f) Assume following StackX class given in Figure 2 is implemented. You need to introduce a new method to StackX class to reverse individual words of a given string. Write a Java code segment for ***String reverseString(String input)*** method to take a string input from the user. Use only a stack data structure and obtain the following output. (
Input: "Hello how are you" ***Output:*** "olleH woh era uoy"

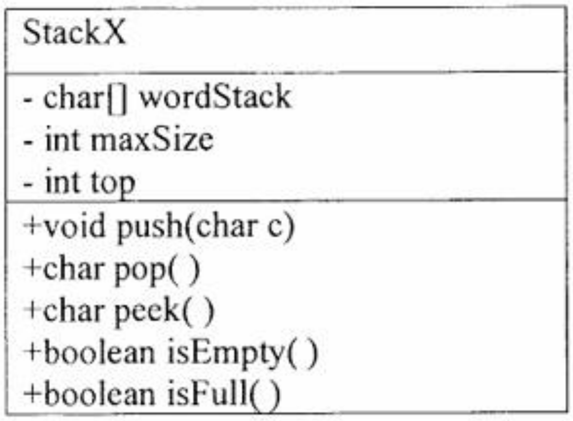
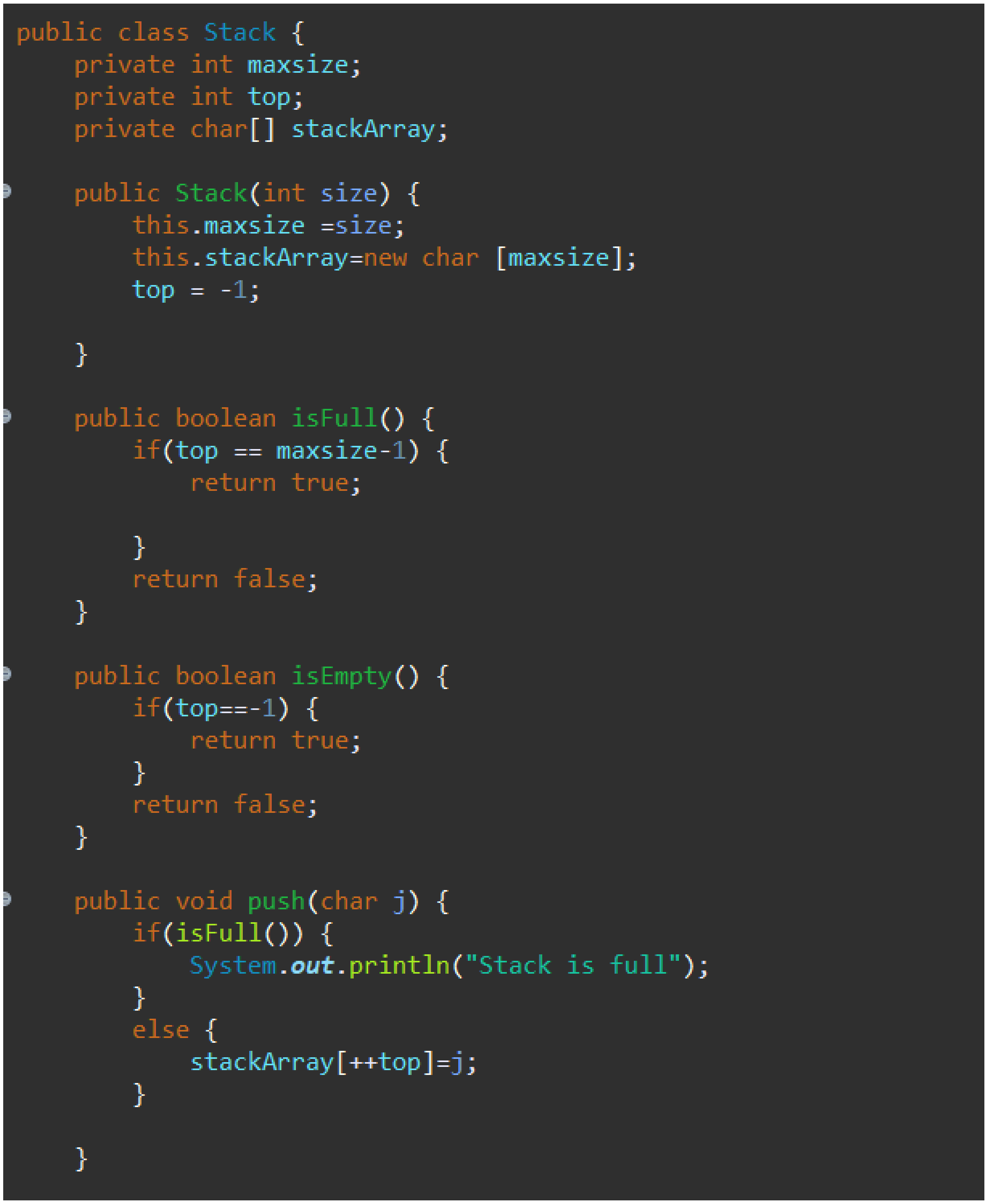
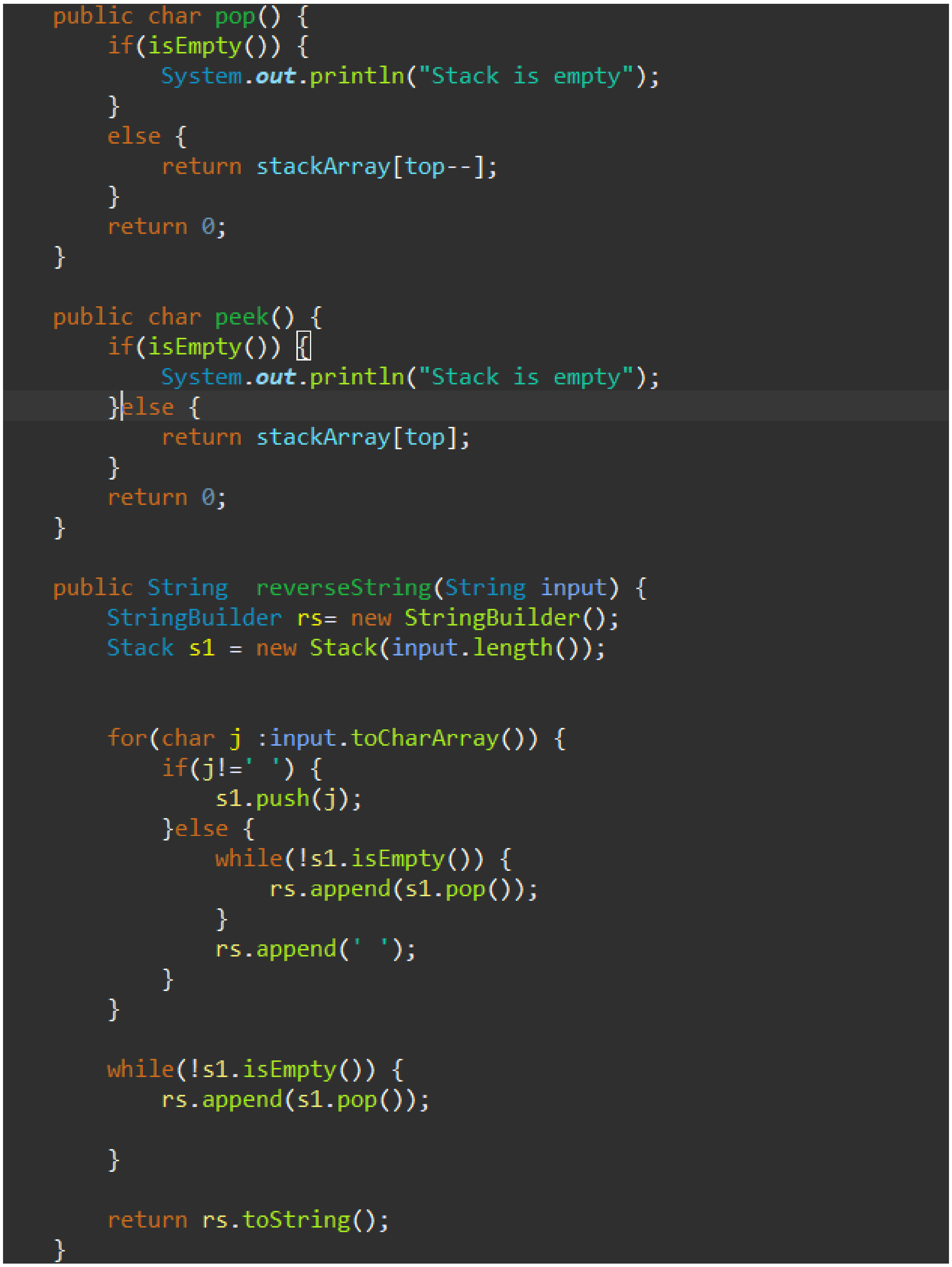


