|  |  |  |
| --- | --- | --- |
| Sukkur_IBA_New_Logo | **Sukkur IBA University**  **Department of Computer Science** | **C:\Users\Saif Hassan\Downloads\CS logo (3).jpg** |

**DATA STRUCTURES**

**Lab01 – Arrays, LinkesLists**

**Instructor: Saif Hassan**

**READ IT FIRST**

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don’t waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

**KEEP IT WITH YOU!**

1. Indent your code inside the classes and functions. It’s a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don’t jump to the formula directly.

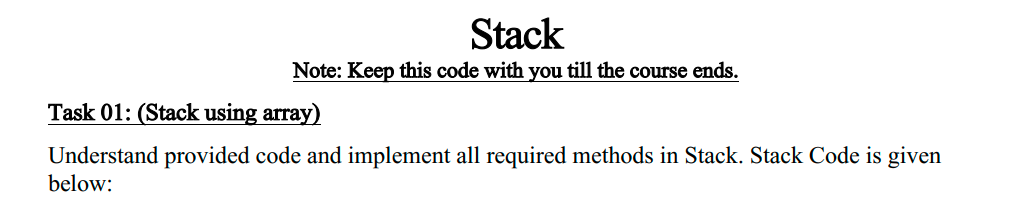
I\_**Amjad Ali**\_ with student ID \_**191-21-0001**\_

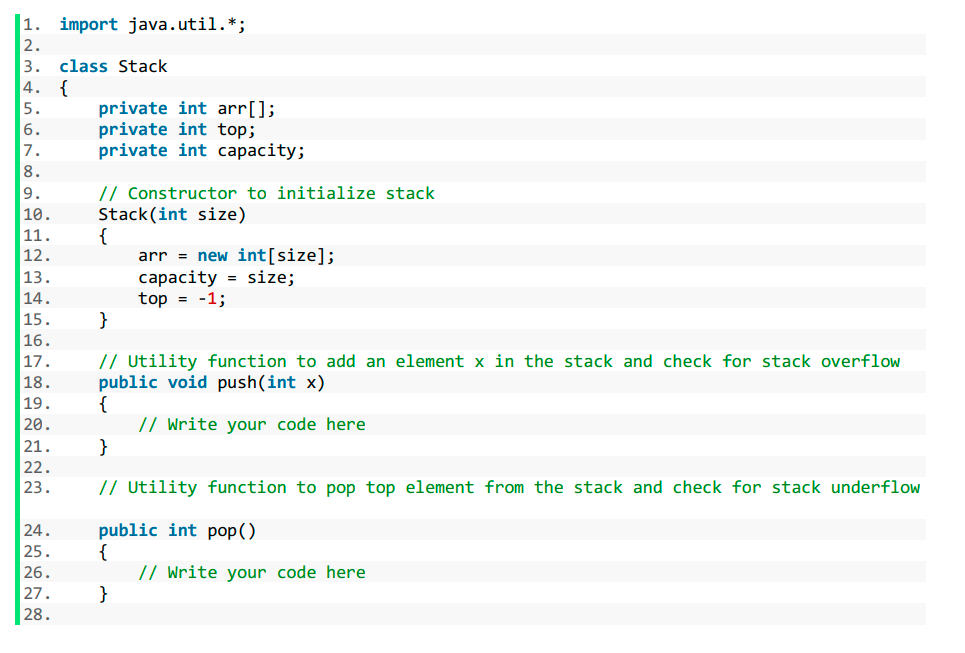
Section **\_\_\_”A”\_\_\_\_**hereby declare that I do understand the instructions above and follow them. This is

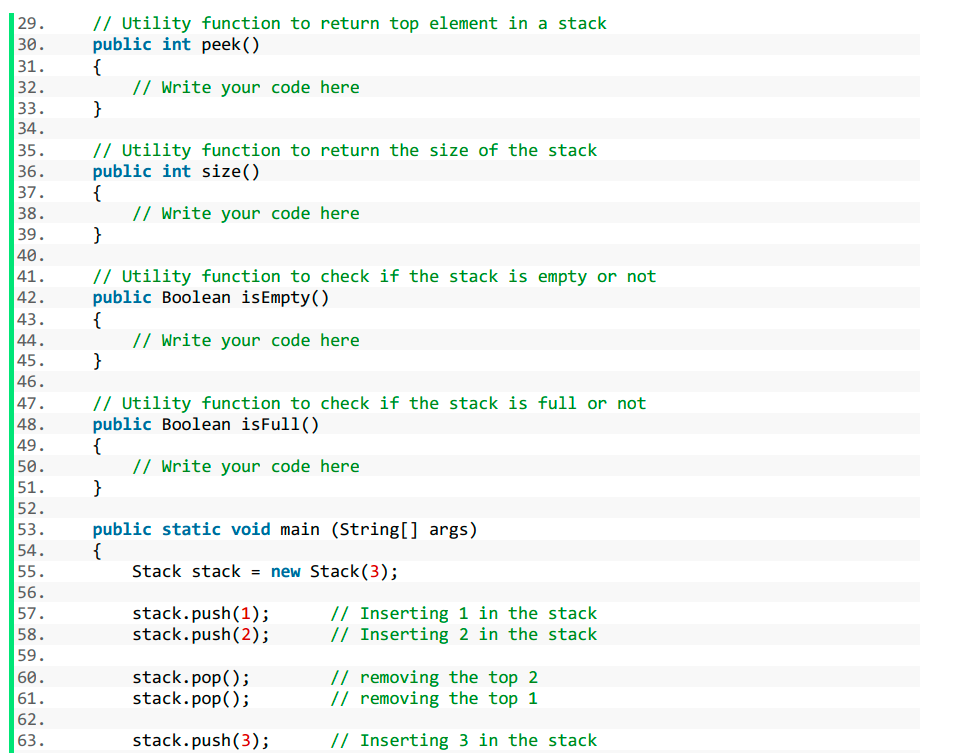
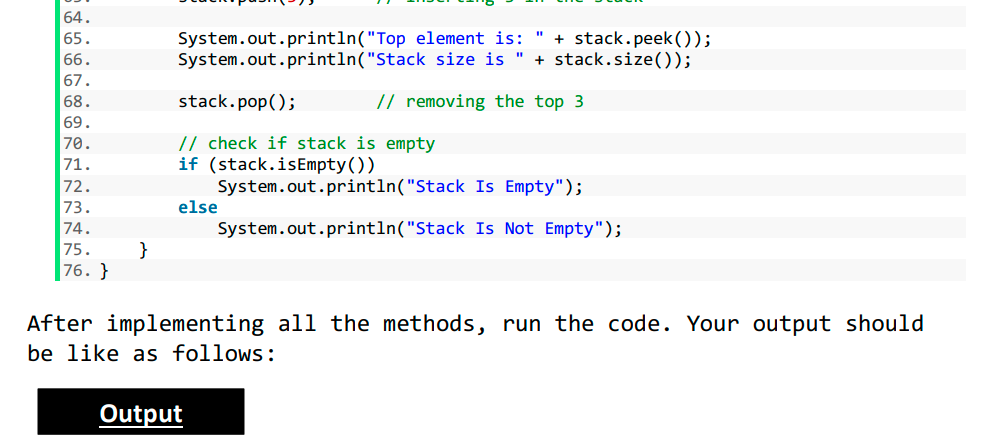
my own work.

