ASSIGNMENT 1 FRONT SHEET

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | Unit 19: Data Structures and Algorithms | | |
| **Submission date** |  | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | PHAN MINH TRI | **Student ID** | GCC18015 |
| **Class** | GCC0701 | **Assessor name** | DUONG TRUNG NGHIA |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  | | **Student’s signature** | PHAN MINH TRI |

# Grading grid

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | M1 | M2 | M3 | D1 | D2 |
|  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| * **Summative Feedback:**  **Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Internal Verifier’s Comments:** | | |
| **IV Signature:** | | |

**ASSIGNMENT 1 BRIEF**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Business** | | |
| **Unit number** | Unit 19: Data Structures and Algorithms | | |
| **Assignment title** | Examine and specify ADT and DSA | | |
| **Academic Year** |  | | |
| **Unit Tutor** |  | | |
| **Issue date** |  | **Submission date** |  |

**IV name and date**

|  |
| --- |
| **Submission Format:** |
| *Format:* The submission is in the form of an individual written report and a presentation. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system.  *Submission* Students are compulsory to submit the assignment in due date and in a way requested by the Tutors. The form of submission will be a soft copy in PDF posted on corresponding course of <http://cms.greenwich.edu.vn/>  *Note:* The Assignment *must* be your own work, and not copied by or from another student or from books etc. If you use ideas, quotes or data (such as diagrams) from books, journals or other sources, you must reference your sources, using the Harvard style. Make sure that you know how to reference properly, and that understand the guidelines on plagiarism. *If you do not, you definitely get fail* |
| **Assignment Brief and Guidance:** |
| **Scenario**: You work as in-house software developer for Softnet Development Ltd, a software body-shop providing network provisioning solutions. Your company is part of a collaborative service provisioning development project and your company has won the contract to design and develop a middleware solution that will interface at the front-end to multiple computer provisioning interfaces including SOAP, HTTP, JML and CLI, and the back-end telecom provisioning network via CLI .  Your account manager has assigned you a special role that is to inform your team about designing and implementing abstract data types. You have been asked to create a presentation for all collaborating partners on how ADTs can be utilised to improve software design, development and testing. Further, you have been asked to write an introductory report for distribution to all partners on how to specify abstract data types and algorithms in a formal notation.  **Tasks**  **Part 1**  You will need to prepare a presentation on how to create a design specification for data structures, explaining the valid operations that can be carried out on the structures using the example of:   1. A queue ADT, a concrete data structure for a First In First out (FIFO) queue. 2. Two sorting algorithms.   **Part 2**  You will need to provide a formal written report that includes the following:  1. Explanation on how to specify an abstract data type using the example of software stack. |

|  |
| --- |
| 1. Explanation of the advantages of encapsulation and information hiding when using an ADT. 2. Discussion of imperative ADTs with regard to object orientation. |

|  |  |  |
| --- | --- | --- |
| **Learning Outcomes and Assessment Criteria** | | |
| **Pass** | **Merit** | **Distinction** |
| **LO1** Examine abstract data types, concrete data structures and algorithms | | **D1** Analyse the operation, using illustrations, of two sorting algorithms, providing an example of each. |
| **P1** Create a design specification for data structures explaining the valid operations that can be carried out on the structures.  **P2** Determine the operations of a memory stack and how it is used to implement function calls in a computer. | **M1** Illustrate, with an example, a concrete data structure for a First In First out (FIFO) queue.  **M2** Compare the performance of two sorting algorithms. |
| **LO2** Specify abstract data types and algorithms in a formal notation | | **D2** Discuss the view that imperative ADTs are a basis for object orientation and, with justification, state whether you agree. |
| **P3** Using an imperative definition, specify the abstract data type for a software stack. | **M3** Examine the advantages of encapsulation and information hiding when using an ADT. |

**P1** **Create a design specification for data structures explaining the valid operations that can be carried out on the structures.**