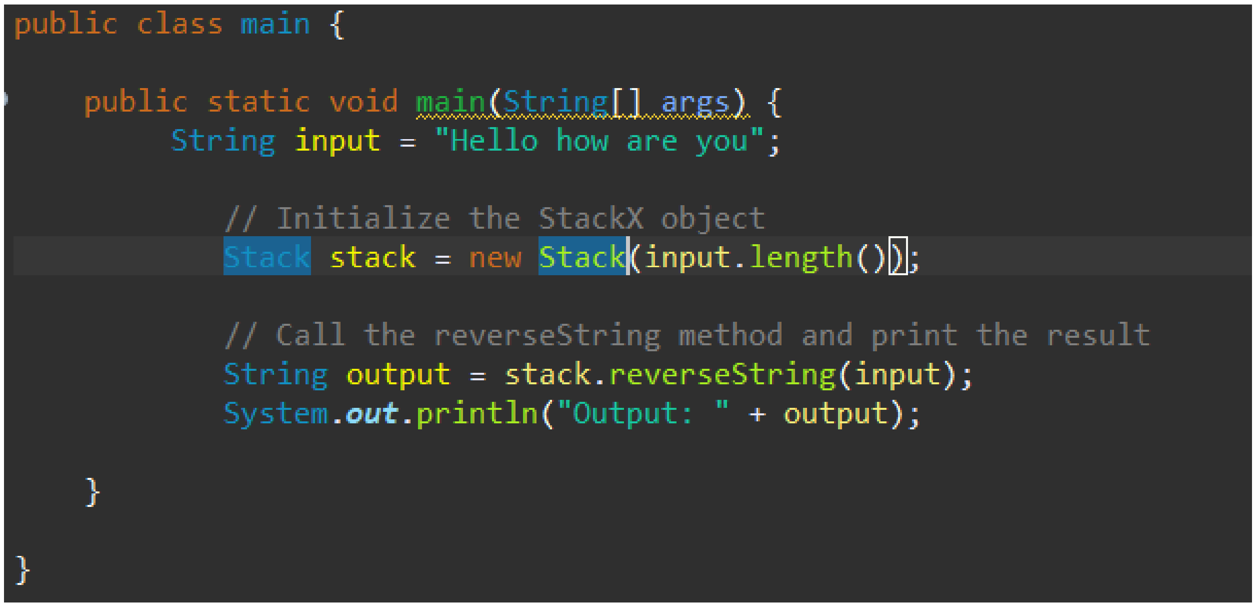
Figure 2. StackX Class

g) Write the main program to call reverseString(String input) and display the reversed string. ***Hint: You can predefine the size of the stack.*** (

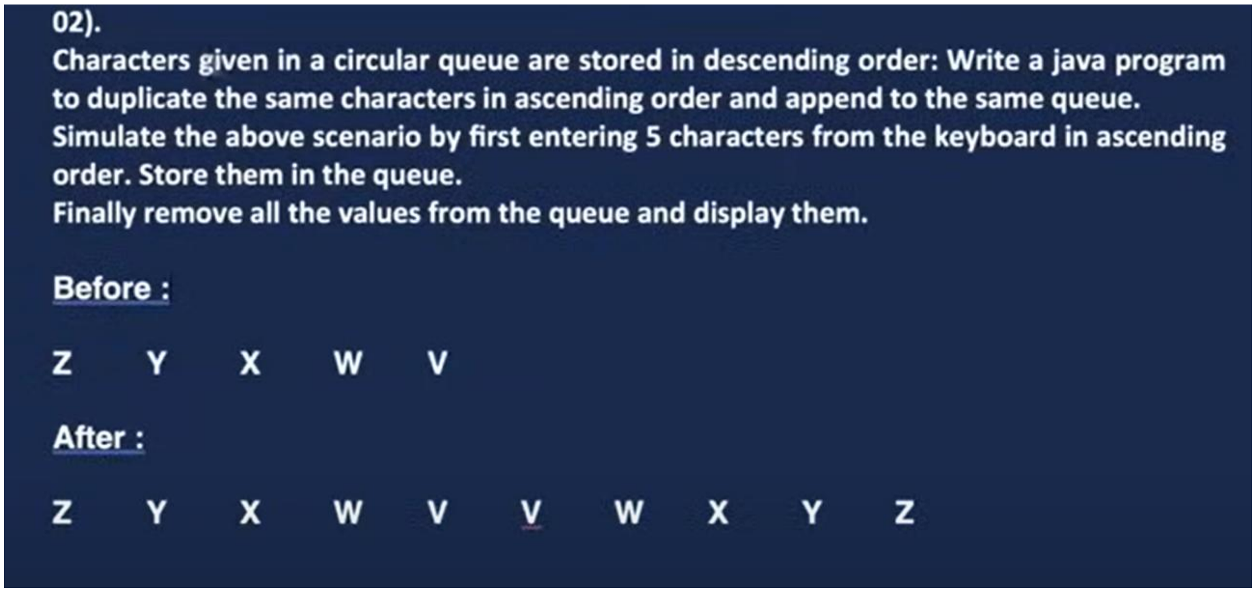


```java
public class Stack {
    private int maxsize;
    private int top;
    private char[] stackArray;

    public Stack(int size) {
        this.maxsize =size;
        this.stackArray=new char [maxsize];
        top = -1;

    }

    public boolean isFull() {
        if(top == maxsize-1) {
            return true;

        }
        return false;
    }

    public boolean isEmpty() {
        if(top==-1) {
            return true;
        }
        return false;
    }

    public void push(char j) {
        if(isFull()) {
            System.out.println("Stack is full");
        }
        else {
            stackArray[++top]=j;
        }

    }
```


```java
    public char pop() {
        if(isEmpty()) {
            System.out.println("Stack is empty");
        }
        else {
            return stackArray[top--];
        }
        return 0;
    }

    public char peek() {
        if(isEmpty()) {
            System.out.println("Stack is empty");
        }else {
            return stackArray[top];
        }
        return 0;
    }

    public String  reverseString(String input) {
        StringBuilder rs= new StringBuilder();
        Stack s1 = new Stack(input.length());


        for(char j :input.toCharArray()) {
            if(j!=' ') {
                s1.push(j);
            }else {
                while(!s1.isEmpty()) {
                    rs.append(s1.pop());
                }
                rs.append(' ');
            }
        }

        while(!s1.isEmpty()) {
            rs.append(s1.pop());

        }

        return rs.toString();
    }
```



```java
public class main {

    public static void main(String[] args) {
         String input = "Hello how are you";

            // Initialize the StackX object
            Stack stack = new Stack(input.length());

            // Call the reverseString method and print the result
            String output = stack.reverseString(input);
            System.out.println("Output: " + output);

    }

}
```



