LINEAR ABSTRACT DATA TYPES – LAB 03

EC 4070

DATA STRUCTURES AND ALGORITHMS

NAME : **WIJAYAWARDHANA W.A.H.A.**

REGISTRATION NO. : **2019/E/166**

SEMESTER : **SEMESTER 04**

DATE ASSIGNED : **10 MARCH 2022**

01.
a).

Code:-

```
public class StackOperation {
   int arraySize;
   int[] stackElementArray = new int[arraySize];
   int topValue;
   int newElement;
   boolean stackEmpty;
   boolean stackFull;

   public void StackOperation()
   {
      arraySize = 0;
      topValue = -1;
   }
   public void StackOperation(int arraySize , int[] stackElementArray,int topValue)
   {
      this.arraySize = arraySize;
      this.stackElementArray = stackElementArray;
      this.topValue = topValue;
      stackEmpty = false;
      stackFull = false;
   }
   /*
   public void setStack()
   {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the stack size : ");
      for(int)
   }
 */
   public void isEmpty()
   {
      if(topValue == -1)
      {
         stackEmpty = true;
      }
      else
      {
         stackEmpty = false;
      }

   }
```

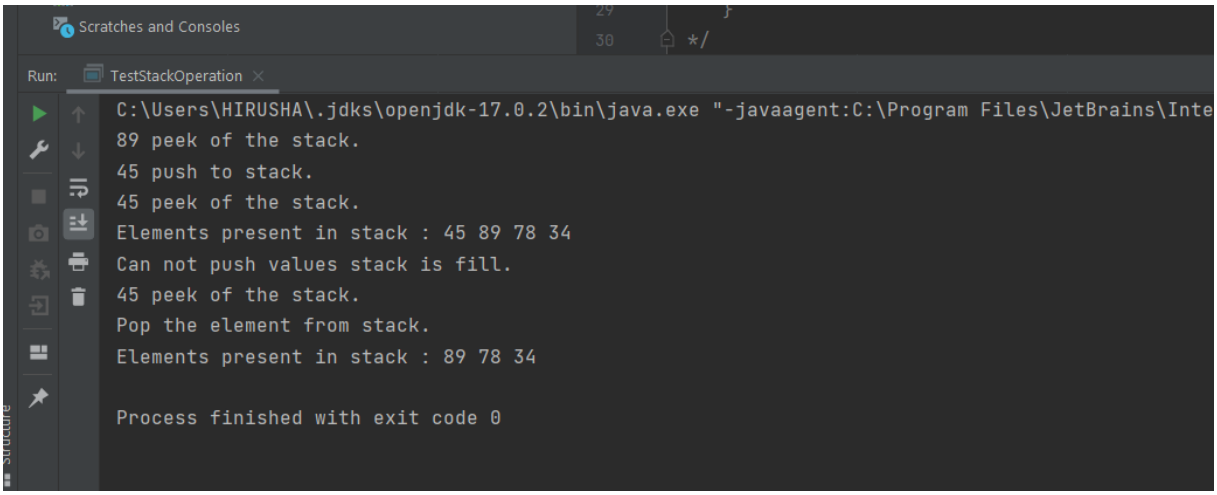
```
public void isFull()
{
   if(topValue == stackElementArray.length)
   {
      stackFull =  true;
   }
   else
   {
      stackFull =  false;
   }
}

public void peek()
{
   isEmpty();
   if(stackEmpty == true)
   {
      System.out.println("Stack is empty.");
   }
   else
   {
      System.out.println(stackElementArray[topValue-1] + " peek of the stack.");
   }
}

public void push(int newElement)
{
   this.newElement = newElement;
   isFull();
   if(stackFull == true)
   {
      System.out.println("Can not push values stack is fill.");
   }
   else
   {
      stackElementArray[topValue] = newElement;
      topValue++;
      System.out.println(newElement + " push to stack.");
   }
}

public void pop()
{
   if(stackEmpty == true)
   {
      System.out.println("Stack is empty can not pop values.");
   }
   else
   {
```

```
            stackElementArray[topValue-1] = 0;
            topValue--;
            System.out.println("Pop the element from stack.");
        }
    }

    public void printElement()
    {
        System.out.print("Elements present in stack : ");
        for(int i = topValue-1; i>=0; i--)
        {
            System.out.print(stackElementArray[i]+ " ");
        }
        System.out.println();
    }

}
```

Main Class Code:-

```
public class TestStackOperation {
    public static void main(String[] args) {
        StackOperation newObject = new StackOperation();
        //System.out.println("");
        int arraySize = 4;
        int[] elementArray = new int[]{34, 78, 89, 0};
        newObject.StackOperation(arraySize, elementArray, 3);
        newObject.peek();
        newObject.push(45);
        newObject.peek();
        newObject.printElement();
        newObject.push(66);
        newObject.peek();
        newObject.pop();
        newObject.printElement();
    }
}
```

Outputs:-



b). 1.

