AP Computer Science | Project 22 - Cards | Name:
Dr. Paul L. Bailey | Monday, November 2, 2017

We have developed several cards classes. The source code we developed is attached. Each class should, of course, reside in its own source file. Compare your source code to the attached code, and make sure yours has the same functionality.

Answer the following questions. Then, on the next page, code the indicated methods. After you have completely debugged your program, neatly write the source code in the space provided.

Problem 1. State what you understand about the difference between overloading and overriding. Give examples from the current project.

Problem 2. State what you understand about the difference between inheritance and composition.

Problem 3. State what you understand regarding Java enumerations.

Problem 4. State what you understand regarding Java interfaces.

Program 1. Create a method `public Hand deal(int n)` int the `Stack` class which deals and returns one hand containing n cards. The cards should come off the top (front) of the deck, and be removed from the deck once they are dealt. Write the source code below.

Write code below

Program 2. Create a method `public Hand[] deal(int n, int k)` int the `Stack` class which deals and returns an array containing k hands, where each hand contains n cards. The cards should come off the top (front) of the deck, and be dealt into the hands in standard card dealing order (one card to the first hand, the next card to the second hand, and so forth, until each hand contains n cards). The cards should be removed from the deck once they are dealt. Write the source code below.

Write code below

Detach the following code and turn in the previous sheet.

```
public enum Color
{
    Black,
    Red;
}

public enum Suit
{
    Clubs(Color.Black),
    Diamonds(Color.Red),
    Hearts(Color.Red),
    Spades(Color.Black);

    private Color color;

    private Suit(Color color)
    {
        this.color = color;
    }

    public Color getColor()
    {
        return color;
    }
}

public enum Rank
{
    Two,
    Three,
    Four,
    Five,
    Six,
    Seven,
    Eight,
    Nine,
    Ten,
    Jack,
    Queen,
    King,
    Ace;
}
```

```
public class Card implements Comparable<Card>
{
    private Rank rank;
    private Suit suit;

    public Card(Rank rank, Suit suit)
    {
        this.rank = rank;
        this.suit = suit;
    }

    public Card(Suit suit, Rank rank)
    {
        this.rank = rank;
        this.suit = suit;
    }

    public Color getColor()
    {
        return suit.getColor();
    }

    public String toString()
    {
        return rank + " of " + suit;
    }

    public boolean equals(Card that)
    {
        return this.rank == that.rank && this.suit == that.suit;
    }

    public int compareTo(Card that)
    {
        int c = this.suit.compareTo(that.suit);
        if (c == 0)
        {
            c = this.rank.compareTo(that.rank);
        }
        return c;
    }
}
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

public class Stack extends ArrayList<Card>
{
    Random random = new Random(0);

    public void print()
    {
        for (Card card : this)
        {
            System.out.printf("%-17s %s\n", card, card.getColor());
        }
    }

    public void shuffle()
    {
        for (int i = size() - 1; i > 0; i--)
        {
            int j = random.nextInt(i + 1);
            Card card = get(i);
            set(i, get(j));
            set(j, card);
        }
    }

    public void sort()
    {
        Collections.sort(this);
    }

    public Card deal()
    {
        Card card = get(0);
        remove(0);
        return card;
    }
}

public class Deck extends Stack
{
    public void build()
    {
        Suit[] suits = Suit.values();
        Rank[] ranks = Rank.values();
        clear();
        for (Suit suit : suits)
        {
            for (Rank rank : ranks)
            {
                Card card = new Card(suit, rank);
                add(card);
            }
        }
    }
}
```

```
public class Hand extends Stack
{ }

public class Program
{
    public static void main(String[] args)
    {
        testA();
    }

    public static void testA()
    {
        Deck deck = new Deck();
        deck.build();
        deck.shuffle();
        Hand[] hands = deck.deal(5, 3);

        for (Hand hand : hands)
        {
            hand.print();
            System.out.println();
        }
    }
}
```

If you have seeded your random number generator with 0, you can expect the following output from the `testA` method above.

```
Ten of Spades     Black
Queen of Hearts   Red
Ace of Spades     Black
Seven of Clubs    Black
Jack of Spades    Black

King of Diamonds  Red
Four of Diamonds  Red
Jack of Clubs     Black
Ten of Hearts     Red
Queen of Clubs    Black

Ace of Diamonds   Red
Six of Spades     Black
Queen of Diamonds Red
Seven of Diamonds Red
Seven of Hearts   Red
```