Data Structures & Applications

Summer 2022

Lab 03 – Stacks, Queues

**Instructor: Saif Hassan Date: 17h September, 2022**

**Instructions:**

* At the end of this Lab, you will have to submit all files on LMS.
* File format should be **.zip/.rar** file containing required **.java** files and additional if required.
* File Name should be your **CMSID\_Name\_Lab03.zip.**
* Create a project named lab03\_dsa and perform following tasks.

**Stack**

**Note: Keep this code with you till the course ends.**

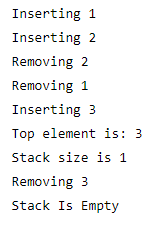
**Task 01: (Stack using array)**

Understand provided code and implement all required methods in Stack. Stack Code is given below:

1. **import** java.util.\*;
3. **class** Stack
4. {
5. **private** **int** arr[];
6. **private** **int** top;
7. **private** **int** capacity;
9. // Constructor to initialize stack
10. Stack(**int** size)
11. {
12. arr = **new** **int**[size];
13. capacity = size;
14. top = -1;
15. }
17. // Utility function to add an element x in the stack and check for stack overflow
18. **public** **void** push(**int** x)
19. {
20. // Write your code here
21. }
23. // Utility function to pop top element from the stack and check for stack underflow
24. **public** **int** pop()
25. {
26. // Write your code here
27. }
29. // Utility function to return top element in a stack
30. **public** **int** peek()
31. {
32. // Write your code here
33. }
35. // Utility function to return the size of the stack
36. **public** **int** size()
37. {
38. // Write your code here
39. }
41. // Utility function to check if the stack is empty or not
42. **public** Boolean isEmpty()
43. {
44. // Write your code here
45. }
47. // Utility function to check if the stack is full or not
48. **public** Boolean isFull()
49. {
50. // Write your code here
51. }
53. **public** **static** **void** main (String[] args)
54. {
55. Stack stack = **new** Stack(3);
57. stack.push(1);      // Inserting 1 in the stack
58. stack.push(2);      // Inserting 2 in the stack
60. stack.pop();        // removing the top 2
61. stack.pop();        // removing the top 1
63. stack.push(3);      // Inserting 3 in the stack
65. System.out.println("Top element is: " + stack.peek());
66. System.out.println("Stack size is " + stack.size());
68. stack.pop();        // removing the top 3
70. // check if stack is empty
71. **if** (stack.isEmpty())
72. System.out.println("Stack Is Empty");
73. **else**
74. System.out.println("Stack Is Not Empty");
75. }
76. }

After implementing all the methods, run the code. Your output should be like as follows:

**Output**



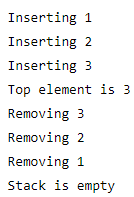
**Task 02:(Stack using Linked list)**

Understand provided code and implement all required methods in Stack. Stack Code is given below:

1. **import** java.util.\*;
3. // A linked list node
4. **class** Node
5. {
6. **int** data;       // integer data
7. Node next;      // pointer to the next node
8. };
10. **class** Stack
11. {
12. **private** Node top;
14. **public** Stack() {
15. **this**.top = **null**;
16. }
18. // Utility function to add an element x in the stack
19. **public** **void** push(**int** x) // insert at the beginning
20. {
21. // Write your code here
22. }
24. // Utility function to check if the stack is empty or not
25. **public** **boolean** isEmpty()
26. {
27. // Write your code here
28. }
30. // Utility function to return top element in a stack
31. **public** **int** peek()
32. {
33. // Write your code here
34. }
36. // Utility function to pop top element from the stack and check for Stack underflow
37. **public** **void** pop() // remove at the beginning
38. {
39. // Write your code here
40. }
41. }
43. **class** StackImpl
44. {
45. **public** **static** **void** main(String[] args)
46. {
47. Stack stack = **new** Stack();
49. stack.push(1);
50. stack.push(2);
51. stack.push(3);
53. System.out.println("Top element is " +  stack.peek());
55. stack.pop();
56. stack.pop();
57. stack.pop();
59. **if** (stack.isEmpty()) {
60. System.out.print("Stack is empty");
61. } **else** {
62. System.out.print("Stack is not empty");
63. }
64. }
65. }

After implementing all the methods, run the code. Your output should be like as follows:

**Output**



**Queue**

**Note: Keep this code with you till the course ends.**

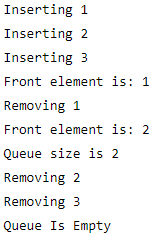
**Task 03: (Queue using array)**

Understand provided code and implement all required methods in Queue. Queue Code is given below:

1. import java.util.\*;
3. // Class for queue
4. **class** Queue
5. {
6. **private** **int** arr[];
7. **private** **int** front;
8. **private** **int** rear;
9. **private** **int** capacity;
10. **private** **int** count;
12. // Constructor to initialize queue
13. Queue(**int** size)
14. {
15. arr = **new** **int**[size];
16. capacity = size;
17. front = 0;
18. rear = -1;
19. count = 0;
20. }
22. // Utility function to remove front element from the queue and check for Queue Underflow
23. **public** **void** dequeue()
24. {
25. // Write your code here
26. }
28. // Utility function to add an item to the queue and check for queue overflow
29. **public** **void** enqueue(**int** item)
30. {
31. // Write your code here
32. }
34. // Utility function to return front element in the queue and check for Queue Underflow
35. **public** **int** peek()
36. {
37. // Write your code here
38. }
40. // Utility function to return the size of the queue
41. **public** **int** size()
42. {
43. // Write your code here
44. }
46. // Utility function to check if the queue is empty or not
47. **public** Boolean isEmpty()
48. {
49. // Write your code here
50. }
52. // Utility function to check if the queue is empty or not
53. **public** Boolean isFull()
54. {
55. // Write your code here
56. }
57. }
59. **class** Main
60. {
61. // main function
62. **public** **static** **void** main (String[] args)
63. {
64. // create a queue of capacity 5
65. Queue q = **new** Queue(5);
67. q.enqueue(1);
68. q.enqueue(2);
69. q.enqueue(3);
71. System.out.println("Front element is: " + q.peek());
72. q.dequeue();
73. System.out.println("Front element is: " + q.peek());
75. System.out.println("Queue size is " + q.size());
77. q.dequeue();
78. q.dequeue();
80. **if** (q.isEmpty())
81. System.out.println("Queue Is Empty");
82. **else**
83. System.out.println("Queue Is Not Empty");
84. }
85. }

After implementing all the methods, run the code. Your output should be like as follows:

**Output**



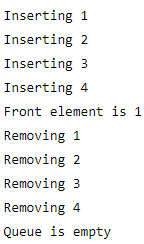
**Task 04:(Queue using Linked list)**

Understand provided code and implement all required methods in Queue. Queue Code is given below:

1. // A linked list node
2. **class** Node
3. {
4. **int** data;       // integer data
5. Node next;      // pointer to the next node
7. **public** Node(**int** data)
8. {
9. // set the data in allocated node and return the node
10. **this**.data = data;
11. **this**.next = null;
12. }
13. }
15. **class** Queue
16. {
17. **private** **static** Node rear = null, front = null;
19. // Utility function to remove front element from the queue and check for Queue Underflow
20. **public** **static** **int** dequeue()     // delete at the beginning
21. {
22. // Write your code here
23. }
25. // Utility function to add an item in the queue
26. **public** **static** **void** enqueue(**int** item)    // insertion at the end
27. {
28. // Write your code here
29. }
31. // Utility function to return top element in a queue
32. **public** **static** **int** peek()
33. {
34. // Write your code here
35. }
37. // Utility function to check if the queue is empty or not
38. **public** **static** boolean isEmpty()
39. {
40. // Write your code here
41. }
42. }
44. **class** Main {
45. **public** **static** **void** main(String[] args)
46. {
47. Queue q = **new** Queue();
48. q.enqueue(1);
49. q.enqueue(2);
50. q.enqueue(3);
51. q.enqueue(4);
53. System.out.printf("Front element is %d\n", q.peek());
55. q.dequeue();
56. q.dequeue();
57. q.dequeue();
58. q.dequeue();
60. **if** (q.isEmpty()) {
61. System.out.print("Queue is empty");
62. } **else** {
63. System.out.print("Queue is not empty");
64. }
65. }
66. }

After implementing all the methods, run the code. Your output should be like as follows:

**Output**



Queue using two Stacks

Question 5: Understand provided code and implement all required methods in Queue Class. Sample Code is given below:

1. import java.util.\*;
3. // Implement Queue using two stacks
4. **class** Queue {
5. **private** Stack s1, s2;
7. // Constructor
8. Queue() {
9. s1 = **new** Stack();
10. s2 = **new** Stack();
11. }
13. // Enqueue an item to the queue
14. **public** **void** enqueue(**int** data)
15. {
16. // Write your code here
17. }
19. // Dequeue an item from the queue
20. **public** **int** dequeue()
21. {
22. // Write your code here
23. }
25. **public** **static** **void** main(String[] args) {
26. **int**[] keys = { 1, 2, 3, 4, 5 };
27. Queue q = **new** Queue();
29. // insert above keys
30. **for** (**int** key : keys) {
31. q.enqueue(key);
32. }
34. System.out.println(q.dequeue());    // print 1
35. System.out.println(q.dequeue());    // print 2
36. }
37. }

After implementing all the methods, run the code.

**Bonus Task:** Think about the inverse of task 05 (Stack using queue)

**Specifications, notes, and hints**

Your program needs to meet the following specifications:

* Submit all files LinkedList.java and additional files if applicable.
* When commenting your code use Javadoc style comments at the beginning of each method.
* Put comments at the top of the file (Java File) with your name, S\_ID, S\_Name, date and course, and a short (one or two line) description of what the program does. Make sure your code runs on machine.
* Submit your source code files via the classroom by the due date (remember the course syllabus for the late policy).