение по умолчанию для поля isVisible = true.

Пример перемешивание ArrayList с именем myList:

Collections.shuffle (myList);

Метод:

public static ArrayList <Card> generateRandomDeck () {

}

2.3 Добавьте статическое поле под названием **deck** типа ArrayList <Cards>, инициализируйте это поле. (это колода карт)

2.4 Добавьте конструктор без формальных параметров, внутри конструктора назначьте полю **deck** вызов метода **generateRandomDeck()**

2.5 Добавить геттер / сеттер для поля deck

2.6 Добавить метод toString ()

3. Создайте класс под названием Dealer.

3.1 Добавьте следующие поля: deckFactory типа DeckFactory, name типа String, hand типа ArrayList <Card> (инициализируйте его), handValue типа int.

3.2 Добавьте конструктор с указанными выше полями

3.3 Добавить метод, имитирующий процесс извлечения карты из колоды.

```
public Card drawCard () {
    // ваш код здесь
}
```

В приведенном выше методе вы должны извлечь карту с последним индексом из ArrayLista<Card> deck объекта deckFactory.

Примечание: вы можете получить доступ к колоде карт в следующем образом:

deckFactory.getDeck ()

Затем удалите карту из ArrayList с помощью метода remove.

3.4 Добавьте метод, который будет рассчитывать количество баллов карт на руках у дилера:

Каждая карта имеет числовое значение, если это карта с номиналом валет, дама, король или туз, значение карты равно 10.

В противном случае используйте значение карты, написанное на ней, как 2,3,4,5,6,7,8,9,10.

```
public int calculateHandValue () {
     // присваиваем вычисленное значение полю handValue и затем возвращаем
    //  результат
}
```

3.5 Добавьте метод, который будет печатать карты, которые есть у дилера, и общее количество очков. Метод должен распечатать их в следующем формате:

[King of Hearts], [Two of Diamonds] очки: 12

Если карта для поля IsVisible установлено значение false, распечатайте его следующим образом:

[King of Hearts], [Hidden Card] очки: 10

```
public void printHand () {
// ваш код здесь
}
```

3.6 Добавьте метод под названием hit, метод вытянет карту с помощью метода drawCard и добавит ее в руку дилера.

```
public void hit (boolean isVisible) {
// ваш код здесь
}
```

Прежде чем карта будет добавлена в руку дилера (ArrayList), вы должны установить в поле isVisible карты значение параметра видимости (true/false).

4. Добавьте класс под названием BlackApp.

4.1 Добавить основной метод

4.2 Внутри основного метода создайте DeckFactory, создайте дилера, протестируйте методы printHand, calculateHandValue и hit.