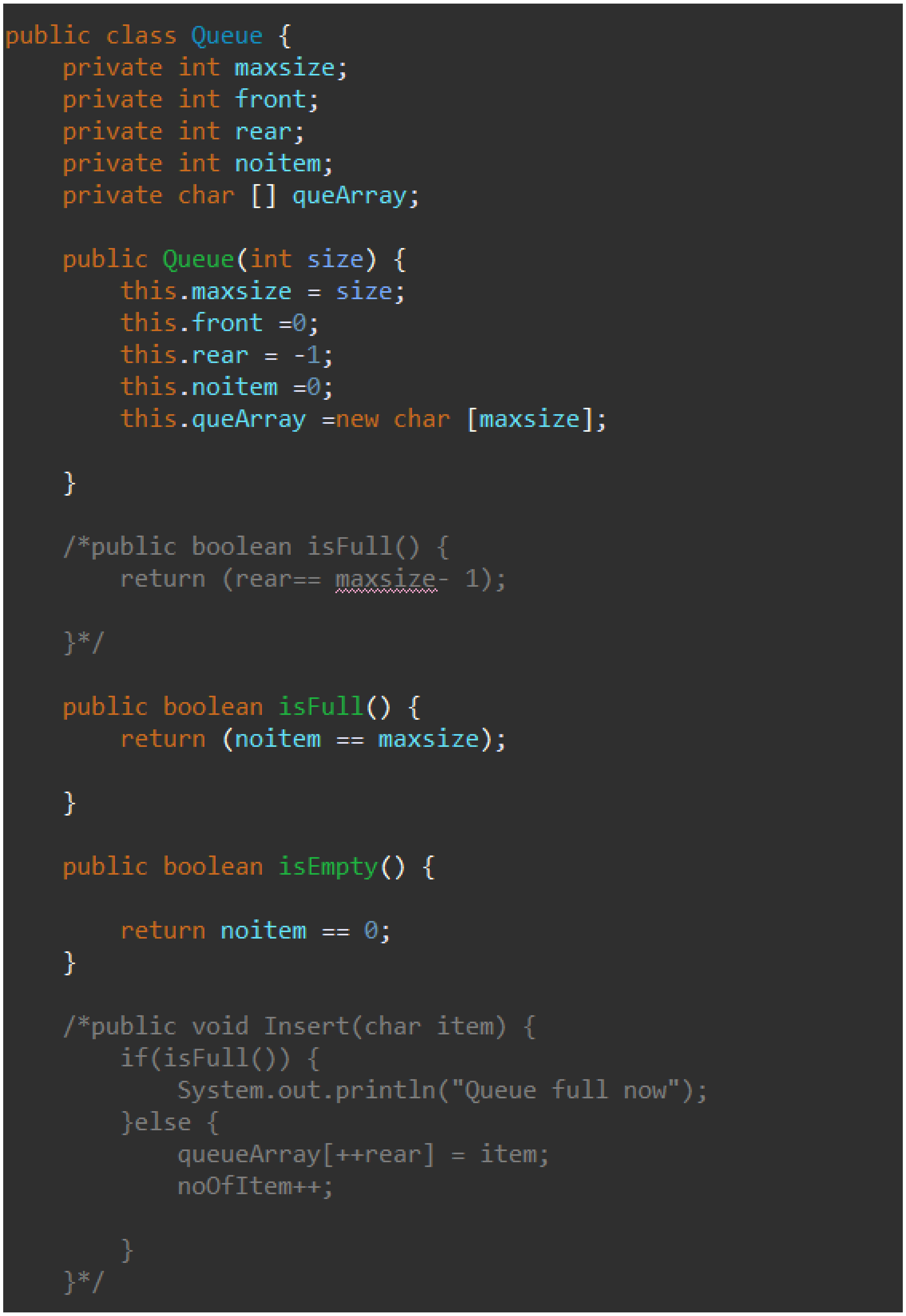
02).
Characters given in a circular queue are stored in descending order: Write a java program to duplicate the same characters in ascending order and append to the same queue.
Simulate the above scenario by first entering 5 characters from the keyboard in ascending order. Store them in the queue.
Finally remove all the values from the queue and display them.