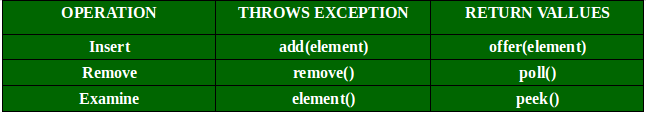
**Queue data structure:**

The Queue functionality is included in the java. util bundle and extends the Set functionality. The queue set is used to keep the items that are about to be handled and to execute many operations, such as addition, elimination, etc. This is the ordered set of objects whose application is restricted to adding elements at the end of the list and removing elements from the beginning of the list, i.e. this follows the FIFO (First-In-First-Out theory). Having a queue interface includes a particular declaration type, and the most common classes are PriorityQueue and LinkedList in Java.



**Methods in Queue:**

1. **add()-** This method is used to add elements at the tail of queue. More specifically, at the last of linked-list if it is used, or according to the priority in case of priority queue implementation.
2. **peek()-** This method is used to view the head of queue without removing it. It returns Null if the queue is empty.
3. **element()-** This method is similar to peek(). It throws *NoSuchElementException* when the queue is empty.
4. **remove()-** This method removes and returns the head of the queue. It throws *NoSuchElementException* when the queue is empty.
5. **poll()-** This method removes and returns the head of the queue. It returns null if the queue is empty.
6. **size()-** This method return the no. of elements in the queue.



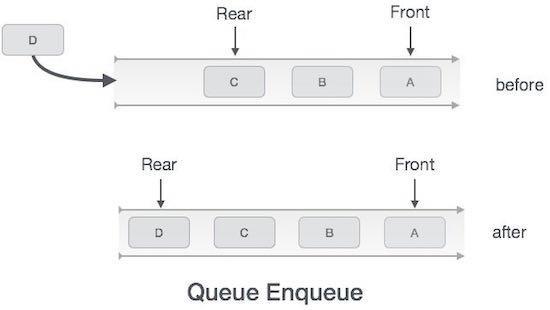
*Enqueue: Adds a line object to the list. If the queue is complete, so it is assumed to be overflow. (Geeksforgeeks, n.d.)

*Enqueue operation in the queue data structure: Step 1: check if the queue is full.

Step 2: If the queue is full, the process fails and exits.

Step 3: If the queue is not full, increase the rear cursor to point to the next free memory location.

Step 4: add data element to the position of the rear cursor pointing in the queue. Step 5: return success.



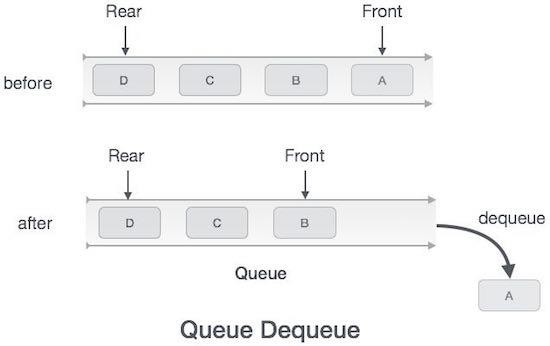
*Dequeue : The Java.util. Dequeue module is a Java.util. Queue module subtype. The Dequeue is connected to a double-ended queue that allows the insertion or removal of elements from either end of the data structure, and can be used as a queue (first-in-first- out / FIFO) or as a stack (last-in-first-out / LIFO). They're quicker than Stack and LinkedList. (Geeksforgeeks, n.d.)

*Dequeue operation in the queue data structure:

Step 1 − Check if the queue is empty.