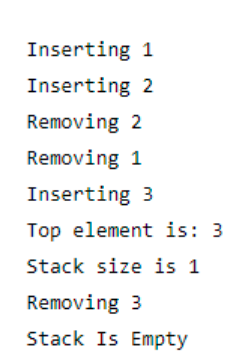
**Exercises**

**Task1 Description**







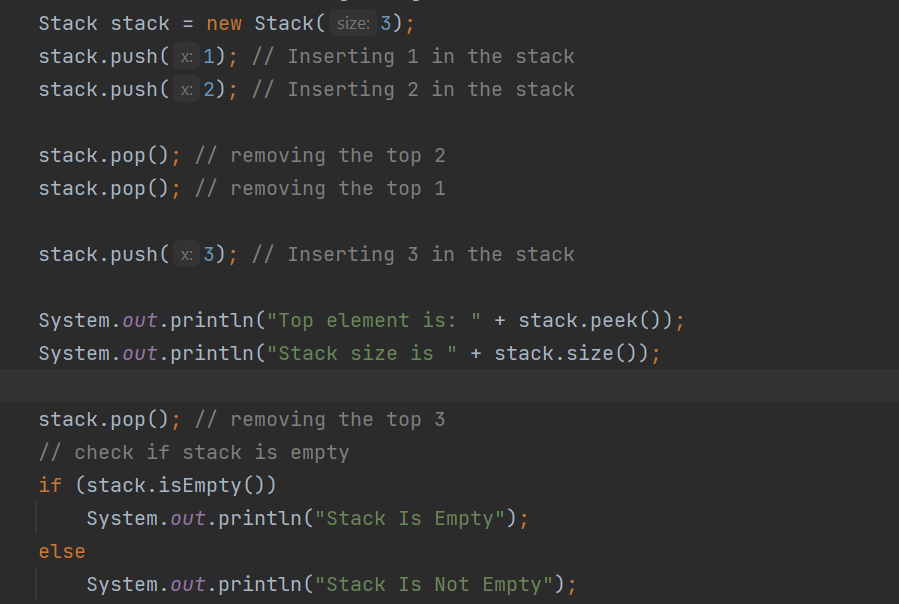


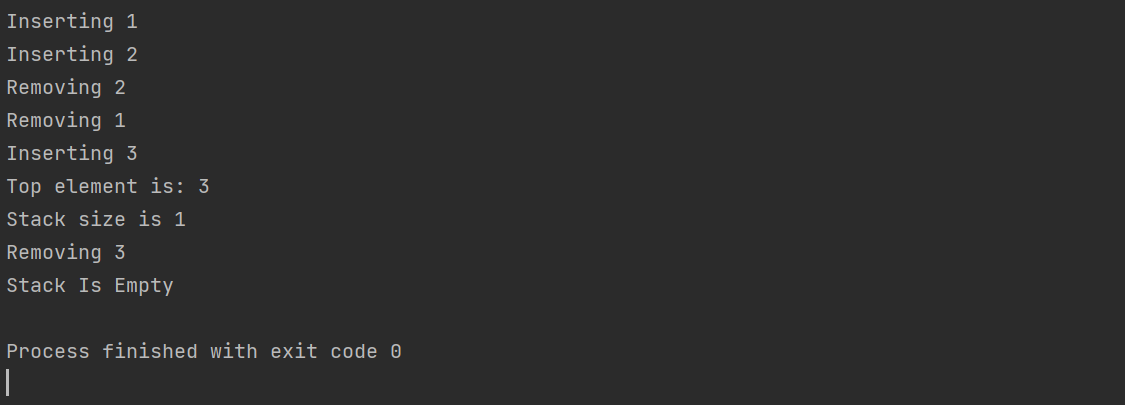
**Solution:**

Code:

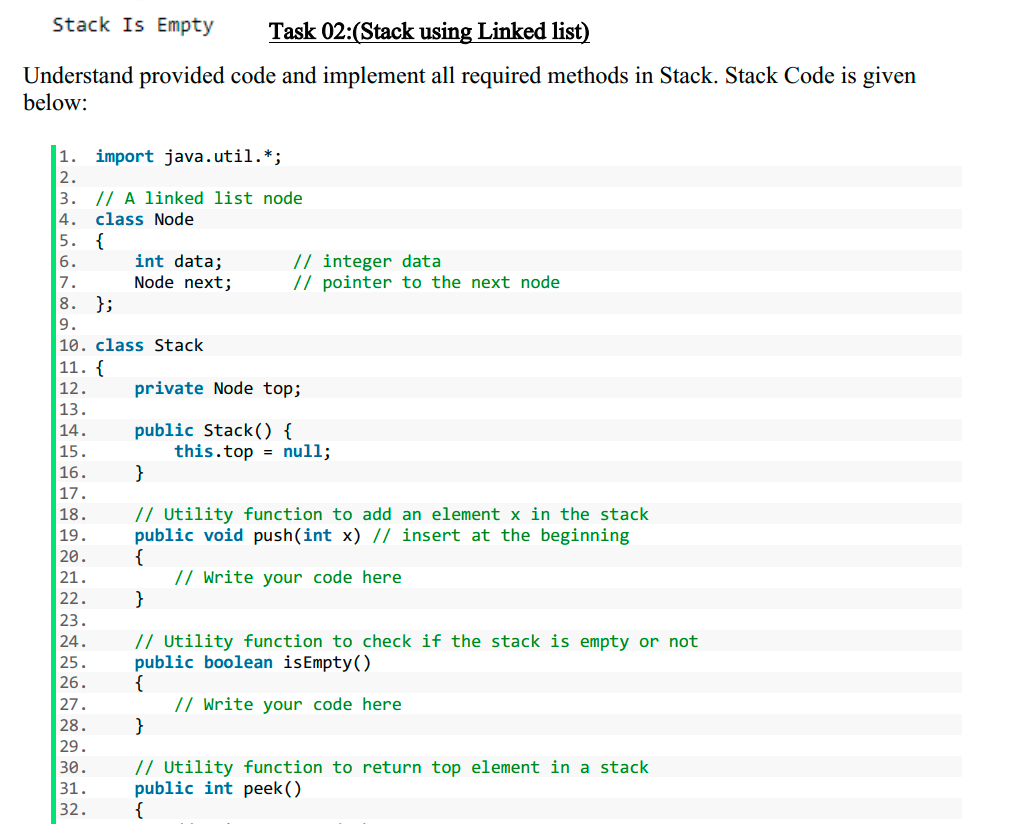
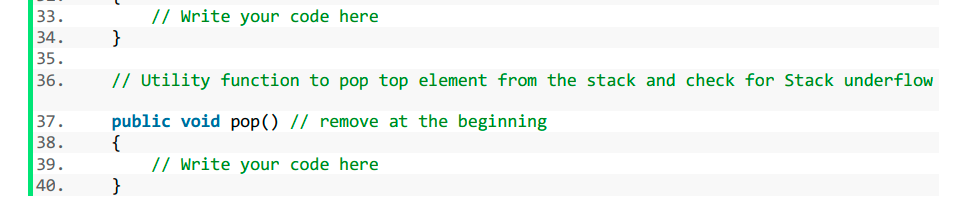
1. import java.util.\*;
3. class Stack {
4. private int arr[];
5. private int top;
6. private int capacity;
8. // Constructor to initialize stack
9. Stack(int size) {
10. arr = new int[size];
11. capacity = size;
12. top = -1;
13. }
15. // Utility function to add an element x in the stack and check for stack overflow
16. public void push(int x) {
17. // Write your code here
18. if (isFull()) {
19. System.out.println("Stack Overflow");
20. } else {
21. System.out.println("Inserting "+x);
22. arr[++top] = x;
23. }
24. }
25. // Utility function to pop top element from the stack and check for stack underflow
27. public int pop() {
28. // Write your code here
29. if(isEmpty())
30. {
31. System.out.println("Stack underflow");
32. return -1;
33. }
34. else{
35. System.out.println("Removing "+arr[top]);
36. return arr[top--];
37. }
38. }
40. // Utility function to return top element in a stack
41. public int peek() {
42. // Write your code here
43. if(isEmpty())
44. {
45. System.out.println("Stack underflow");
46. return -1;
47. }
48. else{
49. return arr[top];
50. }
51. }
53. // Utility function to return the size of the stack
54. public int size() {// Write your code here
55. return top + 1;
56. }
58. // Utility function to check if the stack is empty or not
59. public Boolean isEmpty() {// Write your code here
60. return top == -1;
61. }
63. // Utility function to check if the stack is full or not
64. public Boolean isFull() {
65. // Write your code here
66. return top + 1 == capacity;
67. }
69. public static void main(String[] args) {
70. Stack stack = new Stack(3);
71. stack.push(1); // Inserting 1 in the stack
72. stack.push(2); // Inserting 2 in the stack
74. stack.pop(); // removing the top 2
75. stack.pop(); // removing the top 1
77. stack.push(3); // Inserting 3 in the stack
79. System.out.println("Top element is: " + stack.peek());
80. System.out.println("Stack size is " + stack.size());
82. stack.pop(); // removing the top 3
83. // check if stack is empty
84. if (stack.isEmpty())
85. System.out.println("Stack Is Empty");
86. else
87. System.out.println("Stack Is Not Empty");
88. }
89. }

