Chap12Yourname.java

Arrays of objects

```
/*
 * Chap 12 Arrays of objects
 * Card.java
 */

import java.io.PrintStream;

public class Chap12 {

	public static void main(String[] args) {
		PrintStream out = System.out;
```

Example set of 52 playing cards; 13 of each suit clubs, diamonds, hearts, and spades

	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
Clubs													
Diamonds													
Hearts													
Spades													

12.1, 12.2, 12.4 Creating the Card class

```
out.println("12.1, 12.2, 12.4 Creating the Card class");
//Create the Card class.
//Mapping for suits: Clubs = 0, Diamonds = 1, Hearts = 2, Spades = 3;
```

12.1, 12.2, 12.4 Creating the Card class

```
//Declaring the instance variables:
private int rank;
private int suit;
//Mapping for suits: Clubs = 0, Diamonds = 1, Hearts = 2, Spades = 3;
//Mapping for ranks: Ace = 1, Jack = 11, Queen = 12, King = 13;
//We declare rank and suit as int to compare cards which has a higher rank or suit.
```

12.1, 12.2, 12.4 Creating the Card class

```
//Value constructor:
public Card(int rank, int suit) {


}
```

12.1, 12.2, 12.4 Creating the Card class

```
//Value constructor:
public Card(int rank, int suit) {
    this.rank = rank;
    this.suit = suit;
}
```

12.1, 12.2, 12.4 Creating the Card class

```
//Create card objects: Ace of Hearts and 2 Diamonds.
Card aceHearts = new Card(      );
Card twoDiamonds = new Card(      );
```

12.1, 12.2, 12.4 Creating the Card class

```
//Create card objects: Ace of Hearts and 2 Diamonds.
Card aceHearts = new Card(1, 2);
Card twoDiamonds = new Card(2, 1);
```

12.1, 12.2, 12.4 Creating the Card class

```
//Create the toString method using arrays for Card class so that the result of
//System.out.println(aceHearts);
//is Ace of Hearts and the result of
//System.out.println(twoDiamonds);
//is 2 of Diamonds.
out.println(aceHearts);
out.println(twoDiamonds);
```

12.1, 12.2, 12.4 Creating the Card class

```
public String toString() {



}
```

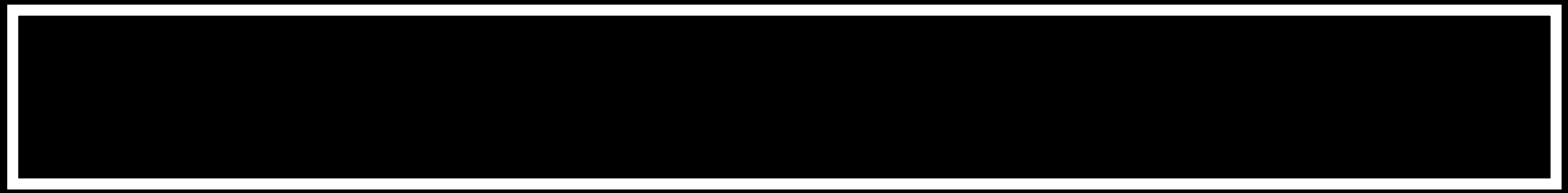
12.1, 12.2, 12.4 Creating the Card class

```
public String toString() {
    String[] ranks = {null, "Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10",
                "Jack", "Queen","King"};
    //null indicates an unused element.
    String[] suits = {"Clubs", "Diamonds", "Hearts", "Spades"};
    return ranks[this.rank] + " of " + suits[this.suit];
}
```

12.1, 12.2, 12.4 Creating the Card class

```
//Create the equals method.
out.println(aceHearts.equals(twoDiamonds));
```

12.1, 12.2, 12.4 Creating the Card class



12.1, 12.2, 12.4 Creating the Card class

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

12.1, 12.2, 12.4 Creating the Card class

```
//Create the compareTo method.
//Comparison of suits comes first: Clubs < Diamonds < Hearts < Spades.
out.println(aceHearts.compareTo(twoDiamonds));
```

12.1, 12.2, 12.4 Creating the Card class

12.1, 12.2, 12.4 Creating the Card class

```
public int compareTo(Card that) {
    if (this.suit < that.suit) {
        return -1;
    }
    if (this.suit > that.suit) {
        return 1;
    }
    if (this.rank < that.rank) {
        return -1;
    }
    if (this.rank > that.rank) {
        return 1;
    }
    return 0;
}
```

12.1, 12.2, 12.4 Creating the Card class

```
//Create the getters.
out.println(aceHearts.getRank());
out.println(aceHearts.getSuit());
out.println();
```

12.1, 12.2, 12.4 Creating the Card class

```
public int getRank() {

}

public int getSuit() {

}
```

12.1, 12.2, 12.4 Creating the Card class

```
public int getRank() {
    return this.rank;
}

public int getSuit() {
    return this.suit;
}
```

