COSC2436 Programming Fundamentals III /ITSE2445 Data Structures

The submission opens two days before the due day, and closes right at the due time on due day. One submission is allowed per project. It is students' responsibility to submit timely and correctly. Students MUST start each project when it is first discussed in class in order to complete it on time. All projects are individual projects. Project due days may be changed to better align with lectures.

If one of the following is true, not credits will be given:

- The project is late.
- The algorithm or class design is missing.
- The project has errors.
- There are no comments (Javadoc format required) at all in your code.
- Wrong files are submitted.
- **A project is a copy and modification of another student's project. (Both will receive 0.)**

Files to be submitted through Canvas:

- The UML design (UML class diagram)
- All source codes and driver programs
- All supporting files if any

Software Development Project Rubric: (Design is worth 20%; the rest is worth 80 %.)
Analysis: There will be no credit if the software doesn't meet customer specification at all.

- Does the software meet the exact customer specification?
- Does the software read the exact input data and display the exact output data as they are shown in sample runs?
- Does each class include all corresponding data and functionalities?

Design: There will be no credit if the design is missing.

- Is the design (a UML class diagram) an efficient solution?
- Is the design created correctly?

Code: There will be no credit if there are errors.

- Are there errors in the software?
- Are code conventions and name conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?

Debug:

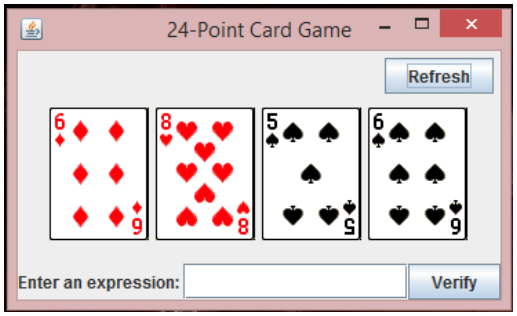
- Are there bugs in the software?

Documentation: There will be no credit if comments are not included.

- Are there enough comments included in the software?
- Class comments must be included before a class header.
- Method comments must be included before a method header.
- More comments must be included inside each method.
- All comments must be written in Javadoc format.

Project 3 The 24-point card game **Due: See calendar**

The 24-point card game is to pick any 4 cards from 52 cards, as shown in the figure below. Note that the Jokers are excluded. Each card represents a number. An Ace, King, Queen, and Jack represent **1**, **13**, **12**, and **11**, respectively. You can click the *Refresh* button to get four new cards.



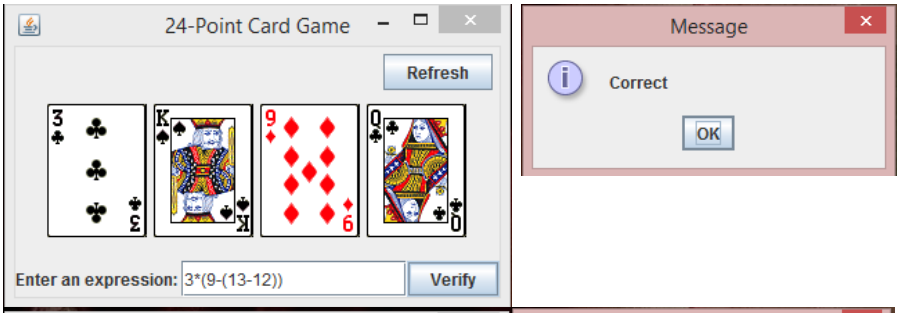
Enter an expression that uses the four numbers from the four selected cards. Each number must be used once and only once. You can use the operators (addition, subtraction, multiplication, and division) and parentheses in the expression. The expression must evaluate to **24**. After entering the expression, click *Verify* button to check whether the numbers in the expression is correct. Display the verification in a message window (see figures below).

Assume that images are stored in files named 1.png, 2.png, … , 52.png, in the order of spades, hearts, diamonds, and clubs. So, the first 13 images are for spades 1, 2, 3, … , 13.