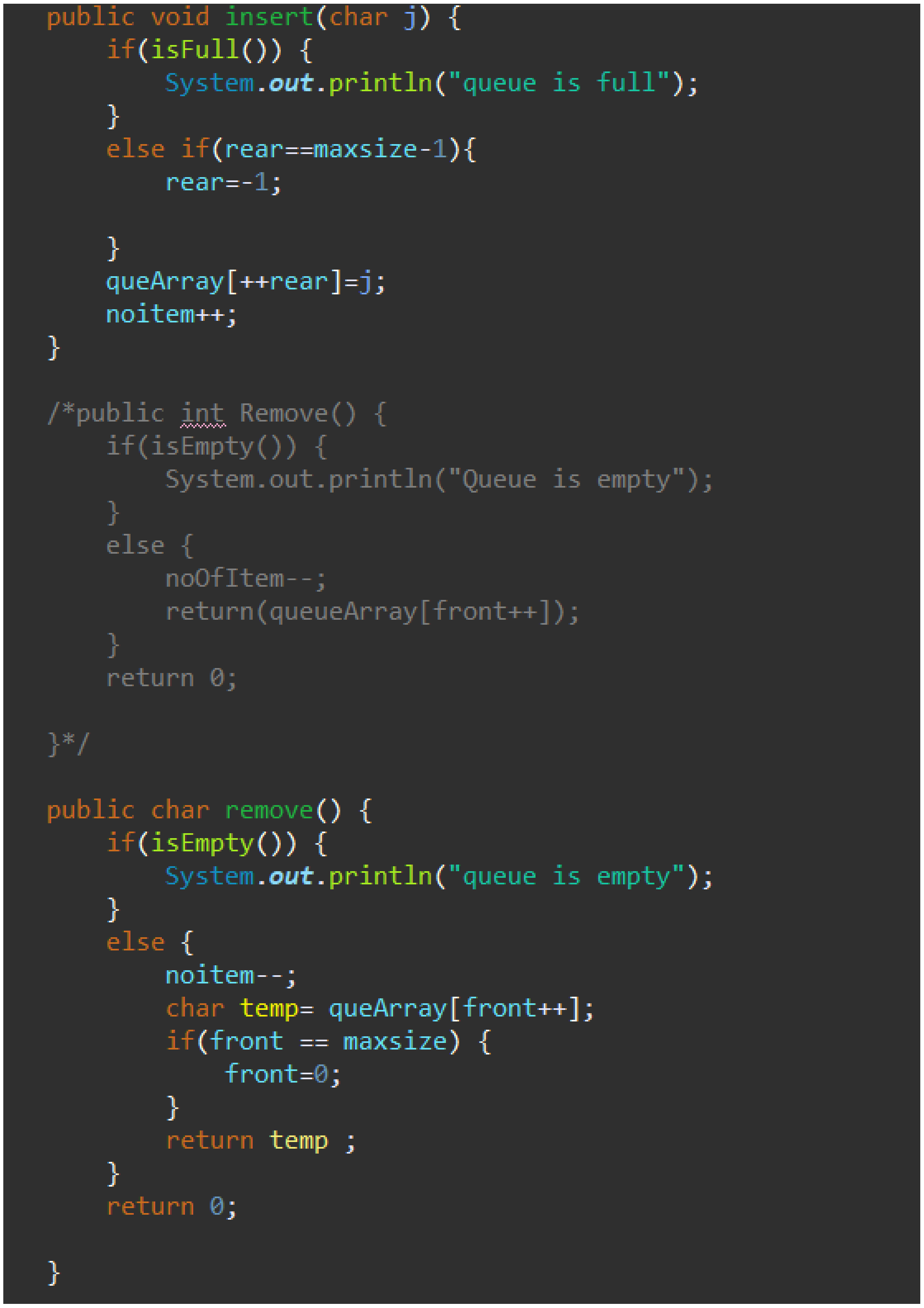
Before :

Z Y X W V

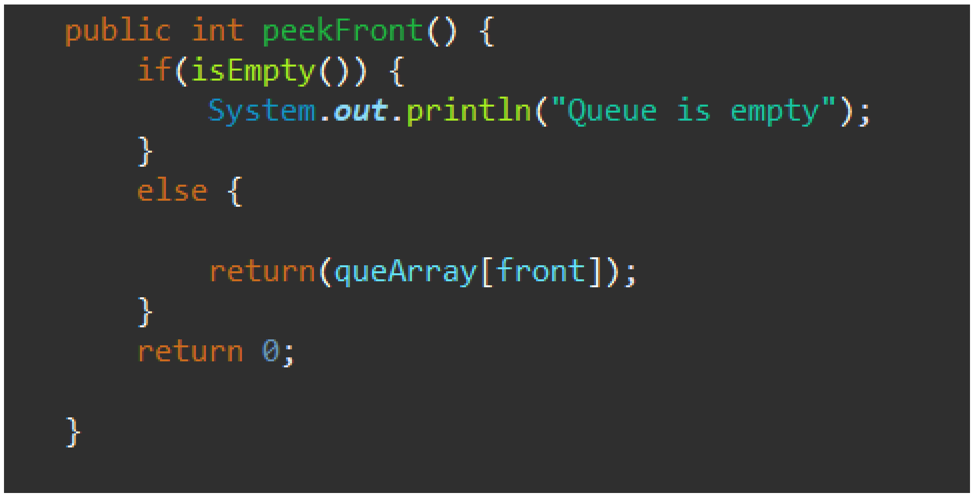
After :

Z Y X W V V W X Y Z

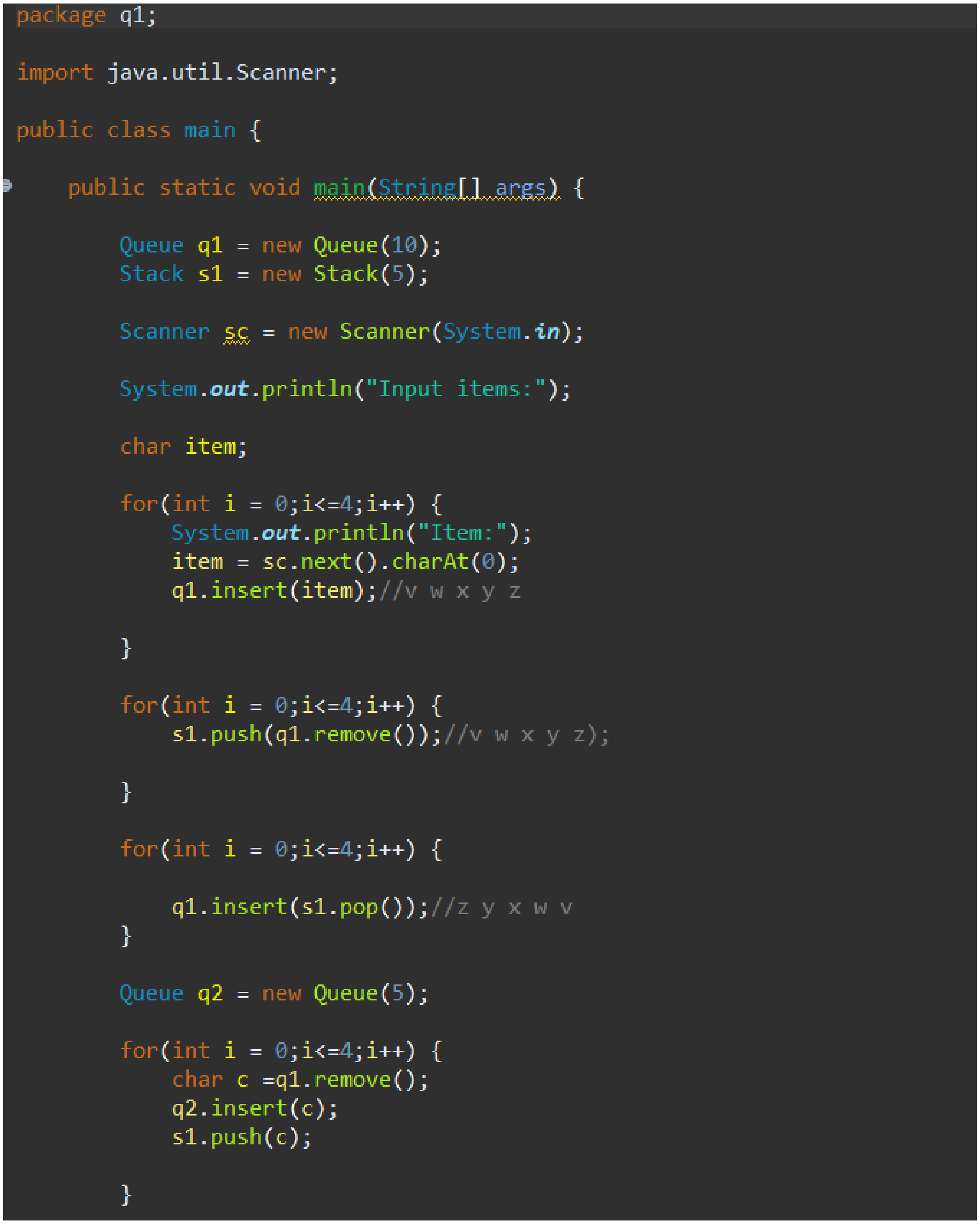

```
public class Queue {
    private int maxsize;
    private int front;
    private int rear;
    private int noitem;
    private char [] queArray;

    public Queue(int size) {
        this.maxsize = size;
        this.front =0;
        this.rear = -1;
        this.noitem =0;
        this.queArray =new char [maxsize];

    }

    /*public boolean isFull() {
        return (rear== maxsize- 1);

    }*/

    public boolean isFull() {
        return (noitem == maxsize);

    }

    public boolean isEmpty() {

        return noitem == 0;
    }

    /*public void Insert(char item) {
        if(isFull()) {
            System.out.println("Queue full now");
        }else {
            queueArray[++rear] = item;
            noOfItem++;

        }
    }*/
```