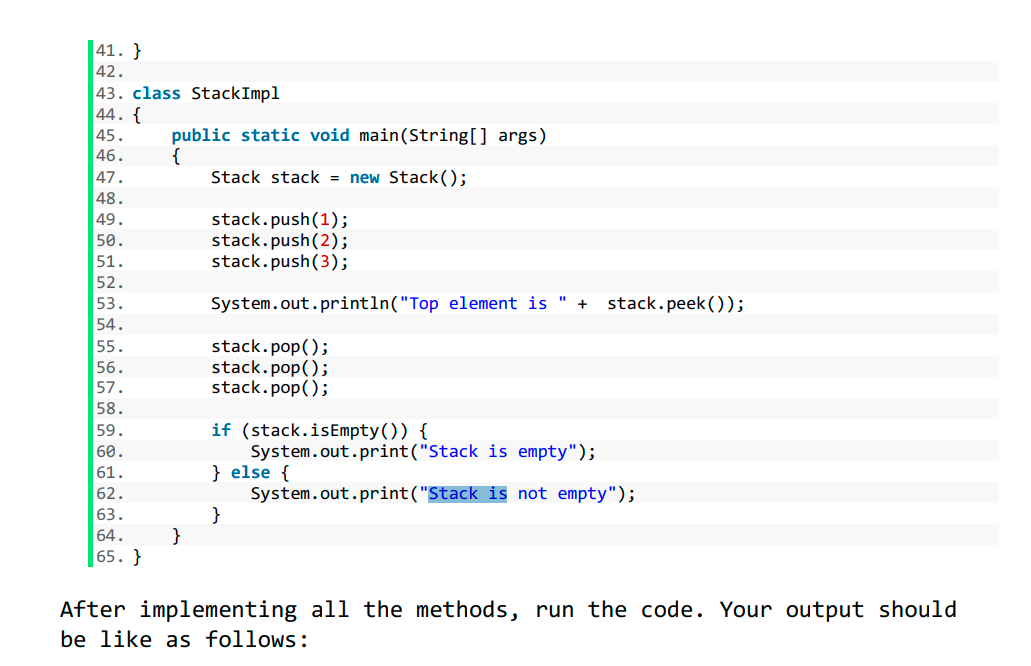
**Sample Input:**

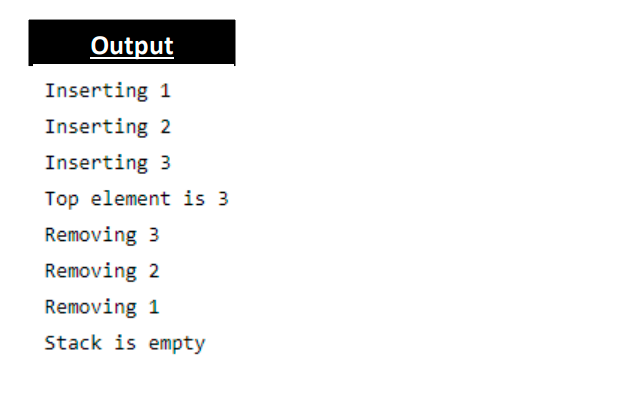
**Sample Output**



**Task2 Description**







**Solution:**

Code:

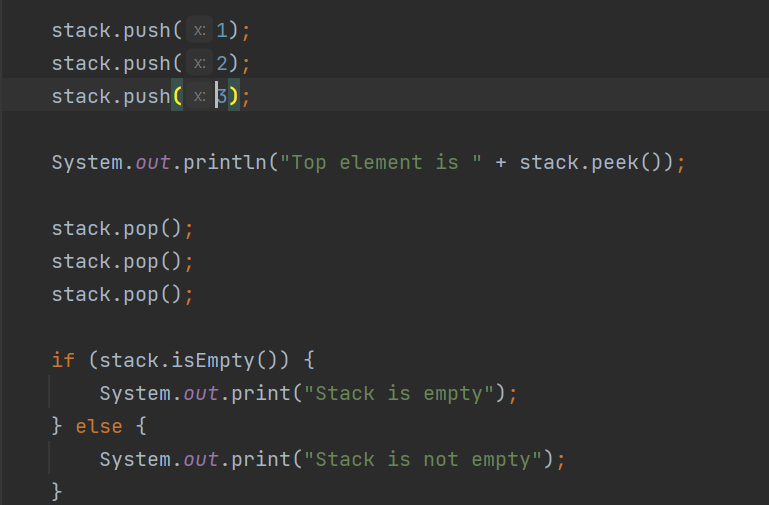
# Node and Stack Classes

1. package com.company;
3. import java.util.\*;
5. // A linked list node
6. class Node {
7. int data; // integer data
8. Node next; // pointer to the next node
10. Node(int data) {
11. this.data = data;
12. this.next = null;
13. }
14. }
16. class Stack {
17. private Node top, tail;
19. Stack() {
20. this.top = null;
21. this.tail = null;
22. }
24. // Utility function to add an element x in the stack
25. public void push(int x) // insert at the beginning
26. {// Write your code here
27. Node newNode = new Node(x);
28. if (isEmpty()) {
29. top = tail = newNode;
30. } else {
31. System.out.println("Inserting " + x);
32. newNode.next = top;
33. top = newNode;
34. }
35. }

