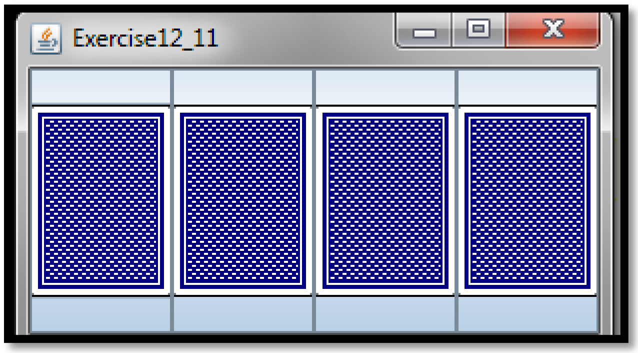
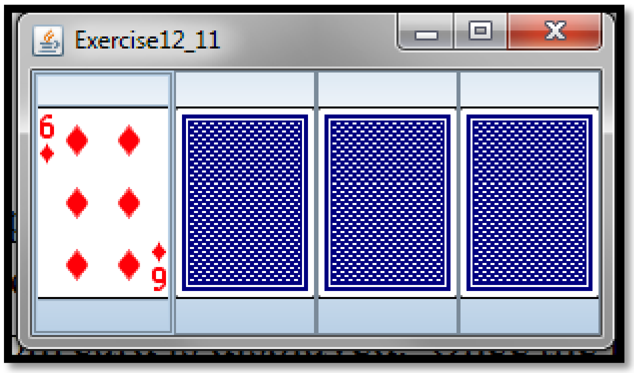
CIT 249: Java II
Chapter 12
Lab3

In this lab we will complete #11 on page 477. When the program is run it will display the backs of four cards. When the mouse passes over the first and third card a random card is displayed. Once you mouse is not over the card the original back of the card displays. When you press on the second or fourth card a random card is displayed. Once the mouse button is released it reverts back to the original card back.





1. Open a new document and save the file as Ch12Lab3.java.
2. Our first step is to import the predefined classes that are required by typing:

```
import java.awt.GridLayout;
import javax.swing.JFrame;
import javax.swing.ImageIcon;
import javax.swing.JButton;
```

3. Next we type our class header and opening brace:

```
public class Ch12Lab3 extends JFrame
{
```

   - Here we extend the JFrame. This predefine class contains all attributes and methods required to create standalone GUI applications.

4. When you extend the JFrame you normally include a constructor method for constructing the components that are added to the frame. The constructor method always has the same name as the class. Type:

```
public Ch12Lab3()
{
```

5. We set the frame's layout to a GridLayout with 1 rows and 4 columns. Type:

```
    setLayout(new GridLayout(1, 4));
```

6. We create an array with a limit of 54, which is for the 52 cards, the joker and the back of the cards. Type:

```
    int[] list = new int[54];
```

7. We load the list with 54 numbers, starting with 1. Type:

```
for (int i = 0; i < list.length; i++)
    list[i] = i + 1;
```

8. We shuffle the list, by invoking the shuffle() method passing the array to it. Type:

```
shuffle(list);
```

9. We construct a new ImageIcon, and four buttons, setting each button with the card as its image. Type:

```
ImageIcon backCover = new ImageIcon("image/card/backCard.png");
JButton jbt1 = new JButton(backCover);
JButton jbt2 = new JButton(backCover);
JButton jbt3 = new JButton(backCover);
JButton jbt4 = new JButton(backCover);
```

10. We add each button to the frame by typing:

```
add(jbt1);
add(jbt2);
add(jbt3);
add(jbt4);
```

11. We use a couple of JButton methods to change the button image based on whether the button is pressed or the cursor passes over the button. Type:

```
jbt1.setRolloverIcon(new ImageIcon("image/card/" + list[0] + ".png"));
jbt2.setPressedIcon(new ImageIcon("image/card/" + list[1] + ".png"));
 jbt3.setRolloverIcon(new ImageIcon("image/card/" + list[2] + ".png"));
jbt4.setPressedIcon(new ImageIcon("image/card/" + list[3] + ".png"));
```

- The setRolloverIcon() method changes the button image when the cursor passes over the button. Once the cursor is not over the button the image changes back to the back cover.
- The setPressedIcon() method changes the button image as long as the button is pressed. Once the left mouse button is released the image changes back to the back cover.

12. Close the constructor method.

13. The shuffle method will randomly shuffle the list. Type:

```
public static void shuffle(int[] list)
{
    for (int i = 0; i < list.length; i++)
    {
        // Generate an index randomly
        int index = (int)(Math.random() * list.length);

        // Swap myList[i] with myList[index]
        int temp =list[i];
        list[i] = list[index];
        list[index] = temp;
    }
}
```

- A for loop is used to go through the list.
- An integer is created and set to a random number times the array length
- Another integer is set to the current list element.
- The current list element is set to the random number.
- The list element with an index number of the random number is set with a value of the temp integer.

14. Close the method.
15. The main method is much the same as with the other two labs. Type:

```
public static void main(String[] args)
{
    Ch12Lab3 frame = new Ch12Lab3();
    frame.setTitle("Chapter 12 Lab3");
    frame.setSize(300, 170);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setVisible(true);
}
```

16. End by closing the class.
17. Compile the program and fix any errors if necessary.
18. Run the program.

19. Compress all files into a single zip or rar file and submit. Include the image folder within the zip or rar file.