According Wikipedia Contract Bridge is a trick-taking game using a standard 52-card deck.

Each card consists of two characters. The first represents the value of the card: “A” (ace), “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “J” (Jack), “Q” (Queen), “K” (King). The second represents the suit of the card: “S”, “H”, “D”, “C”, standing for “spades”, “hearts”, “diamonds”, and “clubs”, respectively.

In Contract Bridge, players first bid against each other; bids consist of a suit (clubs, diamonds, hearts, or spades) and a number of tricks they expect to take beyond the first six. First, though, they must assess the strength of the 13 cards in their hands to determine a reasonable bid. You are to write a program that performs a simple version of this assessment (for the first bidder only) and decides on an opening suit. To determine the opening suit, the point value of the hand must be computed.

You have been given the Card class which represents a card from a standard 52-card deck which is shown below.

public class Card {

private String value;

private String suit;

/\*

\* value.length() == 2 iff value.equals("10")

\* otherwise value.length() == 1

\*

\* value will always contain upper case letters (A (ace), K (king) Q (queen)

\* J (jack) or numbers 2, ... 9, 10

\*

\* suit will be S (spade), D (diamond), H (heart) or C (club).

\* - Always a single upper case letter

\*/

public Card(String v, String s) {

value = v;

suit = s;

}

public String getValue() { return value; }

public String getSuit() { return suit; }

public int hashCode() {

return getValue().hashCode() + getSuit().hashCode();

}

public boolean equals(Object obj) {

Card temp = (Card)obj;

return getValue().equals(temp.getValue()) && getSuit().equals(temp.getSuit());

}

}

You are to complete the getPoints and the getSuit methods in the ContractBridge class. The ContractBridge class has a List<Card> hand as its only instance variable. This instance variable is assigned in the only constructor provided in the class. The instance variable hand holds the 13 cards that make up the hand.

The getPoints method has no parameters and returns an int. The getPoints method returns the number of points the hand is worth according to the following 7 rules:

1. Each ace counts 4 points. Each king counts 3 points. Each queen counts 2 points. Each jack counts one point.
2. Subtract a point for any king in a suit with only one card (that is, where the king is the only card of that suit in your hand).
3. Subtract a point for any queen in a suit with no more than two cards.
4. Subtract a point for any jack in a suit with no more than three cards.
5. Add a point for each suit of which the hand contains exactly two cards.
6. Add two points for each suit of which the hand contains exactly one card.
7. Add three points for each suit of which the hand contains no cards.

In writing the getPoints method, you may assume:

* hand.size() == 13

The following code shows the results of the getPoints method.

|  |  |
| --- | --- |
| The following code | Returns |
| ArrayList<Card> hand = new ArrayList<Card>();  hand.add( new Card("K", "S"));  hand.add( new Card("Q", "S"));  hand.add( new Card("10", "H"));  hand.add( new Card("8", "H"));  hand.add( new Card("4", "H"));  hand.add( new Card("A", "C"));  hand.add( new Card("Q", "C"));  hand.add( new Card("10", "C"));  hand.add( new Card("5", "C"));  hand.add( new Card("K", "D"));  hand.add( new Card("Q", "D"));  hand.add( new Card("J", "D"));  hand.add( new Card("8", "D"));  ContractBridge cb = new ContractBridge(hand); |  |
| cb.getPoints(); | 17 |

The getSuit method has no parameters and returns a String. The getSuit method returns the opening suit which is determined according to the following criteria:

A suit is stopped if it contains an ace, or if it contains a king and at least one other card, or if it contains a queen and at least two other cards.

During the opening assessment, the three most common possibilities are:

* If the hand evaluates to fewer than 14 points, then the player must pass.
* One may open bidding in a suit if the hand evaluates to 14 or more points. Bidding is always opened in one of the suits with the most cards.
* One may open bidding in “no-trump” if the hand evaluates to 16 or more points ignoring rules 5, 6, and 7, and if **all four suits are stopped**. A no-trump bid is always preferred over a suit bid when both are possible.

For the first example below, the evaluation starts with 6 points for the two kings, 4 for the ace, 6 for the three queens, and one for the jack. To this tally of 17 points, we add 1 point for having only two cards in spades, and subtract 1 for having a queen in spades with only one other card in spades. The resulting 17 points is enough to justify opening in a suit.

The evaluation for no-trump is 16 points, since we cannot count the one point for having only two spades. We cannot open in no-trump, however, because the hearts suit is not stopped (note: since hearts is not stopped, clubs is a valid suit though it qualifies as a stopped suit). Hence we must open bidding in a suit. The two longest suits are clubs and diamonds, with four cards each, so the possible choices are to bid clubs or bid diamonds.

For this problem, we will prefer diamonds to clubs (and hearts to diamonds and spades to hearts).

Turn page to see sample code using the getSuit method.

The following code shows the results of the getSuit method.

|  |  |
| --- | --- |
| The following code | Returns |
| ArrayList<Card> hand = new ArrayList<Card>();  hand.add( new Card("K", "S"));  hand.add( new Card("Q", "S"));  hand.add( new Card("10", "H"));  hand.add( new Card("8", "H"));  hand.add( new Card("4", "H"));  hand.add( new Card("A", "C"));  hand.add( new Card("Q", "C"));  hand.add( new Card("10", "C"));  hand.add( new Card("5", "C"));  hand.add( new Card("K", "D"));  hand.add( new Card("Q", "D"));  hand.add( new Card("J", "D"));  hand.add( new Card("8", "D"));  ContractBridge cb = new ContractBridge(hand); |  |
| cb.getSuit(); | “D” |