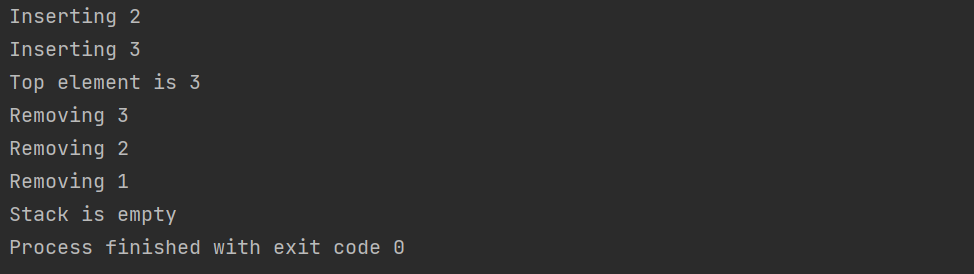
38. // Utility function to check if the stack is empty or not
39. public boolean isEmpty() {
40. // Write your code here
41. return top == null;
42. }
44. // Utility function to return top element in a stack
45. public int peek() {
46. // Write your code here
47. if (isEmpty()) {
48. System.out.println("Stack underflow");
49. return -1;
50. } else {
51. return top.data;
52. }
54. }
56. // Utility function to pop top element from the stack and check for Stack underflow
58. public int pop() // remove at the beginning
59. {// Write your code here
60. if (isEmpty()) {
61. System.out.println("Stack underflow");
62. return -1;
63. } else {
65. int temp = top.data;
66. System.out.println("Removing " + temp);
67. top = top.next;
68. return temp;
70. }
72. }
73. }

# **Stack Implementation Class**

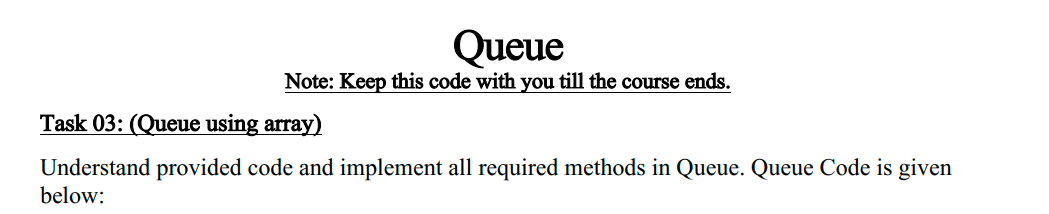
1. package com.company;
3. class StackImpl {
4. public static void main(String[] args) {
5. Stack stack = new Stack();
7. stack.push(1);
8. stack.push(2);
9. stack.push(3);
11. System.out.println("Top element is " + stack.peek());
13. stack.pop();
14. stack.pop();
15. stack.pop();
17. if (stack.isEmpty()) {
18. System.out.print("Stack is empty");
19. } else {
20. System.out.print("Stack is not empty");
21. }
22. }
23. }