Code:-

```
import java.util.Scanner;
public class ReverseWord {
   String word;
   Scanner scanner = new Scanner(System.in);
   public void ReverseWord()
   {
      word = "WORD";
   }
   public void ReverseWord(String word)
   {
      this.word = word;
      setCharactersArray(word);
   }
   public void setWord()
   {
      System.out.print("Enter the word : ");
      word = scanner.nextLine();
      setCharactersArray(word);
   }
   public void setCharactersArray(String word)
   {
      char[] characters = new char[word.length()];
      for(int i =0; i<word.length(); i++)
      {
         characters[i] = word.charAt(i);
      }
```

```
        printWord(characters);
    }

    public void printWord(char[] characters)
    {
        for(int j = characters.length-1; j>=0; j--)
        {
            System.out.print(characters[j]);
        }
    }
}
```

Main Class Code:-

```
public class TestReverseWord {
    public static void main(String[] args) {
        ReverseWord newObject = new ReverseWord();
        newObject.setWord();
    }
}
```

Output:-

2.

Code:-

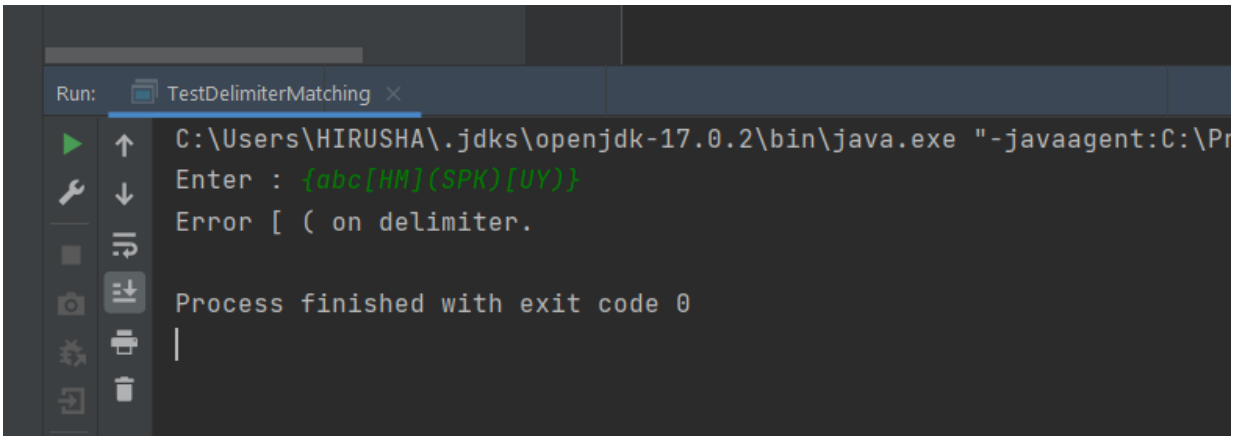
```
import java.util.Scanner;
import java.util.ArrayList;
public class DelimitersMatching {
   String delimiter;
   Scanner scanner = new Scanner(System.in);
   public void DelimitersMatching()
   {
      delimiter = " ";
   }
   public void setDelimiter()
   {
      System.out.print("Enter : ");
      delimiter = scanner.nextLine();
   }

   public void setCharacters()
   {
      int delimiterLength = delimiter.length();
      ArrayList<Character> myList = new ArrayList<Character>();
      char characters = '+';
      int index = -1;
      for(int i =0; i<delimiter.length();i++)
      {
         characters = delimiter.charAt(i);
         if((characters == '{')||(characters == '[')||(characters == '('))
         {
            index++;
            myList.add(index,characters);

         }
         else if((characters == '}')||(characters == ']')||(characters == ')'))
         {
            if(characters == '}')
               characters = '{';
            else if(characters == ')')
               characters = '(';
            else
               characters = '[';
            if(myList.get(index) != characters)
            {
               System.out.println("Error "+myList.get(index) +" "+characters + " on delimiter.");
               return;
            }
            else
```

```
            {
                index--;
            }
        }
    }
    System.out.println("Delimiter matching properly.");
  }

}
```

Main Class Code:-

```
public class TestDelimiterMatching {
  public static void main(String[] args) {
    DelimitersMatching newObject = new DelimitersMatching();
    newObject.setDelimiter();
    newObject.setCharacters();
  }
}
```

Output:-



02.

a).