12.3 Class variables

```
out.println("12.3 Class variables");
//Class variables are defined in a class, before the method definitions.
//They are public (we can access them by calling Card.RANKS),
//static (shared across all instances of the class ),
//and final (can not be reassigned).
//Declare class variables String[] RANKS and String[] SUITS after the instance
//variables of the Card class.
out.println();
```

12.3 Class variables

```
public static final String[] RANKS = {null, "Ace", "2", "3", "4", "5", "6", "7",
          "8", "9", "10",   "Jack", "Queen","King"};
public static final String[] SUITS = {"Clubs", "Diamonds", "Hearts", "Spades"};
//In this class, RANKS and SUITS are mostly used in the toString method.
//Having them defined as class variables, we need not to create and
//garbage-collect them every time when toString is invoked.
```

12.3 Class variables

```
public String toString() {
    return RANKS[this.rank] + " of " + SUITS[this.suit];
}
```

12.5 final instance variables

```
out.println("12.5 final instance variables");
//We do not want to modify any existing card, so we make the Card class
//immutable.
//Declare the instance variables final, so that no other programmers could
//add a modifier later. Once the instance variables of an instance object is
//assigned, they can never be reassigned.
out.println();
```

12.5 final instance variables

```
private final int rank;
private final int suit;
```

12.6 Array of objects

```
out.println("12.6 Array of objects");
Card[] cards = new Card[52];
//Creation of references to 52 Card objects, not creation of 52 Card objects.
//The elements are initialized to null.
//Write a method anyCard(Card[] cards) to check if all elements of cards are
//null.
anyCard(cards);
```

12.6 Array of objects

```
public static void anyCard(Card[] cards) {



}
```

12.6 Array of objects

```
public static void anyCard(Card[] cards) {
        boolean check = false;
        for (int i = 0; i < cards.length; i++) {
                check = check || cards[i] != null;
        }
        if (check == false) {
                System.out.println("No card yet.");
        } else {
                System.out.println("There is at least one card.");
        }
}
```

12.6 Array of objects

//Now we populate the array.

12.6 Array of objects

```
//Now we populate the array.
int i = 0;
for (int suit = 0; suit <= 3; suit++) {
        for (int rank = 1; rank <= 13; rank++) {
                cards[i] = new Card(rank, suit);
                i++;
        }
}
```

12.6 Array of objects

```
//Display the 52-card deck.
printDeck(cards);
out.println();
```

12.6 Array of objects

```
public static void printDeck(Card[] cards) {



}
```

12.6 Array of objects

```
public static void printDeck(Card[] cards) {
        for (int i = 0; i < cards.length; i++) {
                System.out.println(cards[i]);
        }
}
```

12.7 - 12.10 Binary search

```
out.println("12.7 - 12.10 Binary search");
//Write a search(Card[] cards, Card target) method to search for an object
//in the array of objects.
out.println(search(cards, twoDiamonds));
```

12.7 - 12.10 Binary search

```
public static int search(Card[] cards, Card target) {



}
```

12.7 - 12.10 Binary search

```
public static int search(Card[] cards, Card target) {
        for (int i = 0; i < cards.length; i++) {
                if (cards[i] == target) {
                        return i;
                }
        }
        return -1;
}
```

12.7 - 12.10 Binary search

```
public static int search(Card[] cards, Card target) {
        for (int i = 0; i < cards.length; i++) {
                if (cards[i].equals(target)) {
                        return i;
                }
        }
        return -1;
}
```

12.7 - 12.10 Binary search

```
//If the cards are in order, we can use a better algorithm: Binary search.
//1. Choose an index between low and high and call it mid. Compare the card
//        at mid to the target.
//2. If they are equal, return the index.
//3. If the card at mid is lower than the target, search the range from
//        mid + 1 to high.
//4. If the card at mid is higher than the target, search the range from
//        low to mid - 1.
//Write the biSearch(Card[] cards, Card target) method.
```

12.7 - 12.10 Binary search

```
public static int biSearch(Card[] cards, Card target) {
```

12.7 - 12.10 Binary search

```
public static int biSearch(Card[] cards, Card target) {
        int low = 0;
        int high = cards.length - 1;
        while (low <= high) {
                int mid = (low + high) / 2;
                int diff = cards[mid].compareTo(target);
```

12.7 - 12.10 Binary search

```
            if (diff == 0) {
                return mid;
            }
            if (diff < 0) {
                low = mid + 1;
            }
            if (diff > 0) {
                high = mid - 1;
            }
        }
        return -1;
    }
```

12.7 - 12.10 Binary search

```
out.println(biSearch(cards, twoDiamonds));
//If the array contains n elements, binary search requires log2n comparisons,
//and sequential search requires n. For large value of n, binary search can
//be much faster.
```

12.7 - 12.10 Binary search

//If the method you just wrote is iterative, write the recursive version.

//Vice versa.