AP Computer Science A Name_____
Lab 04C date _____ / _____

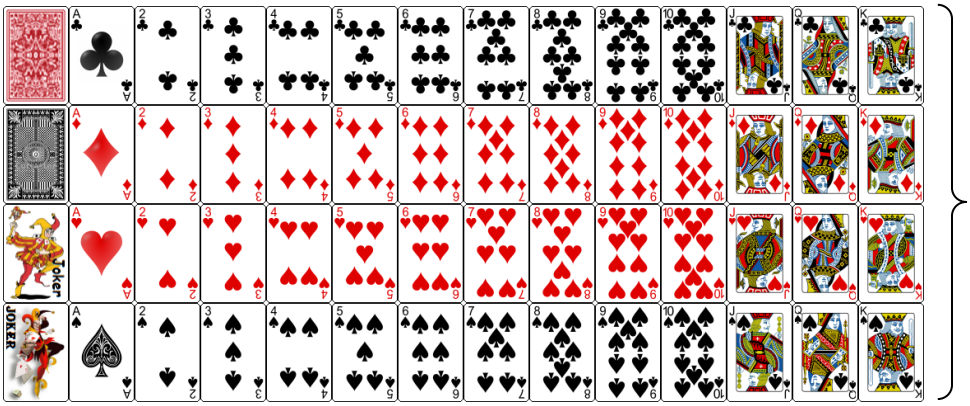
In this lab you will complete the **Card** class which models ordinary playing cards. The **Card** class is used in **Lab04C.java** (and also in a game that you may write in chapter 6).

1. Open the **Lab04C** project in the Teams section of **replit**. When you press the Run ▸ button you should see the app shown below. The coordinates in the upper left corner of the app follow your mouse, and coordinates are recorded as you click the mouse.

2. Open the **Card.java** file. Near the bottom of that file you will see three **private static** variables (and some initialization code).

   The keyword **static** indicates that no matter how many cards your app ever creates, there will only be one (shared) **CARD_WIDTH** variable. This is because all the cards have the same width, so there's no point in making each card have their *own* width instance variable. The **static** variables **CARD_WIDTH** and **CARD_HEIGHT** are **public** so that other classes may access these values, and that they are **final** so that no other class (including the **Card** class!) is allowed to modify their values.

   There is also a **private** (other classes will not be able to access this variable) **static** (shared by all future **Card** instances) **final** (cannot be reassigned) image named **ALL_CARDS**. This image is loaded from a file named **cards.png** before any cards are created and stores the faces (and backs) of all the cards as shown below.



Width: (14 × CARD_WIDTH)
Height: (4 × CARD_HEIGHT)

   Each card object that will be instantiated in the future will have a suit (a **String**) and a numeric value (an **int**). ***Declare*** these two instance variables in the **Card.java** class. Note that these are ***not*** **static** variables because each card will need ***its own copy*** of those variables.

3. In this step you will write two *different* **Card** class constructors:
   A) Code that I wrote in **MainWindow.java** constructs four different cards. The first explicit parameter is a string that will be the suit: **"c"**, **"d"**, **"h"**, or **"s"**. The second parameter is the value of the card (ace = 1, … , jack = 11, queen = 12, king = 13). Write the **Card** constructor to accept these explicit parameters. Be sure to correctly initialize the numeric and suit instance variables.

   B) Code that I wrote in **MainWindow.java** constructs a *joker* by calling a second **Card** constructor with no explicit parameters. Write a second constructor (with no explicit parameters). The numeric value should be 0, and the suit of a *joker* is **"j"**.

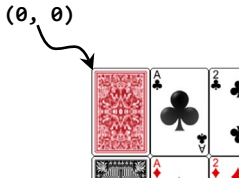
C) When you run the app make sure you ***do not*** see the error message shown below!

```
<<java.lang.NoSuchMethodException>>
    Card.java needs this constructor: public Card(String, int)

<<java.lang.NoSuchMethodException>>
    Card.java needs this constructor: public Card()
```

4. In this step you will write the **`public Image getImage()`** *accessor* method. This method will return an **`Image`** that is a sub-image from the **`static`** variable **`ALL_CARDS`**. Note that (1) all the clubs are in the first row, all the diamonds are in the second row, etc., and (2) the cards are consistently laid out in increasing numeric value.

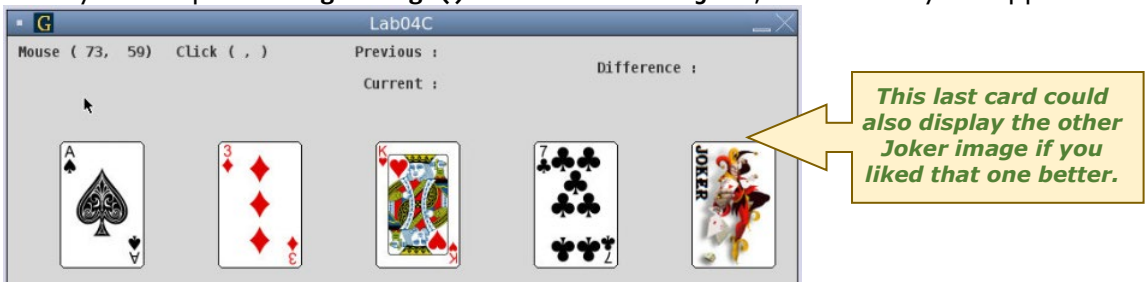
   To return a sub-image you will use **`return ALL_CARDS.getSubimage(x, y, w, h);`**, where the *x* and *y* are the coordinates of the *upper left corner* of the card within the **`ALL_CARDS`** image, and the *w* and *h* are the width and height of a card.