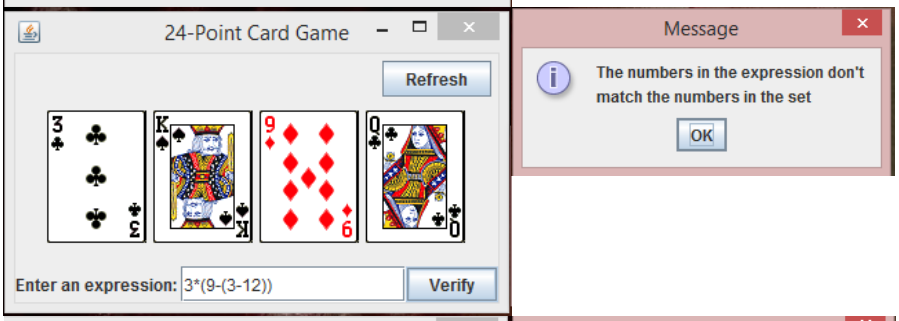
Assume that no spaces are entered before the first token, in between tokens, and after the last tokens.

Assume that only numbers can be entered as operands.

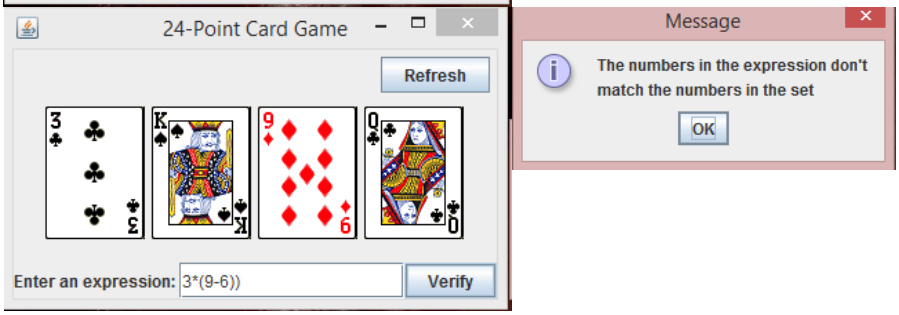
Numbers match and the result is 24.



Numbers don't match.



Numbers don't match.

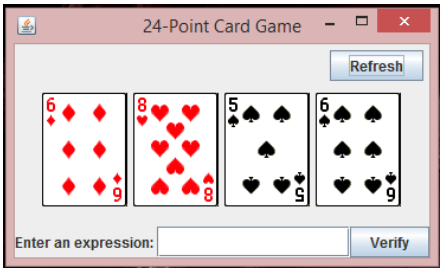


Specification/Design/Implementation/Test/Debug:

- **GUI:**

  A graphical user interface is required. Use swing components/containers. Proper events must be handled. It is recommended that a listener is implemented in the most reusable format. Here is the containment hierarchy in this GUI:

  - A driver program creates a frame in *main*.
  - A JFrame contains the panel.
  - A JPanel contains all components in proper layouts.
    - A card can be made using a label with an image icon.
    - To store images, create folder *image* in the project folder, create a subfolder *card*, and store all images in folder *card*.



The following method can be used to shuffle cards:
void java.util.Collections.shuffle(List<?> list)
Randomly permutes the specified list using a default source of randomness. All permutations occur with approximately
Parameters:
list the list to be shuffled.
Throws:
UnsupportedOperationException - if the specified list or its list-iterator does not support the set operation.

- **Other classes:**

  To evaluate an user-enter infix expression, the infix expression MUST be converted to postfix expression first, and then the postfix expression is evaluated if numbers are correct. Define Class *Expression* including the following methods:

  - *infixToPostfix*: converts an infix expression to a postfix expression. The algorithm is discussed in class.
  - *evaluate*: evaluates a postfix expression. The algorithm is discussed in class.
  - Supporting methods.

  Both algorithms must use a stack to store objects/tokens. Define class *GenericStack* (LIFO) implemented with an array list. It should have the following operations:

  - Constructs an empty stack (*GenericStack(){…}*).
  - Returns the number of objects of this stack(*getSize() {… }*).
  - Returns a reference to the top element of this stack(*peek() {…}*).
  - Adds an object to the top of this stack(*push(E o) {…}*).
  - Removes from the top of this stack(*pop() {…}*).
  - Indicates if this stack is empty(*isEmpty() {…}*).

```
public class GenericStack<E>{
      private java.util.ArrayList<E> list;
    …
}
```