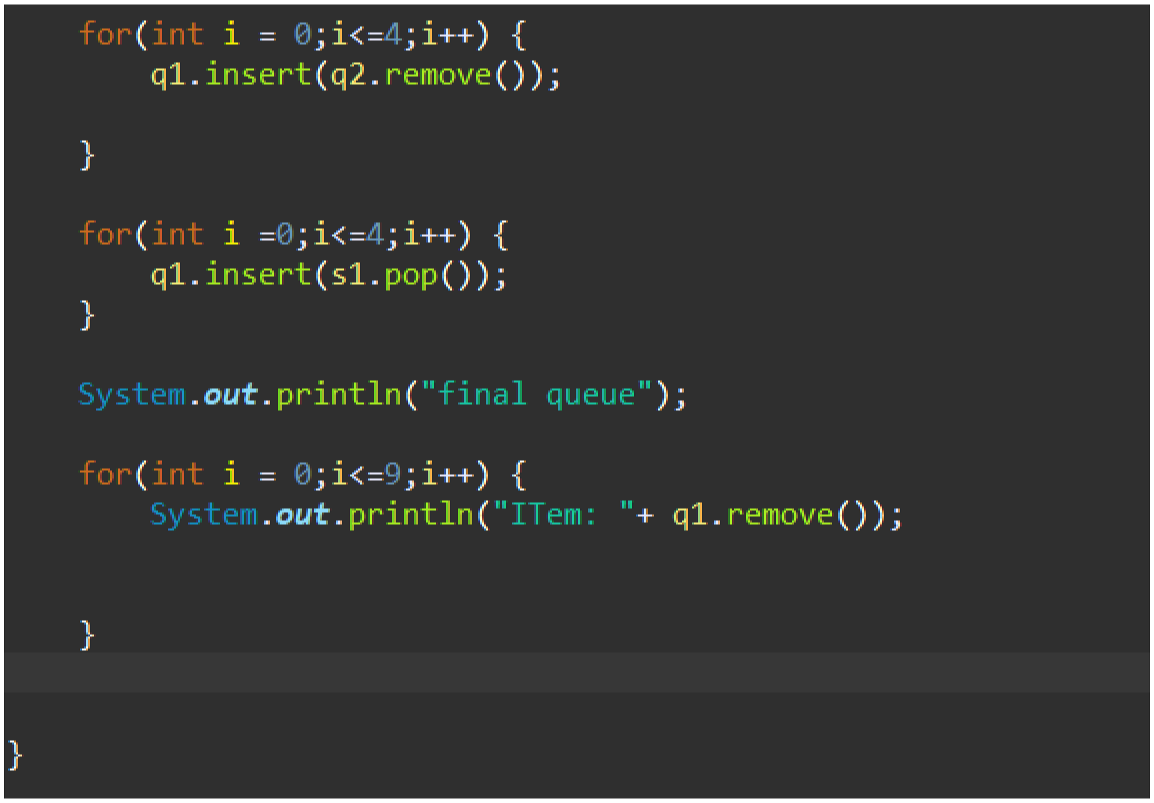


```
public void insert(char j) {
    if(isFull()) {
        System.out.println("queue is full");
    }
    else if(rear==maxsize-1){
        rear=-1;

    }
    queArray[++rear]=j;
    noitem++;
}

/*public int Remove() {
    if(isEmpty()) {
        System.out.println("Queue is empty");
    }
    else {
        noOfItem--;
        return(queueArray[front++]);
    }
    return 0;

}*/

public char remove() {
    if(isEmpty()) {
        System.out.println("queue is empty");
    }
    else {
        noitem--;
        char temp= queArray[front++];
        if(front == maxsize) {
            front=0;
        }
        return temp ;
    }
    return 0;

}
```