Code:-

```
import java.util.Scanner;

public class QueuesOperation {
   int queuesFront;
   int queuesRear;
   int arraySize;
   int[] queuesElement = new int[arraySize];
   Scanner scanner = new Scanner(System.in);
   public void QueuesOperation()
   {
      queuesRear = -1;
      queuesFront = -1;
   }

   public void QueuesOperation(int[] queuesElement , int arraySize, int rearValue)
   {
      this.queuesElement = queuesElement;
      this.arraySize = arraySize;
      queuesRear = rearValue;
      queuesFront = 0;
   }

   public void setQueues()
   {
      System.out.println("Enter size : ");
      arraySize = scanner.nextInt();
      boolean queuesEmpty = isEmpty();
      if(queuesEmpty == true)
      {
         System.out.println("Queues is empty.");
      }
      for(int i = queuesRear; i < arraySize; i++)
      {
         System.out.print("Enter element : ");
         queuesElement[i] = scanner.nextInt();
      }
      System.out.println("Queues is full.");
   }
   public boolean isEmpty()
   {
      if(queuesRear == -1)
```

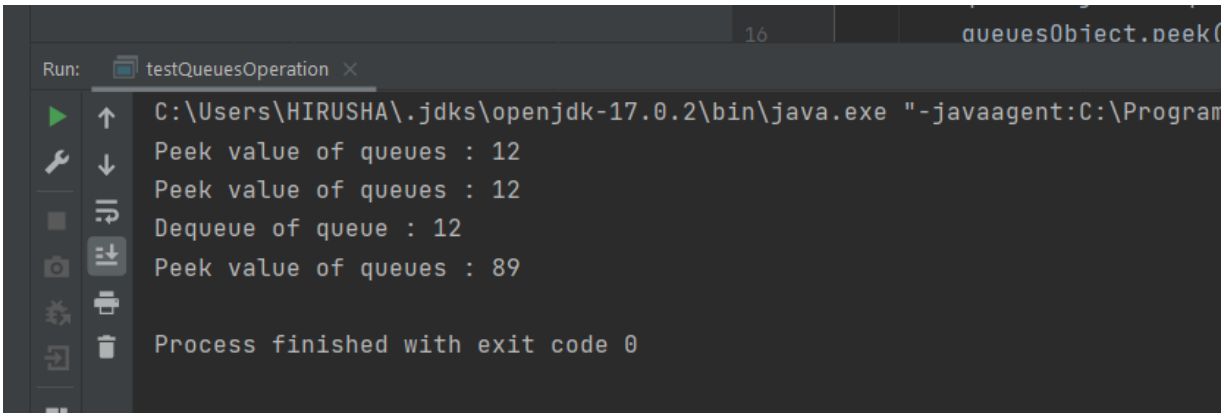
```
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public void peek()
    {
        boolean queuesEmpty = isEmpty();
        if(queuesEmpty == true)
        {
            System.out.println("Queues is empty.");
        }
        else
        {
            System.out.println("Peek value of queues : " + queuesElement[queuesFront]);
        }
    }

    public boolean isFull()
    {
        if(queuesRear == queuesElement.length-1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public void enqueue(int newElement)
    {
        boolean queuesFull = isFull();
        if(queuesFull == true)
        {
            System.out.println("Queues is full.");
        }
        else
        {
            queuesElement[queuesRear-1] = newElement;
            queuesRear++;
        }
    }

    public void dequeue()
    {
```

```
        boolean queuesEmpty = isEmpty();
        if(queuesEmpty == true)
        {
            System.out.println("Queues is empty.");
        }
        else
        {
            System.out.println("Dequeue of queue : "+queuesElement[queuesFront]);
            queuesFront++;
        }
    }
}
```

Main Class Code:-

```
public class testQueuesOperation {
    public static void main(String[] args) {
        QueuesOperation queuesObject = new QueuesOperation();
        int[] queuesArray = new int[10];
        queuesArray[0] = 12;
        queuesArray[1] = 89;
        queuesArray[2] = 55;
        queuesArray[3] = 69;
        queuesArray[4] = 33;
        queuesArray[5] = 84;
        queuesObject.QueuesOperation(queuesArray, 10,5);
        queuesObject.peek();
        queuesObject.enqueue(55);
        queuesObject.peek();
        queuesObject.dequeue();
        queuesObject.peek();
    }
}
```

Output:-



03.

Code:-

```
import java.util.ArrayList;
public class LinkList {
   ArrayList<Integer> linkListArrayList = new ArrayList<Integer>();
   int linkListIndex ;
   public void LinkList()
   {

   }
   public void appendNewNode(int newElement)
   {
      linkListArrayList.add(linkListIndex,newElement);
      linkListIndex++;
   }
   public void prependNewNode(int newElement)
   {
      for(int i =linkListIndex; i>=0; i--)
      {
         linkListArrayList.add(linkListIndex+1,linkListArrayList.get(linkListIndex));
         linkListIndex--;
      }
   }
   public void deleteAtStart()
   {
      for(int i = 0; i<linkListIndex;i++)
      {
         linkListArrayList.add(i,linkListArrayList.get(i+1));
      }
   }

   public void deleteAtSpecificPosition(int indexForDelete)
   {
      for(int i = indexForDelete; i<linkListIndex; i++)
      {
         linkListArrayList.add(i,linkListArrayList.get(i+1));
      }
   }

}
```