AP CSA Short Term Project 1 (Design a Card Class)

Description

Students will work in small groups to produce a Java Class that defines a playing card.

Standards

IT-PGA-2 Describe the software application life cycle and use a prototype development model to develop applications.
IT-PGA-4 Design, develop, and implement accessible and usable interfaces, and analyze applications for engaging the user.

Business Ethics
Students will model work readiness traits required for success in the workplace including teamwork, multitasking, integrity, honesty, accountability, punctuality, time management, and respect for diversity.

Expectations

Students are expected to use the skills and concepts learned in the course to design a working card class.

Objectives

Students will design a class that represents a playing card.

Students will implement the class by creating an object of the class and test all parts.

The class should include private attributes for suit, rank, and pointValue.

The class should include methods to return private attributes, to compare one card to another, and to output the attributes of a card object as a String.

Project Time

The project will take approximately 3 hours to complete.

Rubric

50 points	Class is complete with all the features listed in objectives
25 points	Class and tester are written to industry accepted syntax
25 points	There are no errors when the code is compiled

Well Done! 100 points.

50 points	Class is complete with all the features listed in objectives - 50
25 points	Class and tester are written to industry accepted syntax - 25
25 points	There are no errors when the code is compiled - 25

```
/**
 * Card.java
 *
 * <code>Card</code> represents a playing card.
 */
public class Card {

	/**
	 * String value that holds the suit of the card
	 */
	private String suit;

	/**
	 * String value that holds the rank of the card
	 */
	private String rank;

	/**
	 * int value that holds the point value.
	 */
	private int pointValue;
```

```
/**
     * Creates a new <code>Card</code> instance.
     *
     * @param cardRank  a <code>String</code> value
     *                  containing the rank of the card
     * @param cardSuit  a <code>String</code> value
     *                  containing the suit of the card
     * @param cardPointValue an <code>int</code> value
     *                  containing the point value of the card
     */
    public Card(String cardRank, String cardSuit, int cardPointValue) {
     this.suit= cardSuit;
     this.rank=cardRank;
     this.pointValue=cardPointValue;
            /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
    }



/**
     * Accesses this <code>Card's</code> suit.
     * @return this <code>Card's</code> suit.
     */

    public String suit() {
     return suit;
            /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
```

```
}

        /**
         * Accesses this <code>Card's</code> rank.
         * @return this <code>Card's</code> rank.
         */
        public String rank() {
          return rank;
                /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
        }

/**
         * Accesses this <code>Card's</code> point value.
         * @return this <code>Card's</code> point value.
         */
        public int pointValue() {
          return pointValue;
                /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
        }

        /** Compare this card with the argument.
         * @param otherCard the other card to compare to this
         * @return true if the rank, suit, and point value of this card
         *            are equal to those of the argument;
         *        false otherwise.
         */
        public boolean matches(Card otherCard) {
```

```
 if ((otherCard.rank().equals(this.rank)) && (otherCard.suit().equals(this.suit))&&
 (otherCard.pointValue()==(this.pointValue)))

   return true;

 else

  return false;

          /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
}

/**
 * Converts the rank, suit, and point value into a string in the format
 *     "[Rank] of [Suit] (point value = [PointValue])".
 * This provides a useful way of printing the contents
 * of a <code>Deck</code> in an easily readable format or performing
 * other similar functions.
 *
 * @return a <code>String</code> containing the rank, suit,
 *         and point value of the card.
 */
@Override
public String toString() {
  return this.rank + " of " + this.suit+ " (point value = " +this.pointValue+ ")";
          /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
```

```
        }
}

/**
 * This is a class that tests the Card class.
 */
public class CardTester {

        /**
         * The main method in this class checks the Card operations for consistency.
         *      @param args is not used.
         */
        public static void main(String[] args) {
          Card card1= new Card ("Queen","clubs",12);

          Card card2 = new Card ("Jack","Hearts",11);

          Card card3 = new Card ("King","Diamonds",13);

                /* *** TO BE IMPLEMENTED IN ACTIVITY 1 *** */
                System.out.println("rank: " + card1.rank());
                System.out.println("suit: "+ card1.suit());
                System.out.println("pointvalue: "+ card1.pointValue());
                System.out.println("tostring: "+ card1.toString());
        System.out.println();

                System.out.println("rank: " + card2.rank());
```

```
            System.out.println("suit: "+ card2.suit());
            System.out.println("pointvalue: "+ card2.pointValue());
            System.out.println("tostring: "+ card2.toString());
        System.out.println();

            System.out.println("rank: " + card3.rank());
            System.out.println("suit: "+ card3.suit());
            System.out.println("pointvalue: "+ card3.pointValue());
            System.out.println("tostring: "+ card3.toString());

            System.out.println ("Card1 vs Card2 " + card1.matches (card2));
            System.out.println("Card2 vs Card3 " + card2.matches (card3));
            System.out.println("Card1 vs Card1 " + card1.matches (card1));

    }
}
```