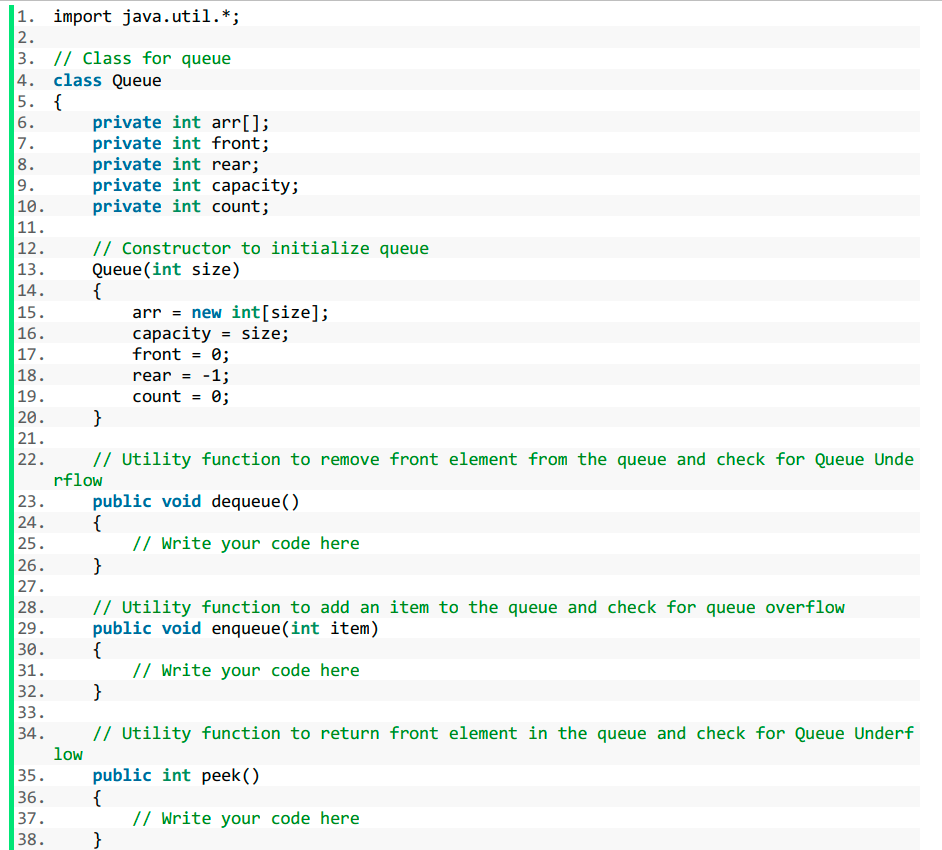
**Sample Input:**

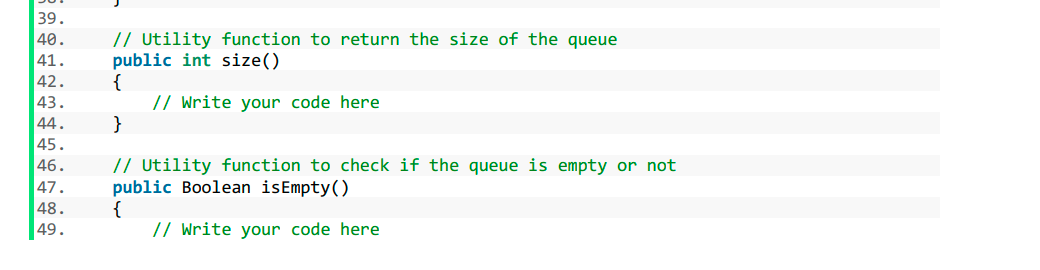
**Sample Output**

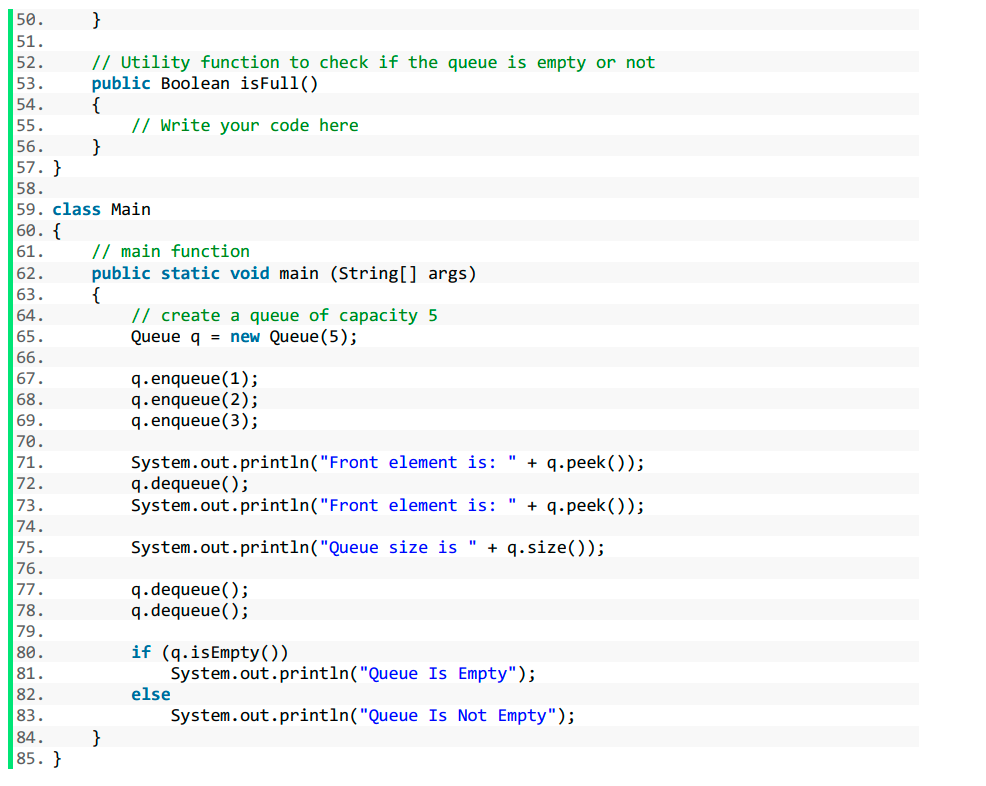


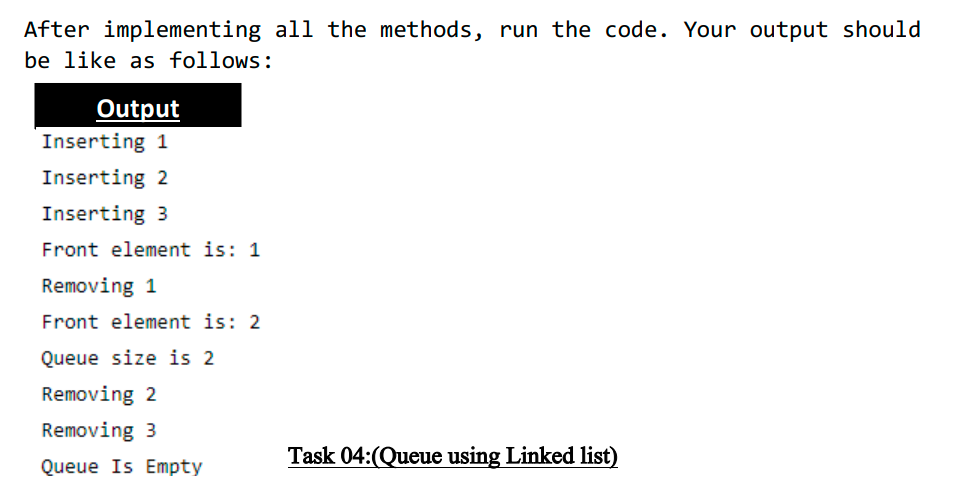
**Task3 Description**











Solution:

Code:-

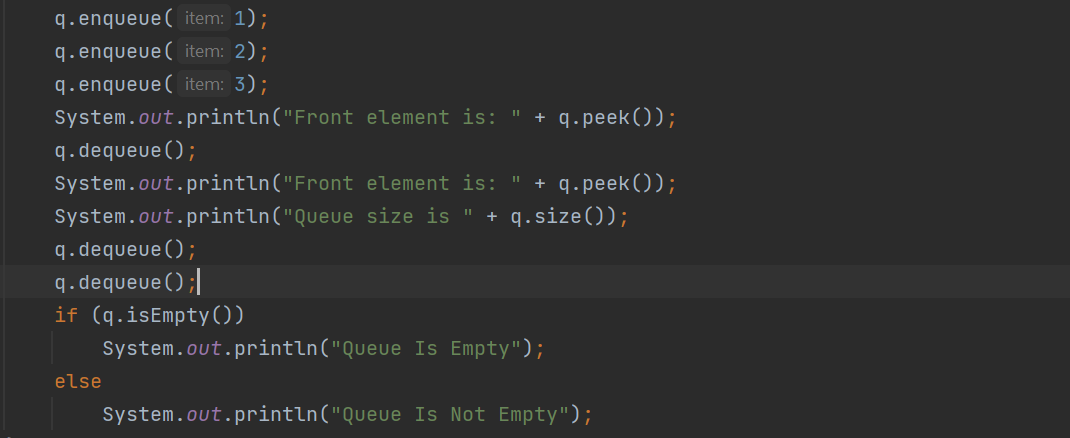
# **Queue Class**

1. package com.company;
3. import java.util.\*;
5. // Class for queue
7. public class Queue {
8. private int arr[];
9. private int front;
10. private int rear;
11. private int capacity;
12. private int count;
14. // Constructor to initialize queue
15. public Queue(int size) {
16. arr = new int[size];
17. capacity = size;
18. front = 0;
19. rear = -1;
20. count = 0;
21. }
23. // Utility function to remove front element from the queue and check for Queue Underflow
24. public void dequeue() {
25. // Write your code here
26. if (isEmpty()) {
27. System.out.println("Stack underflow");
28. } else {
29. System.out.println("Removing " + arr[front]);
30. for (int i = 0; i < rear + 1; i++) {
31. arr[i] = arr[i + 1];
33. }
34. rear = rear - 1;
36. }
38. }
40. // Utility function to add an item to the queue and check for queue overflow
41. public void enqueue(int item) {
42. // Write your code here
43. if (isFull()) {
44. System.out.println("Queue overflow");
45. } else {
46. System.out.println("Inserting " + item);
47. arr[++rear] = item;
48. }
49. }
51. // Utility function to return front element in the queue and check for Queue Underflow
52. public int peek() {
53. // Write your code here
54. if (isFull()) {
55. System.out.println("Queue overflow");
56. return -1;
57. } else {
58. int temp = arr[front];
59. return temp;
60. }
62. }
64. // Utility function to return the size of the queue
65. public int size() {
66. // Write your code here
67. return rear + 1;
68. }
70. // Utility function to check if the queue is empty or not
71. public Boolean isEmpty() {
72. // Write your code here
73. return rear == -1;
74. }
76. // Utility function to check if the queue is empty or not
77. public Boolean isFull() {
78. // Write your code here
79. return rear + 1 == capacity;
80. }
82. }