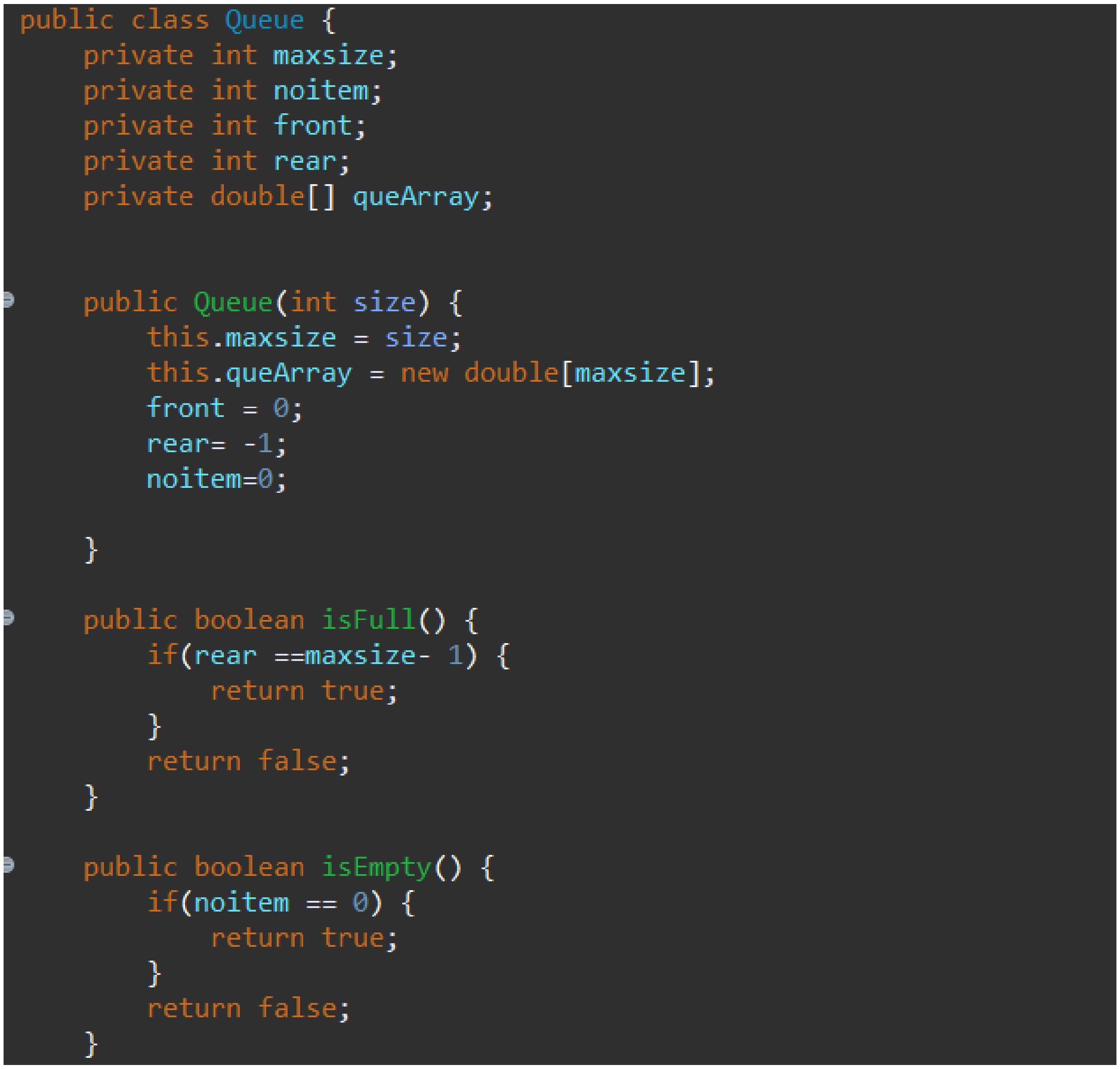
```
public int peekFront() {
    if(isEmpty()) {
        System.out.println("Queue is empty");
    }
    else {

        return(queArray[front]);
    }
    return 0;

}
```


```java
package q1;

import java.util.Scanner;

public class main {

    public static void main(String[] args) {

        Queue q1 = new Queue(10);
        Stack s1 = new Stack(5);

        Scanner sc = new Scanner(System.in);

        System.out.println("Input items:");

        char item;

        for(int i = 0;i<=4;i++) {
            System.out.println("Item:");
            item = sc.next().charAt(0);
            q1.insert(item);//v w x y z

        }

        for(int i = 0;i<=4;i++) {
            s1.push(q1.remove());//v w x y z);

        }

        for(int i = 0;i<=4;i++) {

            q1.insert(s1.pop());//z y x w v
        }

        Queue q2 = new Queue(5);

        for(int i = 0;i<=4;i++) {
            char c =q1.remove();
            q2.insert(c);
            s1.push(c);

        }
```

