CS212 – Lab 4

Extending the Stack Class & Creating JavaDoc

GOALS:

- Practice creating subclasses including:
  - use of existing instance variables
  - calling constructors/methods in the superclass correctly
  - creating new instance methods in the subclass
- Gain a better understanding of the Stack class
- Practice creating and using Exception classes
- Practice creating JavaDoc

Problem:

The purpose of this lab is to develop a discard pile class and implement some of the operations that can be performed on a discard pile. A discard pile contains 0 or more cards. A player in a card game can either choose to place a new card onto the discard pile or pick up 1 or more cards from the pile.

Exercise:

- Create a new project named **lab4**
- Within this project create a new class named **Card**.
  - Code for the Card class can be found at http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/Card.java
  - Modify the comments at the beginning of the class to include your names as the authors.
  - Copy the following into the project.
    http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/StackADT.java
    http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/Stack.java
    http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/StackFullException.java
    http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/StackEmptyException.java

Modify the *StackFullException* and *StackEmptyException* classes:

1. Add comments at the beginning to include a title, a description, and your names as the authors.
2. Include the appropriate JavaDoc for each of method. Refer to the JavaDoc in the Stack class or Card class.

Within this project create a new class named *DiscardPile*:

3. DiscardPile should be a subclass of Stack.
4. Write a default constructor. What should the maximum size of the discard pile be? How many cards should be in the discard pile initially?
5. Include the appropriate JavaDoc.
6. Add comments at the beginning of the class to include your names as the authors

Within this project create a new class named *Lab4App*:

7. Define the main method.
8. Create a DiscardPile object, referenced by **discardPile**. **Answer question #1 on the attached sheet**.
9. Populate the discard pile by pushing the following cards: (**Note:** For now you can simply acknowledge that the main method may/can throw a StackFullException by including the throws clause at the end of the main method header)

```
discardPile.push(new Card(8));
discardPile.push(new Card(32));
discardPile.push(new Card(48));
discardPile.push(new Card(2));
discardPile.push(new Card(17));
discardPile.push(new Card(20));
discardPile.push(new Card(25));
discardPile.push(new Card(50));
discardPile.push(new Card(19));
discardPile.push(new Card(41));
```

Within the *Stack* class:

10. Note the ***toString*** method in the Stack class; modify it to suit yourself.

Within the *application* class:

11. Display the contents of the discard pile. Your output should look like the following.

```
4 of Spades
8 of Diamonds
K of Spades
A of Diamonds
9 of Diamonds
6 of Diamonds
4 of Clubs
J of Spades
8 of Hearts
10 of Clubs
```

12. **Answer question #2 on the attached sheet.**

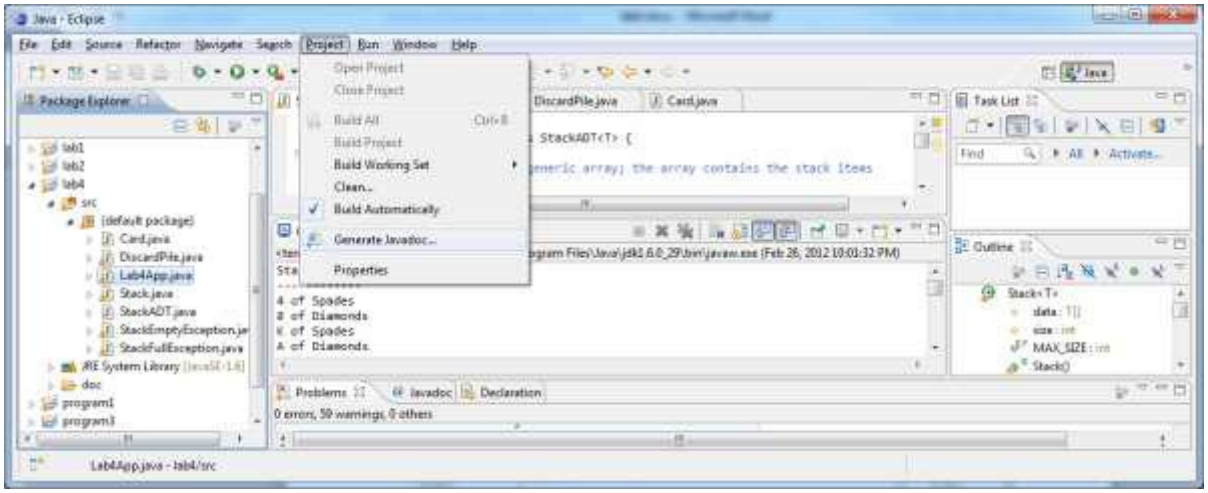
Within the *DiscardPile* class:

