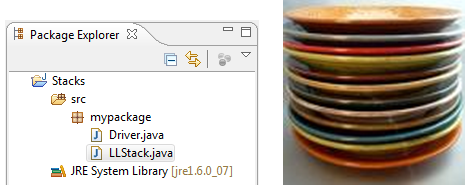
******

***Hands-On Exercise 4.1 [20-points]: Stacks***

### *Instructions:*

* According to some research, industry values documentation, and excellent written and oral communication skills. The purpose of this part of the class is to encourage you to gain these skills.
* Backup your work to your USB drive for this material may come out as part of your examination.
* Make a copy of this entire document and add your work into it.
* Submit to Blackboard at the same link where you got this document.
* Points will be a deducted if submitted on the wrong place, or if these instructions are not followed.

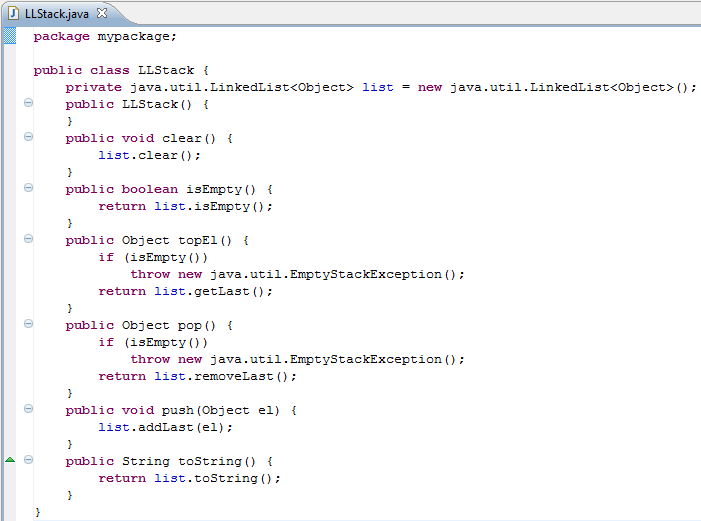
***IDE structure:***



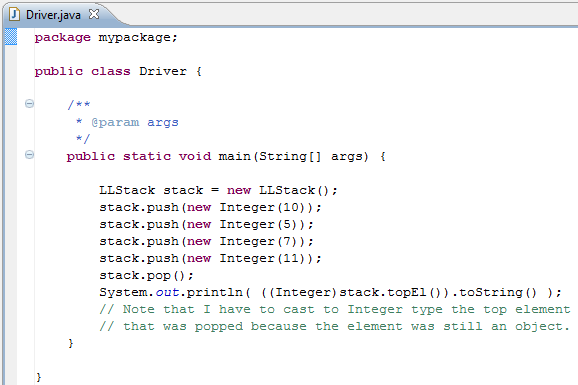
**LIFO – Last In, First Out**

Stacks can be used for stacking images or 3d/2d models to be used for LOD (level of details) during transformation (e.g. scaling and projecting). It may use in calculator, or any application that wants to keep the chronological order.

1.❑ Add a LLStack class and type the code below



2.❑ Add Driver class and type the code below



3.❑ Paste your code here.

**package** Exercise1;

**public** **class** LLStack {

**private** java.util.LinkedList<Object> list = **new** java.util.LinkedList<Object>();

**public** LLStack(){

}

**public** **boolean** isEmpty(){

**return** list.isEmpty();

}

**public** Object topEl(){

**if**(isEmpty())

**throw** **new** java.util.EmptyStackException();

**return** list.getLast();

}

**public** Object pop(){

**if**(isEmpty())

**throw** **new** java.util.EmptyStackException();

**return** list.removeLast();

}

**public** **void** push(Object el){

list.addLast(el);

}

**public** String toSting(){

**return** list.toString();

}

}

**package** Exercise1;

**public** **class** Driver {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

LLStack stack = **new** LLStack();

stack.push(**new** Integer(10));

stack.push(**new** Integer(5));

stack.push(**new** Integer(7));

stack.push(**new** Integer (11));

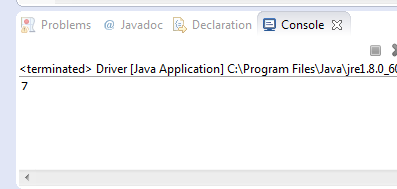
stack.pop();

System.***out***.println((((Integer)stack.topEl()).toString()));

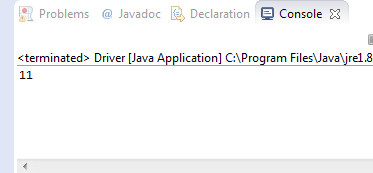
}

}

4.❑ Paste your screen shot output here [Ctrl] + [PrtScn]. Make sure you magnified it.



5.❑ Add another element in the stack with int value of 11, paste below the code that you added and paste the new output.



6.❑ Write your topmost question regarding this topic.

Is there a primitive type of stack? I remember reading about using a stack when I was looking into assembler.

7.❑ **Critical Thinking:** If you are asked to make a test question based on this topic, what would be the question and what is your answer?

True of False: A stack and be accessed from either the top or the bottom.

Answer: False.

[](http://images.google.com/imgres?imgurl=www.skyscript.co.uk/im/trophy.jpg&imgrefurl=http://www.skyscript.co.uk/im/&h=214&w=180&sz=6&tbnid=ECCiP8U-7NsJ:&tbnh=99&tbnw=84&prev=/images?q=trophy&svnum=10&hl=en&lr=&ie=UTF-8&oe=UTF-8&sa=G)Congratulations! You’ve just learned how to implement stacks using Java utility’s linked list class.

**Submission Procedure**

1. Write your **name** here:\_Joshua LeGoff\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Date: \_10/27/2015\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Backup** your work to your USB drive, this material may come out as part of your exam.
4. **Submit** to Blackboard at the link where you got it.

**Note:**

* Submit back to Blackboard where you get it.
* 2-points deduction if you submit it on the wrong place.
* 2-points deduction if you did not follow these instructions.
* Make sure you submit it at the correct location where you got it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GRADING RUBRIC | | | | |
| Grading Criteria | 3  **Exceeds**  *Excellent*  Epic Wow | 2  **Meets**  *Satisfactory*  O.K. | 1  **Partially Meets**  *Below Expectations*  Not Yet | 0  **Does Not Meet**  *Unacceptable*  Fail |
| **Completeness** | +5-Completed all the required work and added more examples. | +2-Completed all the work required. | +1-Partially completed the work required. | Unfortunately, did not complete the work required. |
| **Coding** | +10- Code is excellent, comments are added, and different techniques were used. | +7-Code is O.K., and program works. | +4-Code works, but still needs improvement. | Unfortunately, no coding. |
| **Output** | +5-Outputs are correct, and provided additional output cases. | +2-Output meets requirement and is readable. | +1-There is output, but not readable, and/or needs improvement. | Unfortunately, no output. |
| **Late** | Excellent, you submitted it before the deadline. | -5, unfortunately for submitting after the deadline. | -7, unfortunately for submitting several weeks after the deadline. | -10, unfortunately, for submitting very late. |