# **Queue Implementation Class**

1. import com.company.Queue;
3. public class QueueImpl {
4. // main function
5. public static void main(String[] args) {
6. // create a queue of capacity 5
7. Queue q = new Queue(5);
9. q.enqueue(1);
10. q.enqueue(2);
11. q.enqueue(3);
13. System.out.println("Front element is: " + q.peek());
14. q.dequeue();
15. System.out.println("Front element is: " + q.peek());
17. System.out.println("Queue size is " + q.size());
19. q.dequeue();
20. q.dequeue();
22. if (q.isEmpty())
23. System.out.println("Queue Is Empty");
24. else
25. System.out.println("Queue Is Not Empty");
26. }
27. }

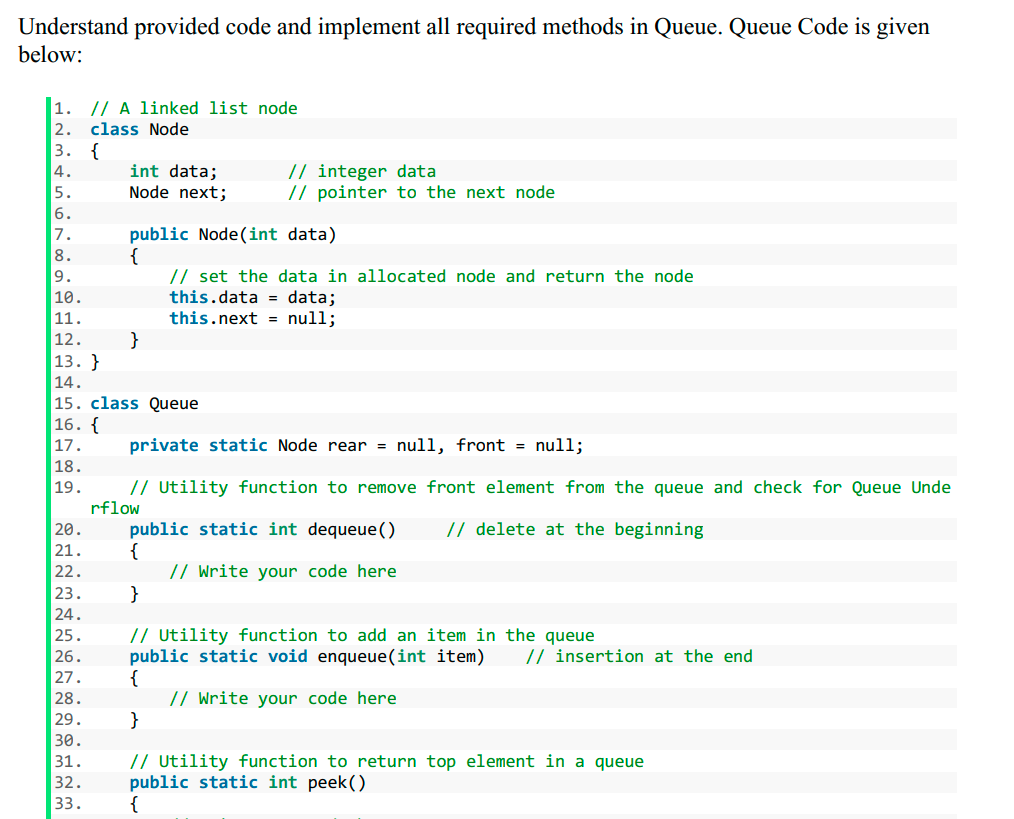
**Sample Input:**

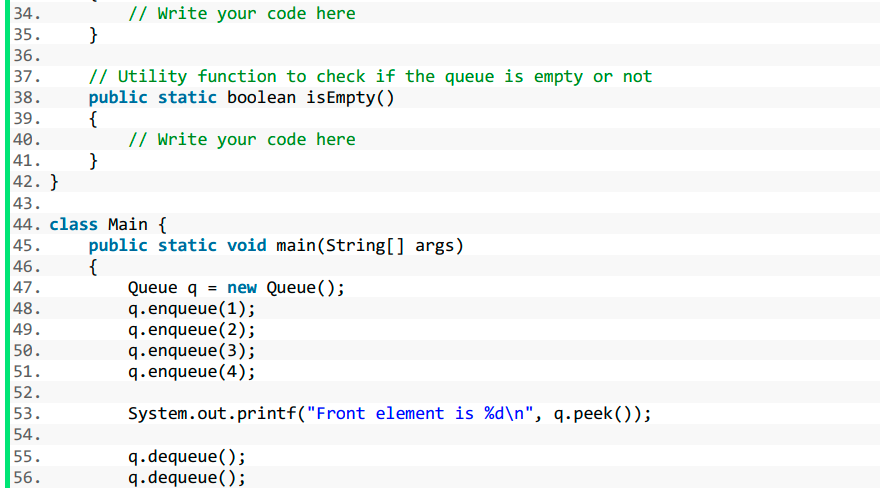
**Sample Output**

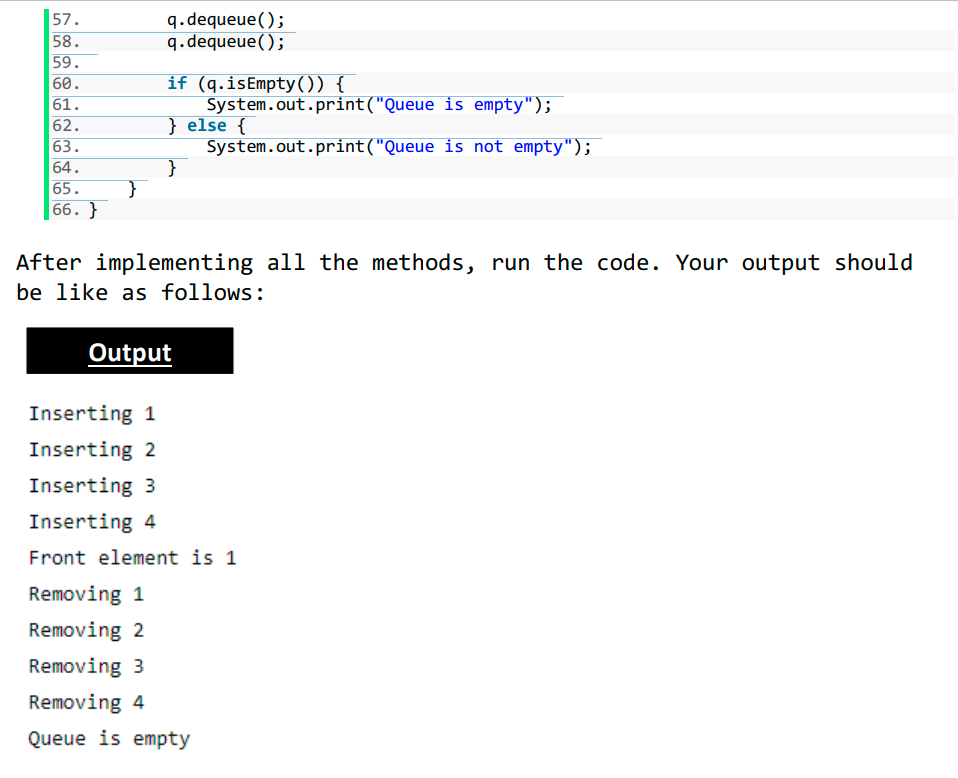


**Task4 Description**









**Solution:**

# **Node and Queue Class**

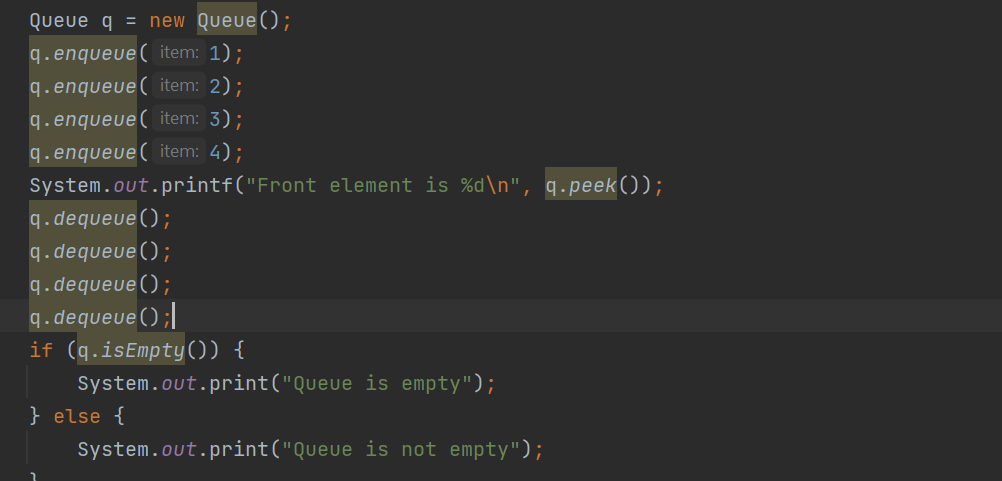
1. // A linked list node
2. class Node2 {
3. int data; // integer data
4. Node2 next; // pointer to the next node
6. public Node2(int data) {
7. // set the data in allocated node and return the node
8. this.data = data;
9. this.next = null;
10. }
11. }
13. class Queue {
14. private static Node2 rear = null;
15. private static Node2 front = null;
17. // Utility function to remove front element from the queue and check for Queue Underflow
18. public static int dequeue() // delete at the beginning
19. {
20. // Write your code here
21. if (isEmpty()) {
22. System.out.println("Queue underflow");