   Example: The 8 of heart's *upper left corner* is located at (**`8 * CARD_WIDTH, 2 * CARD_HEIGHT`**).

   Example: The joker's *upper left corner* is located at (**`0 * CARD_WIDTH, 2 * CARD_HEIGHT`**). You could have also used (**`0 * CARD_WIDTH, 3 * CARD_HEIGHT`**) if you liked that image more.

   After you complete the **`getImage()`** method in **`Card.java`**, make sure your app looks like this:

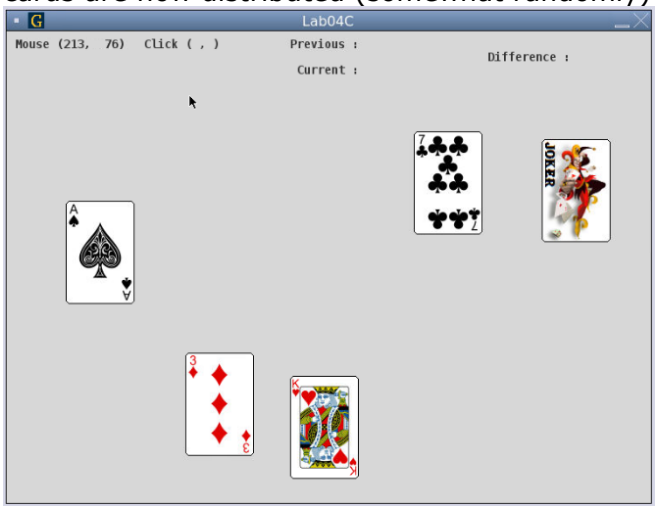


This last card could also display the other Joker image if you liked that one better.

5. Each card will need to keep track of its location on the **`Lab04C`** app: this location is ***not*** the (*x*, *y*) that you used in step #4. ***Declare*** two more integer instance variables (for the *x*-value and *y*-value) in the **`Card.java`** class. Initialize the instance variables for the *x*-value and *y*-value to 0 in both constructors.

6. Write the **`Card`** class *mutator* method **`public void setXY(int x, int y)`**. This method does not need to return a value, but it will change the instance variables for the card's upper left corner to **`x`** and **`y`**.

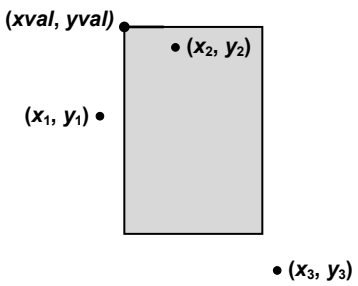
7. Write the **`Card`** class **`getX()`** and **`getY()`** *accessor* methods. When you run the app you should see that cards are now distributed (somewhat randomly) over the app screen.



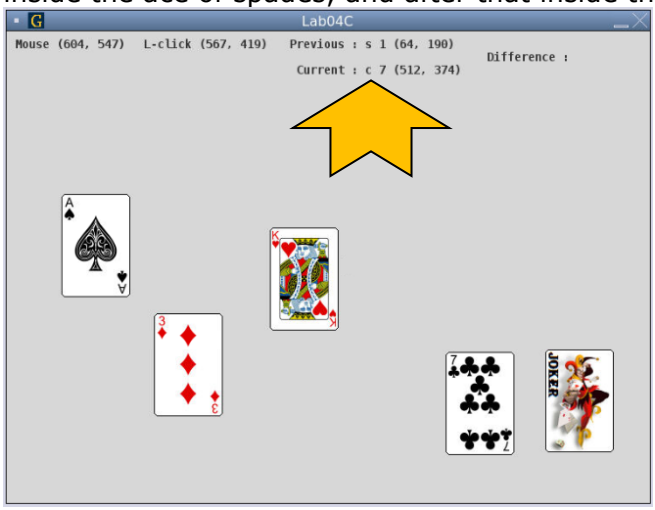
8. Write the **`Card`** class **`getValue()`** and **`toString()`** *accessor* methods. The **`toString()`** method should return a string in the following format: **`s 10 (250, 50)`**.

9. Write the *accessor* method **`contains(int x, int y)`**. This method will return either the value **`true`** or the value **`false`** depending on whether the explicit parameter variables (x, y) lie within the card's boundaries.

   The example shown at right shows three different points. The **`contains`** method should return **`false`** for (x_1, y_1) and (x_3, y_3), and should return **`true`** for (x_2, y_2). The ***xval*** and ***yval*** in the diagram refer to the card's instance value coordinates (from steps 5 – 7).



   Test your **`contains`** method by clicking in the app. Whenever you click inside a card, the **Current:** label should change to show which card you clicked in. In the example shown below I first clicked inside the ace of spades, and after that inside the seven of clubs.



10. Write the *accessor* method **`valueDifference(Card c)`**. This method will return an integer that follows these rules:

    - The value difference between any card and a Joker (or from a Joker to any card) is 1
    - The value difference between a king and an ace (or from an ace to a king) is 1
    - Otherwise, the difference between cards is the absolute value of the difference of their values.

    Run the app and click inside the cards to make sure the difference between cards follows the rules stated above. Specifically check that ace to king (and king to ace) is **one**, joker to anything (including itself!) is **one**, and that the difference between other pairs of cards is correct.

11. Test your app by **right**-clicking in the app window. Code that I wrote in **`MainWindow.java`** will place a random card at the location where you right-clicked. Then click within the cards to make sure that the **`Card.java`** methods you wrote still work correctly.

12. Once you have finished, click the ✓Submit button near the top right of your browser. This notifies me that you think your lab assignment is complete and ready for grading.