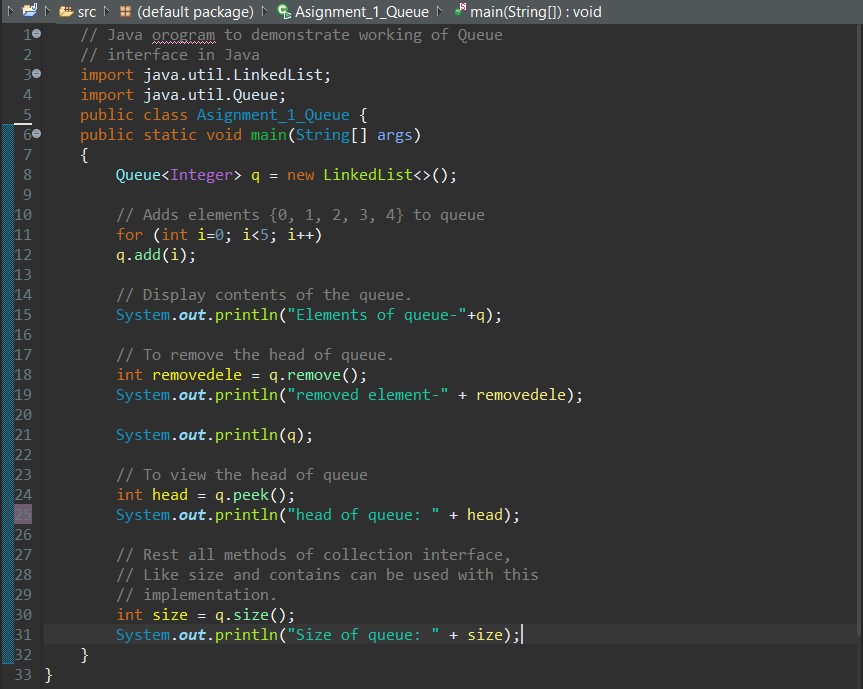
Step 2 − If the queue is empty, produce underflow error and exit.

Step 3 − If the queue is not empty, access the data where front is pointing. Step 4 − Increment front pointer to point to the next available data element. Step 5 − Return success.

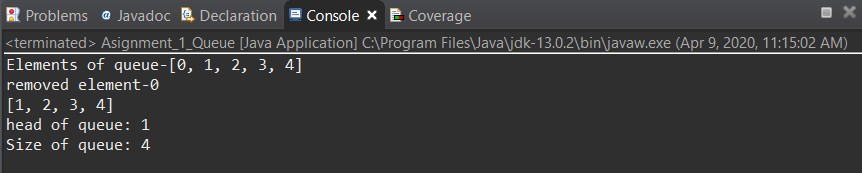


For example:

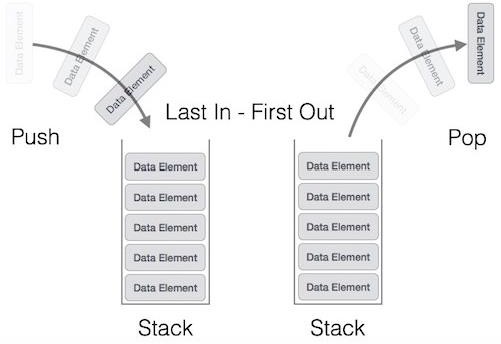
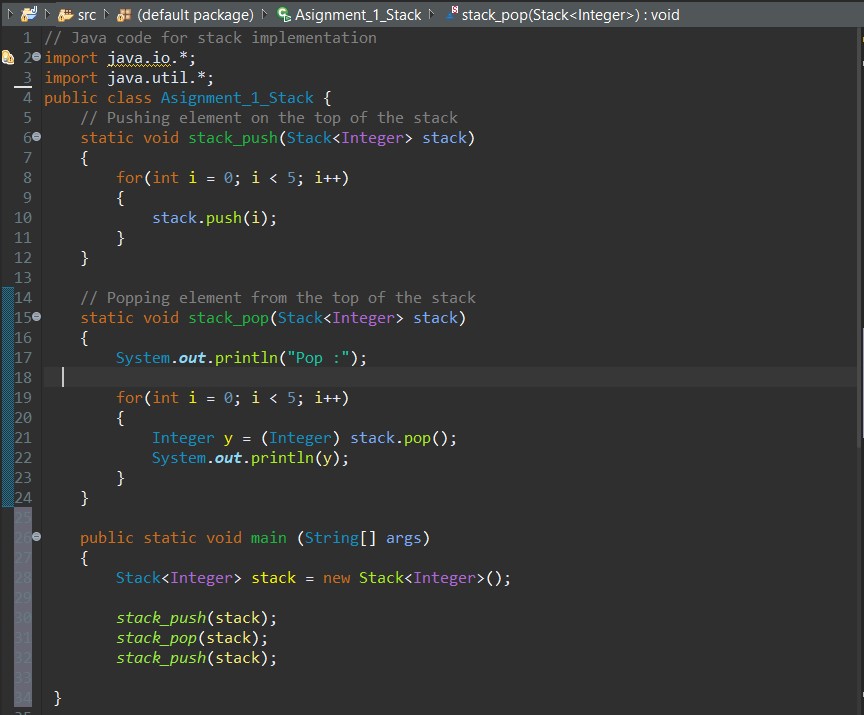
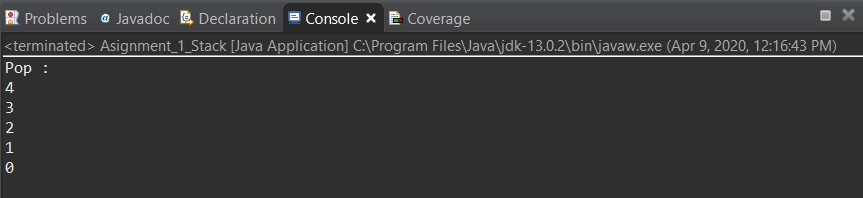
* + Input:



* Output:

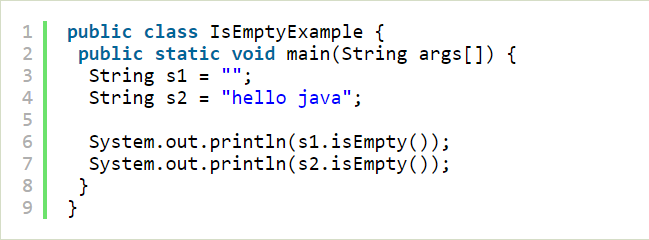


**P2** **Determine the operations of a memory stack and how it is used to implement function calls in a computer.**

* Stack: A stack is an abstract data structure (Abstract Data Type - abbreviated ADT and is often found in every programming language. Called as a stack since it behaves like a real-life stack, like a card deck or a cd stack, etc.
* This function renders the stack a LIFO(Last-In-First-Out) data structure. Here, the inserted (inserted, added) item will be accessed first. In stack terminology, the insert operation is called the PUSH operation and the deletion operation is called the POP operation.
* Stack implementation:
  + Input:
  + Output:
* Methods in Stack class:
* Item push(Object element): moves the element to the top of the stack.
* Pop(): Removes and returns the top portion of the stack. The 'EmptyStackException' exception is thrown when we call pop) (when the calling stack is empty.

**P3: Using an imperative definition, specify the abstract data type for a software stack.**

1. isEmpty: The isEmpty() method when empty string returns true, otherwise returns false. For example:

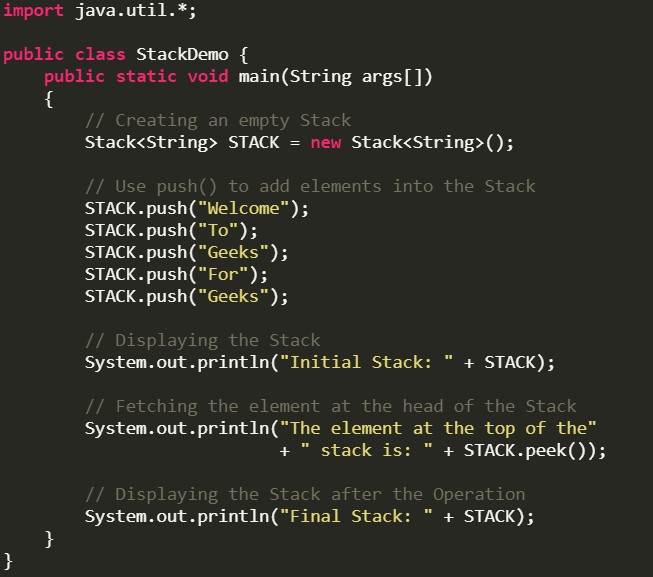
Input:

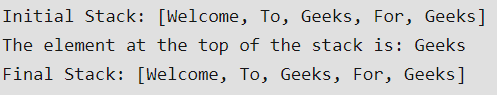
Output:



1. Peek: The java.util.Stack.peek() method in Java is used to retrieve or fetch the first element of the Stack or the element present at the top of the Stack. The element retrieved does not get deleted or removed from the Stack.

For example: Input:

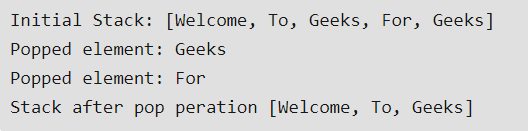


Output:

1. Pop: The Java.util. Stack.pop() method is used to lift an element out of the stack. The variable is popped from the top of the stack and removed from the stack.

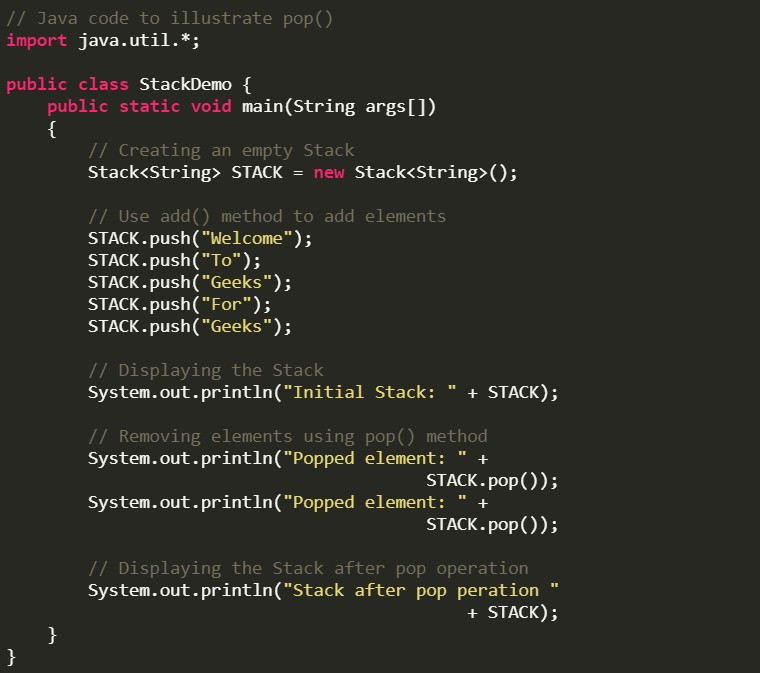
For example, Input:

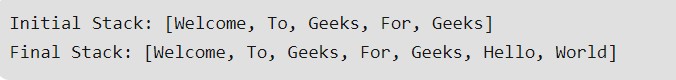


Output:

1. Push: The Java.util. Stack.pop()method is used to lift an element out of the stack. The variable is popped from the top of the stack and removed from the stack.

For example: Input:

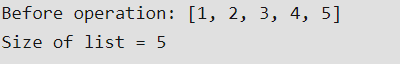


Output:

1. Size: The **size()** method of List interface in Java is used to get the number of elements in this list. That is, this method returns the count of elements present in this list container. For example:

 Input:

Output:



# References

**There are no sources in the current document.**

<https://www.geeksforgeeks.org/queue-data-structure/>

<https://techtalk.vn/>

Techtalk, n.d. *Teachtalk.* [Online]

Available at: https://techtalk.vn/ngan-xep-stack-la-gi.html [Accessed 09 04 2020].

Viettuts, n.d. *Viettuts.* [Online]

Available at: https://viettuts.vn/java-string/phuong-thuc-isempty [Accessed 09 04 2020].