Sherwin Labadan

CS310 Data Structures

Project Assignment 1

A. Package (Program Structure)

- Container
  - LLStack
  - StackQ
  - Node
- Data
  - DataClass
- Driver
  - Driver   \\ driver is the main class
- Interface
  - QueueSpecs
  - StacksSpecs

B. Description

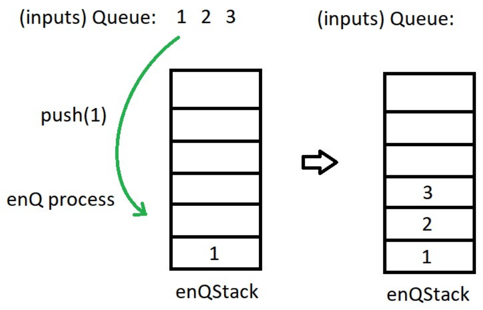
- **LLStack** - This class is in the Container package
  - It implements the interface StackSpecs.

- public LLStack – it initializes the top to null;
- public int getStackSize – it returns the stack size.
- public int setStackSize – it sets the stack size from the data to this.stackSize.
- public getTop – it returns the top to the node.
- public void setTop – get the data and set it to this.top.
- public boolean isEmpty – Boolean method to return if the top is empty or not. If top is null then it return true if not then it returns false.
- public void push – before inserting the data, it checks if the top of the node is empty or not, if its empty, it will insert the data as the top of the node. If the top is not empty, it will make the top as the nextNode and make the current data as the top node. Then it will increment the stackSize by 1.
- public E pop – the method will check if the top of the node is empty or not. If its empty then it will return null. If the its not empty then it put the top of the node into a temporary data

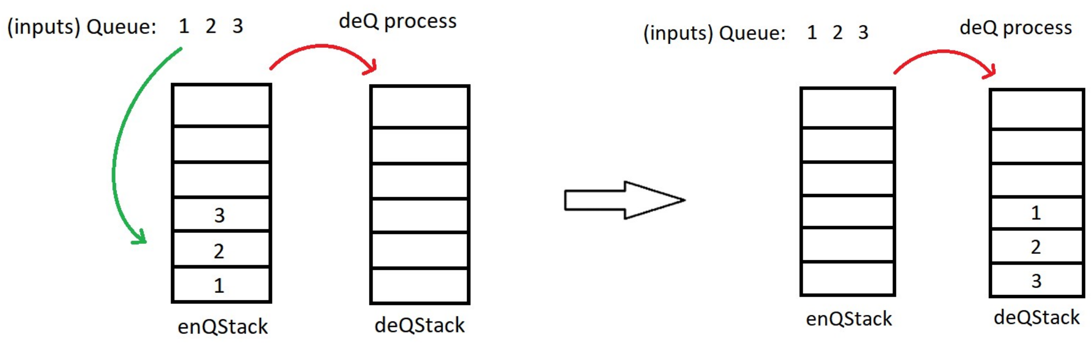
address, then it will make the nextNode as the top node. After, it will return the temporary data address.

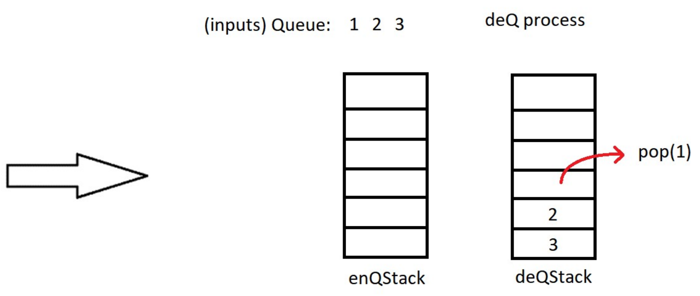
- public E peek – checks if the top node is empty, if empty then it will return null if not then it will return that data.
- public String toString – Prints the queue that was created.

- **StackQ** - This class is in the Container package
  - implements QueueSpecs.
- public StackQ – Constructor for StackQ class
- public int length – it calls the stack size from LLStack and returns it as length.
- public Boolean isEmpty – checks if enQStack and deQstacks are empty. If empty, it returns p true, if not then it return false.
- public void enQ – it calls the push method from LLStack class to push the input into enQStack.



- public E deQ – first it checks if deQStack is empty or not, if there are data in deQstack then it will call pop( ) to remove the top of the stack. If deQStack is empty then it check if enQstack is empty or not. If enQStack is empty while deQstack is also empty, it will print “Stack is empty”. If deQStack is empty while enQStack is not empty, it will keep popping the head of enQStack and pushing it into the deQStack.





- public E peek – calls peek method from LLStack and checks top of the enQStack. If its empty it returns "Stack is empty" else it returns the top of the stack.
- public String toString – it checks enQStack if its empty or not, if not empty then it calls toString from LLStack and prints out enQStack, if empty, then it returns "Stack is empty'
- public String toString1 – it checks deQStack if its empty or not, if not empty then it calls toString from LLStack and prints out deQStack, if empty then it returns "Stack is empty'
- public String to String2 – it prints out the Queue Stack (both enQStack and deQStack).

- Node - This class is the node for the linked list.
- public Node (E data) – it creates the data for the node
- public Node(E data, Node<E> node) – it gets the data and sets it to this.data and sets the node to this.nextNode.
- public E getData – returns data
- public void setData – gets the data and sets it to this.data
- public Node<E> getNextNode – it returns the netNode
- public void setNextNode – gets the next node and sets it to this.nextNode

- DataClass – used in main class to print the inputs
- public DataClass(String dataName, int dataID) gets dataName and dataID and sets it to this.dataName and this.dataID respectively.
- public String getDataName - returns dataName
- public void setDataName(String dataName) – sets dataName to this.dataName
- public int getDataID – returns dataID
- public void setDataID(int dataID) – sets dataID to this.dataID

- public String toString – prints out dataID and dataName

- Driver (main class)
  - Displays user options:
    - a. Enqueue in the Queue
    - b. Dequeue in the Queue
    - c. Peek from the Queue
    - d. Display the Queue
    - e. Display enQStack and deQStack
    - f. Display Size of the Queue
    - g. Exit
  - User is prompted to pick and enters their user choice from a to g.
  - If user entered something else other than option "a to g"
    - It will print "Not Valid, Please Enter a Letter Between a to g" then repeats to ask for user option.
  - If user pick option A, it will be asked to enter an ID number
    - If user enters a non integer, then it will print "Numbers only" and repeats the user options.
    - Once ID is entered, the input will be enQ( ) to the stack then the program will return to the main menu
  - If user pick option B, it will implement the deQ( ) to dequeue the stack
  - If user pick option C, it will print the top of the stack, if its empty the it will print "Stack is empty". After the program will return to the main menu.
  - If user pick option D, it will call toString2 to print the current Queue
  - If user pick option E, it will call both toString and toString2 to print enQStack and deQstack.
  - If user pick option F, it will call the method length and prints the length of the Queue.
  - If user pick option F, then it will print "Goodbye" then the program will exit.