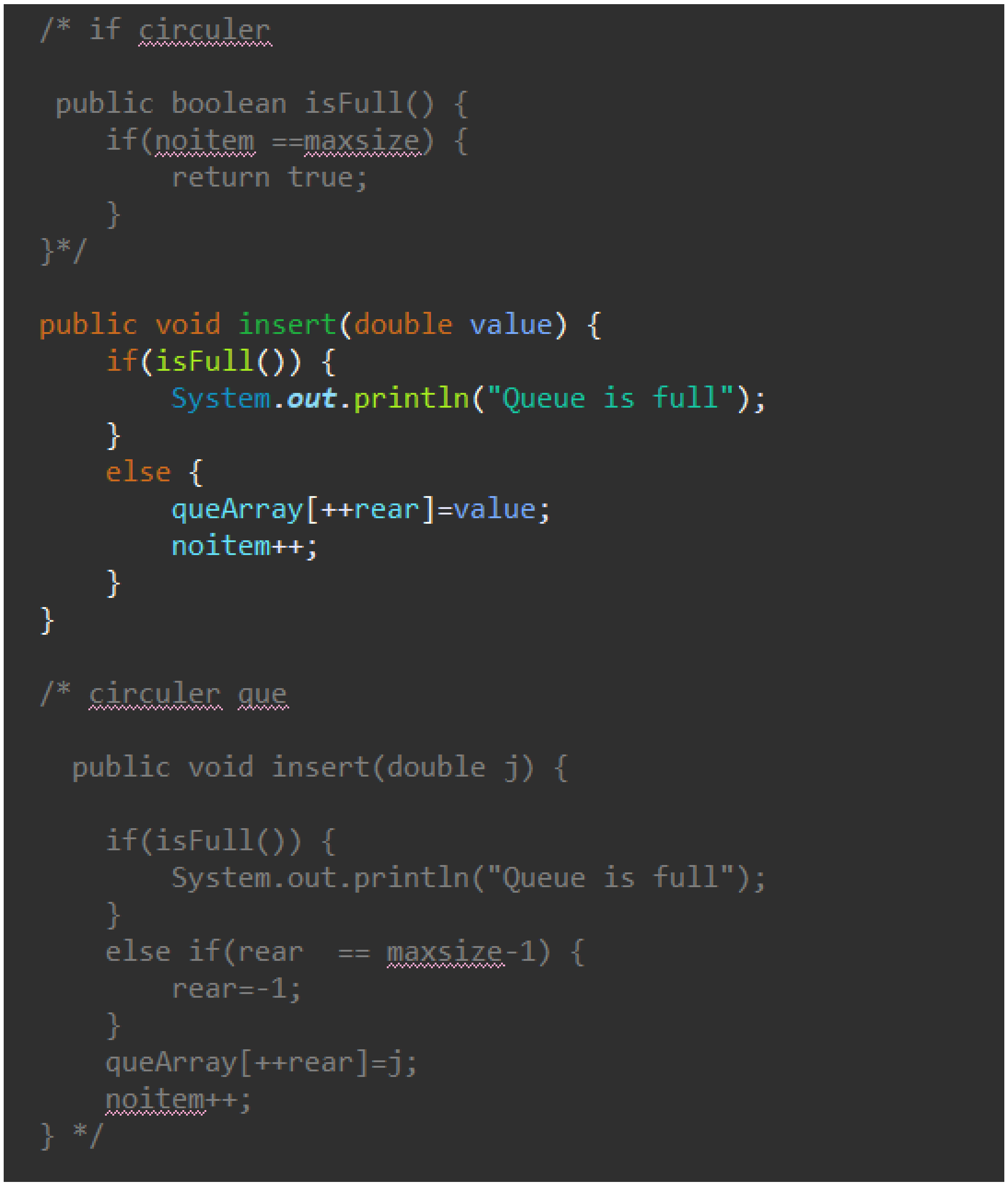


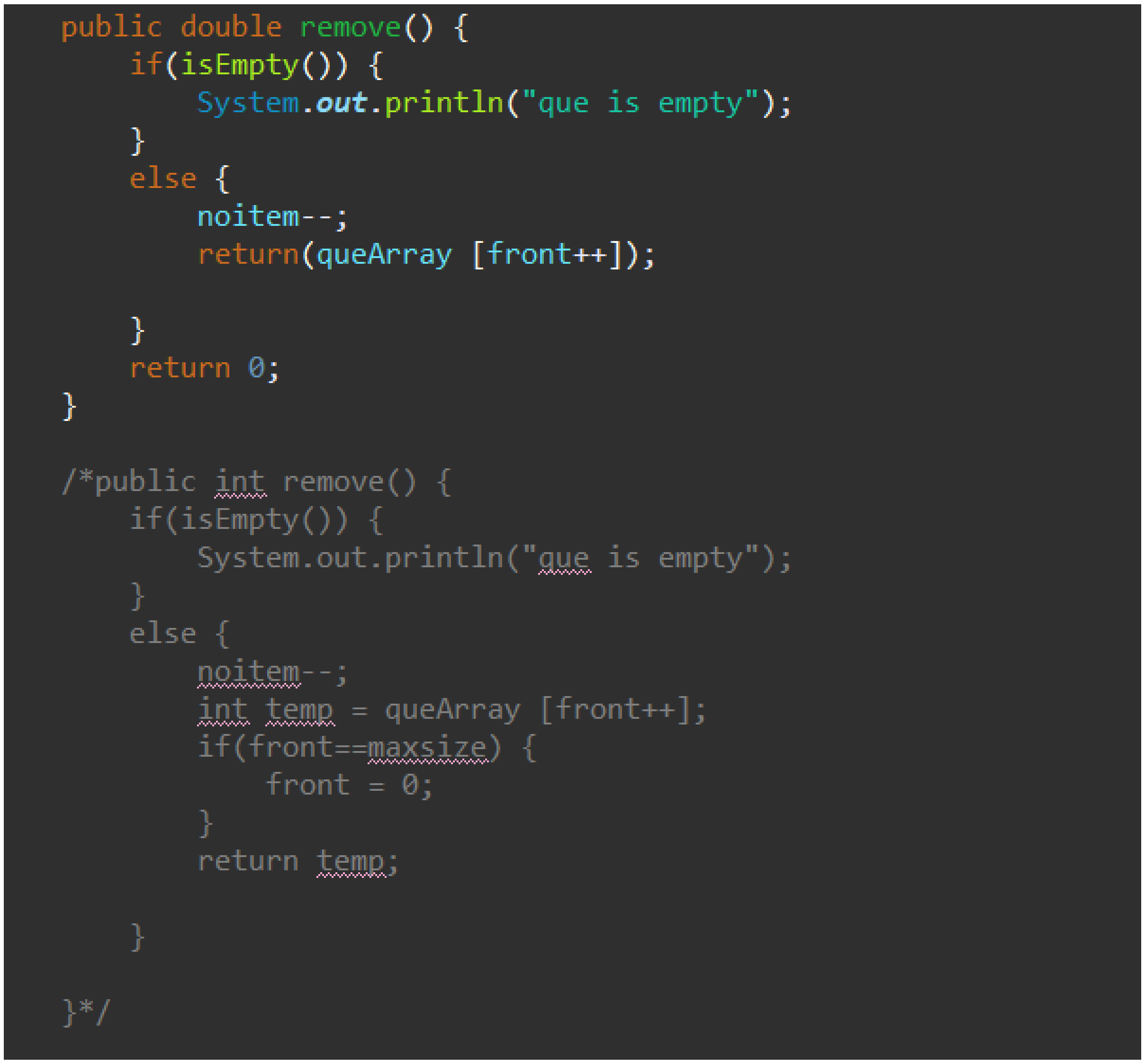
```
    for(int i = 0;i<=4;i++) {
        q1.insert(q2.remove());

    }

    for(int i =0;i<=4;i++) {
        q1.insert(s1.pop());
    }

    System.out.println("final queue");

    for(int i = 0;i<=9;i++) {
        System.out.println("ITem: "+ q1.remove());


    }


}
```

5) Consider a linear queue to store double values. Implement a method to calculate the mean value of the currently available values in the queue and insert it to the same queue. Note that the queue class is already implemented and assume the queue object is "q". (4 marks)

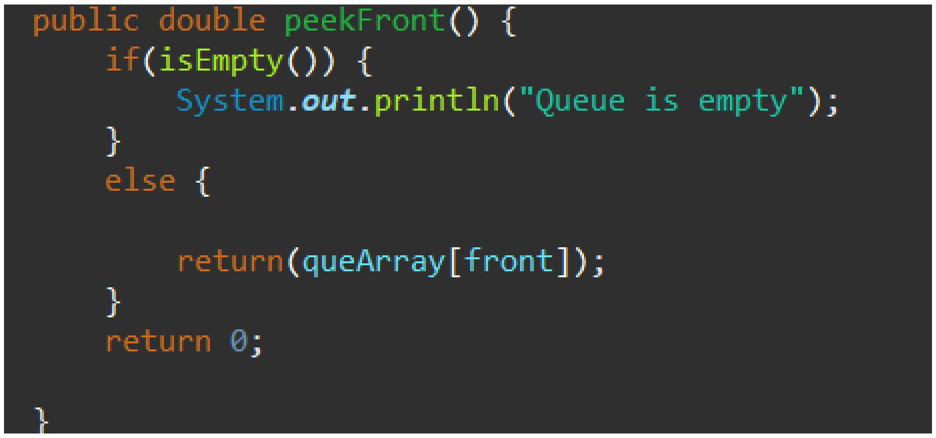


```java
public class Queue {
    private int maxsize;
    private int noitem;
    private int front;
    private int rear;
    private double[] queArray;


    public Queue(int size) {
        this.maxsize = size;
        this.queArray = new double[maxsize];
        front = 0;
        rear= -1;
        noitem=0;

    }

    public boolean isFull() {
        if(rear ==maxsize- 1) {
            return true;
        }
        return false;
    }

    public boolean isEmpty() {
        if(noitem == 0) {
            return true;
        }
        return false;
    }
```