23. return -1;
24. } else {
25. int temp = front.data;
26. System.out.println("Removing " + temp);
27. front = front.next;
28. return temp;
29. }
30. }
32. // Utility function to add an item in the queue
33. public static void enqueue(int item) // insertion at the end
34. {// Write your code here
35. Node2 newNode = new Node2(item);
36. if (isEmpty()) {
37. front = rear = newNode;
38. } else {
39. System.out.println("Inserting " + item);
40. rear.next = newNode;
41. rear = newNode;
42. }
43. }
45. // Utility function to return top element in a queue
46. public static int peek() {// Write your code here
47. if (isEmpty()) {
48. System.out.println("Queue underflow");
49. return -1;
50. } else {
51. int temp = front.data;
52. return temp;
53. }
54. }
56. // Utility function to check if the queue is empty or not
57. public static boolean isEmpty() {
58. // Write your code here
59. return rear == null;
60. }
61. }

# **Queue Class Implementation**

1. public class Queue2Imple {

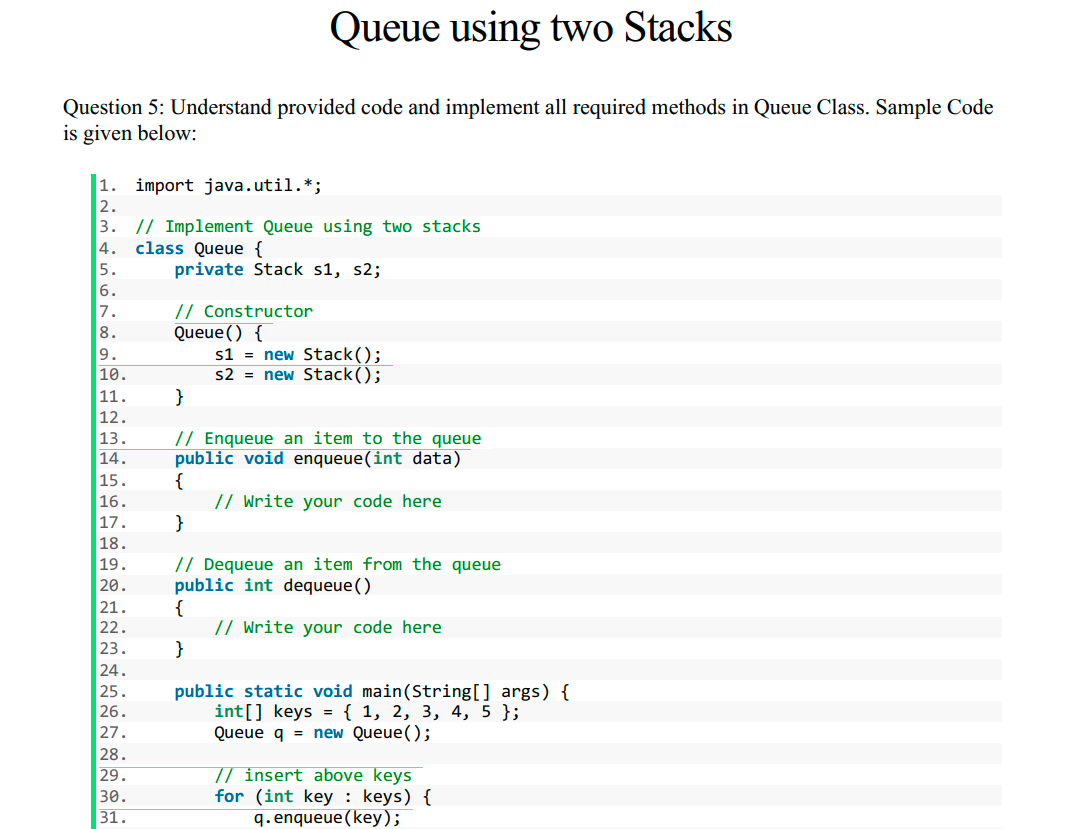
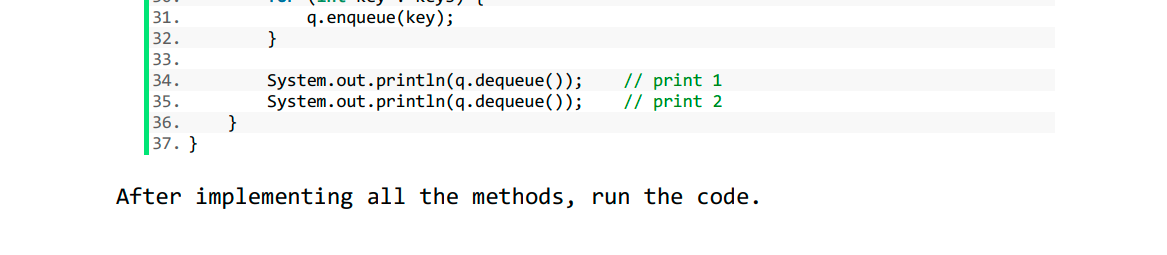
4. public static void main(String[] args) {
5. Queue q = new Queue();
6. q.enqueue(1);
7. q.enqueue(2);
8. q.enqueue(3);
9. q.enqueue(4);
11. System.out.printf("Front element is %d\n", q.peek());
13. q.dequeue();
14. q.dequeue();
15. q.dequeue();
16. q.dequeue();
18. if (q.isEmpty()) {
19. System.out.print("Queue is empty");
20. } else {
21. System.out.print("Queue is not empty");
22. }
23. }}

