Objectives
To master programming with enum and array types.

Overview
The PlayingCard class represents a common playing-card. In this implementation, the
suit and rank of a playing-card are represented by enum types, CardSuit and CardRank.
You will implement 2 additional classes:
• PlayingCardDeck represents a deck of 52 PlayingCards in 4 suits of 13 ranks each.
The representation of PlayingCardDeck is non-traditional. Instead of as an array of 52
PlayingCard elements, it is an array of 52 boolean elements. Each element corresponds
to one of the 52 playing-cards: true if that card is in the deck, false if that card is not in
the deck. Each group of 13 consecutive elements represents the cards of one suit:

<table>
<thead>
<tr>
<th>0</th>
<th>12</th>
<th>13</th>
<th>25</th>
<th>26</th>
<th>38</th>
<th>39</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUBS</td>
<td>DIAMONDS</td>
<td>HEARTS</td>
<td>SPADES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Within each suit, the array elements represent the playing-cards in rank order:

|   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |   T   |   J   |   Q   |   K   |   A   |

• PokerHand represents a poker hand of 5 PlayingCards
The PokerHand class is implemented using an array of 5 PlayingCard elements dealt
from a PlayingCardDeck.

Specific Requirements
1. The PlayingCard class is already implemented. A client to test it is provided. Run the
client and study the PlayingCard code until you understand the implementation.
2. Complete the implementation of the PlayingCardDeck class. An outline with stubs of
all the required methods is provided. Your implementation must
✓ use the array representation described above,
✓ use / and % to map an array index to CardSuit and CardRank ordinals,
✓ use the ordinals to select from CardSuit.values() and CardRank.values() arrays.
A client to test your implementation is provided.
3. Complete the implementation of the PokerHand class. A class outline is provided.
✓ Your addCard() method must insert a PlayingCard being added to a PokerHand
to maintain the PokerHand cards in sorted order.
✓ Your type() method’s algorithm must exploit the sorted order of a PokerHand.
✓ Your type() method must use helper method(s) for each hand-type being tested.
A client to test your implementation is provided.
4. Document your program
✓ Include a Program Id Paragraph into both source files
✓ Provide helpful comments

Submitting your Assignment
Upload your source (.java) files in SCIS Moodle by the due date. No late submissions.