13. Write a method called **removeTopCards** which accepts a reference to a Card object called **theCard**. This method will allow the user to request that all cards up to and including **theCard** be removed from the discard pile and returned in an array, thus the return-type of the method is Card[ ]. It is important that you think through the code required to accomplish this BEFORE you begin writing it. Consider the following as you plan:
    - You can use the ***equals*** method in the Card class to compare two cards.
    - You must use a while loop to find the card in the discard pile (stack).
    - You should throw a Stack exception if the Card is not found. There are two reasons **theCard** may not be found. First, if the discard pile (stack) is empty and second, if the card is not found within the discard pile. You should throw an exception in both cases, but the message stored in the exception object should indicate the type of error that occurred.
    - If the card is found, remove the appropriate cards from the stack and store their references in an array.
    - The cards should be in order from **theCard** to the top of the discard pile, assuming the card has been found.
    - The array reference should be returned, again assuming the card has been found.

Modify Lab4App:

14. Write a call to the **removeTopCards** method. You should test the method by passing a reference to a card whose integer value is 20. Be sure to store the return value appropriately.
15. Display the cards whose references have been stored in the array. Your output should look like
    9 of Diamonds
    A of Diamonds
    K of Spades
    8 of Diamonds
    4 of Spades
    and should be in this order.
16. Display the contents of the discard pile to ensure that the appropriate cards have been removed.
17. Test the method again by calling the removeTopCards method and passing a reference to a card whose integer value is 50. **Note**: you should make the discard pile empty and repopulate it with the same cards as before prior to removing cards. **Answer question #3 on the attached sheet.**
18. Display the cards whose references have been stored in the array. Your output should look like
    K of Spades
    8 of Diamonds
    4 of Spades
    and should be in this order.
19. Display the contents of the discard pile to ensure that the appropriate cards have been removed.

Generate JavaDoc for your classes:



The following comment is formatted using JavaDoc conventions. These comments can be converted to html and the html file can be viewed in a web browser.

```
/**
* methodName – purpose of the method
* @param variableName description of what the variable represents or how it is
*  used by this method
* @return description of what is returned by the method
* @throws Exception name and the reason why it would occur
*/
```

Please be sure to consider the following when writing comments:

a) Specify the name of the method along with a description of what the method does.

b) @param is a tag which should be followed the name of a parameter and then by a brief description of what the method does with the parameter or what the parameter represents.

c) @return is a tag which should be followed by a description of what is being returned.

d) Be consistent! Your JavaDoc comments should have the same look and feel.

e) If the method header already exists, you can position the cursor above the method, type /**, and then press enter. The editor will automatically include some of this for you.

You should refer to these guidelines when asked to write JavaDoc comments.

See JavaDoc generated from some of the classes in this lab:

http://venus.cs.qc.cuny.edu/~aabreu/cs212/lab4/generics.html

Lab #4 – Extending the Stack Class

Questions:
Answer the following questions in a document. Please include blank lines between your answers.

1. Describe the code that is executed. Be specific! Graphically describe what the object referenced by discardPile looks like. (Note: you should leave enough room to draw the DiscardPile object after printing the document)

2. Which card is on the **top** of the discard pile? What card is associated with the integer 2? What card is associated with the integer 19?

3. Why is it necessary to make the discard pile empty and repopulate it before attempting another call to removeTopCards?

Submit:

- DiscardPile.java, Lab4App.java, and the document containing your answers to the questions.