**Sample Input:**

 **Sample Output**



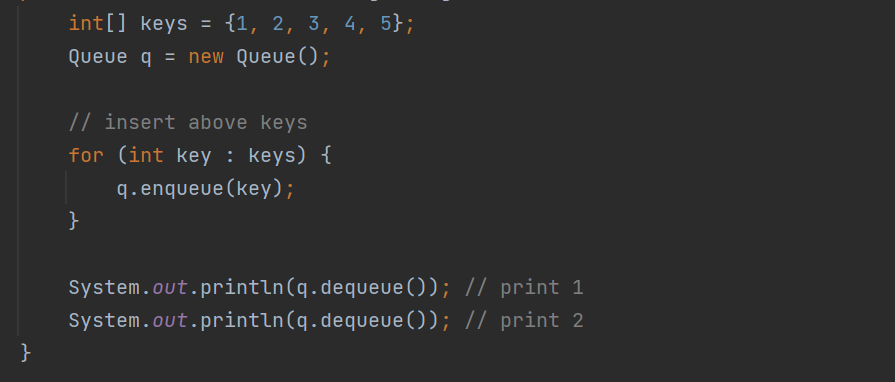
**Task5 Description**

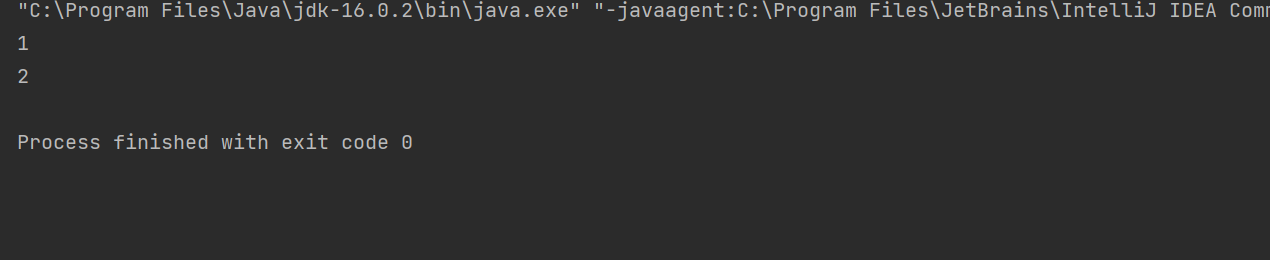


**Solution:**

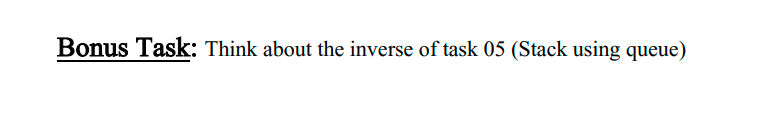
1. import java.util.Stack;
3. public class Queue {
4. Stack<Integer> s1, s2;
6. Queue() {
7. s1 = new Stack();
8. s2 = new Stack();
9. }
11. // Enqueue an item to the queue
12. public void enqueue(int data) {
13. // Write your code here
14. s1.push(data);
15. }
17. // Dequeue an item from the queue
18. public int dequeue() {
19. // Write your code here
20. if (s2.isEmpty()) {
21. while (!s1.isEmpty()) {
22. s2.push(s1.pop());
23. }
24. }
25. return s2.pop();
26. }
28. public static void main(String[] args) {
29. int[] keys = {1, 2, 3, 4, 5};
30. Queue q = new Queue();
32. // insert above keys
33. for (int key : keys) {
34. q.enqueue(key);
35. }
37. System.out.println(q.dequeue()); // print 1
38. System.out.println(q.dequeue()); // print 2
39. }
41. }

**Sample Input:**

 **Sample Output**



**Task6 Description**



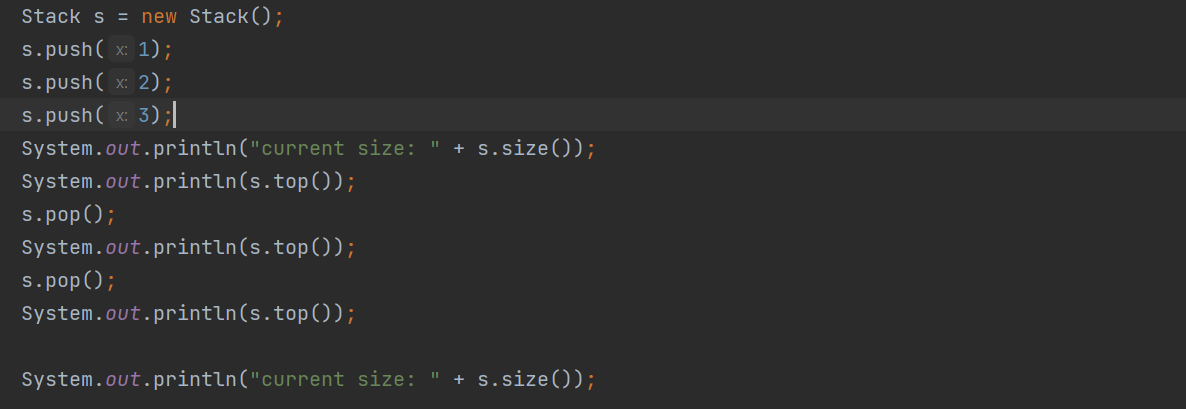
Solution:

1. import java.util.\*;
2. import java.util.Queue;
4. class Stack {
5. // Two inbuilt queues
6. Queue<Integer> q1 = new LinkedList<Integer>();
7. Queue<Integer> q2 = new LinkedList<Integer>();

10. // To maintain current number of
11. // elements
12. int curr\_size;
14. Stack() {
15. curr\_size = 0;
16. }
18. void push(int x) {
19. curr\_size++;
21. // Push x first in empty q2
22. q2.add(x);
24. // Push all the remaining
25. // elements in q1 to q2.
26. while (!q1.isEmpty()) {
27. q2.add(q1.peek());
28. q1.remove();
29. }
31. // swap the names of two queues
32. Queue<Integer> q = q1;
33. q1 = q2;
34. q2 = q;
35. }
37. void pop() {
39. // if no elements are there in q1
40. if (q1.isEmpty())
41. return;
42. q1.remove();
43. curr\_size--;
44. }
46. int top() {
47. if (q1.isEmpty())
48. return -1;
49. return q1.peek();
50. }
52. int size() {
53. return curr\_size;
54. }

57. // driver code
58. public static void main(String[] args) {
59. Stack s = new Stack();
60. s.push(1);
61. s.push(2);
62. s.push(3);
64. System.out.println("current size: " + s.size());
65. System.out.println(s.top());
66. s.pop();
67. System.out.println(s.top());
68. s.pop();
69. System.out.println(s.top());
71. System.out.println("current size: " + s.size());
72. }
73. }

**Sample Input:**

**Sample Output**