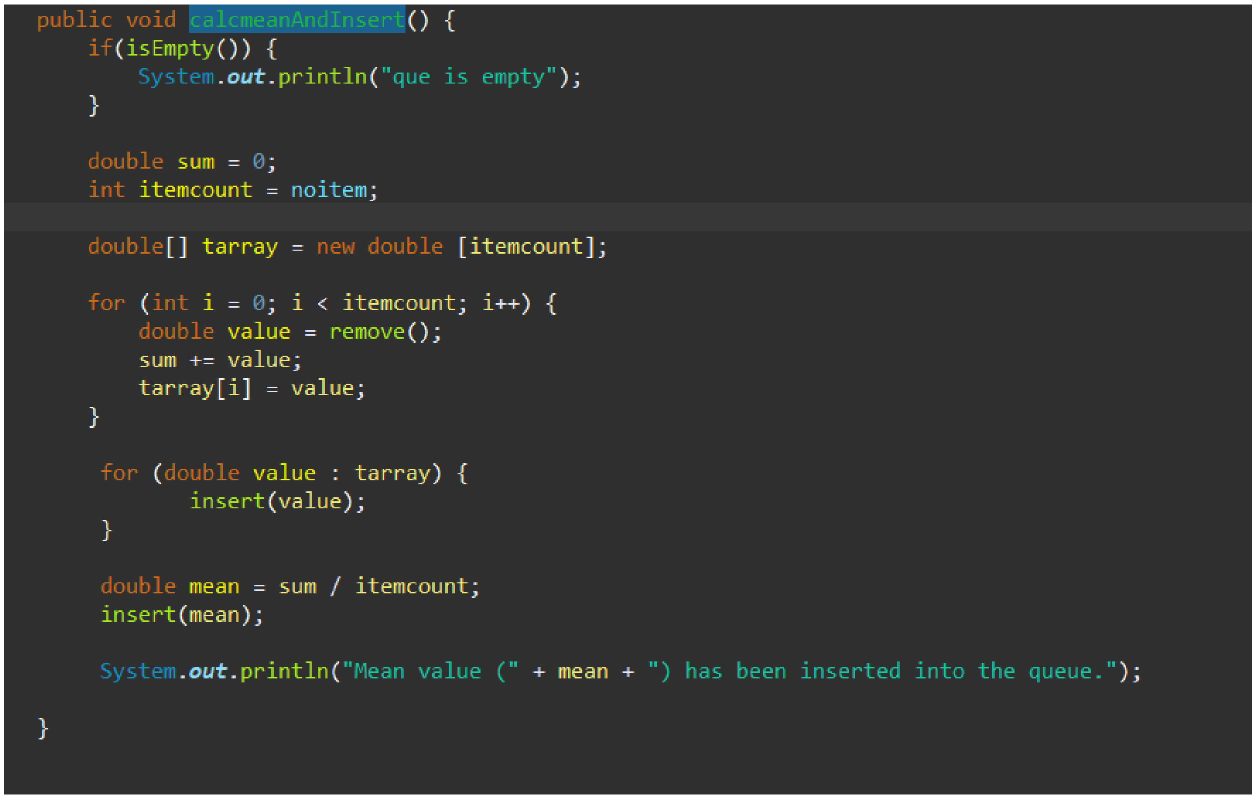

```
/* if circuler

 public boolean isFull() {
    if(noitem ==maxsize) {
        return true;
    }
}*/

public void insert(double value) {
    if(isFull()) {
        System.out.println("Queue is full");
    }
    else {
        queArray[++rear]=value;
        noitem++;
    }
}

/* circuler que

  public void insert(double j) {

    if(isFull()) {
        System.out.println("Queue is full");
    }
    else if(rear  == maxsize-1) {
        rear=-1;
    }
    queArray[++rear]=j;
    noitem++;
} */
```



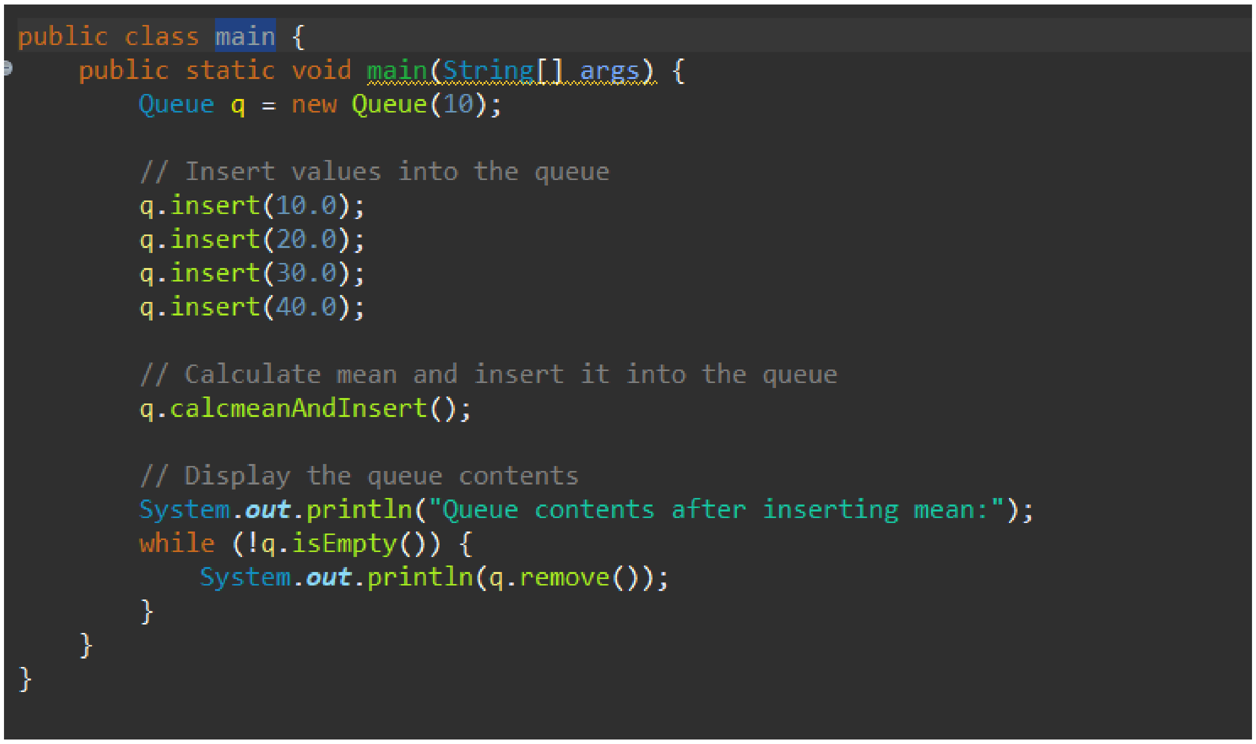
```
  public double remove() {
      if(isEmpty()) {
          System.out.println("que is empty");
      }
      else {
          noitem--;
          return(queArray [front++]);

      }
      return 0;
  }

  /*public int remove() {
      if(isEmpty()) {
          System.out.println("que is empty");
      }
      else {
          noitem--;
          int temp = queArray [front++];
          if(front==maxsize) {
              front = 0;
          }
          return temp;

      }

  }*/
```



```
public double peekFront() {
    if(isEmpty()) {
        System.out.println("Queue is empty");
    }
    else {

        return(queArray[front]);
    }
    return 0;

}
```



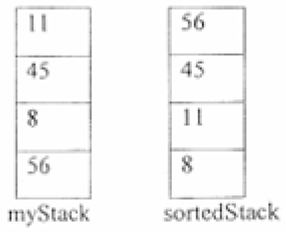
```
public void calcmeanAndInsert() {
    if(isEmpty()) {
        System.out.println("que is empty");
    }

    double sum = 0;
    int itemcount = noitem;

    double[] tarray = new double [itemcount];

    for (int i = 0; i < itemcount; i++) {
        double value = remove();
        sum += value;
        tarray[i] = value;
    }

     for (double value : tarray) {
            insert(value);
     }

     double mean = sum / itemcount;
     insert(mean);

     System.out.println("Mean value (" + mean + ") has been inserted into the queue.");

}
```



```
public class main {
    public static void main(String[] args) {
        Queue q = new Queue(10);

        // Insert values into the queue
        q.insert(10.0);
        q.insert(20.0);
        q.insert(30.0);
        q.insert(40.0);

        // Calculate mean and insert it into the queue
        q.calcmeanAndInsert();

        // Display the queue contents
        System.out.println("Queue contents after inserting mean:");
        while (!q.isEmpty()) {
            System.out.println(q.remove());
        }
    }
}
```

f) Write a code segment in your main program using java to sort the values in a stack in ascending order. Assume the stack object **myStack** is available with values and another empty stack object called **sortedStack** is already created. The sorted values should be stored in sortedStack.
(7 marks)

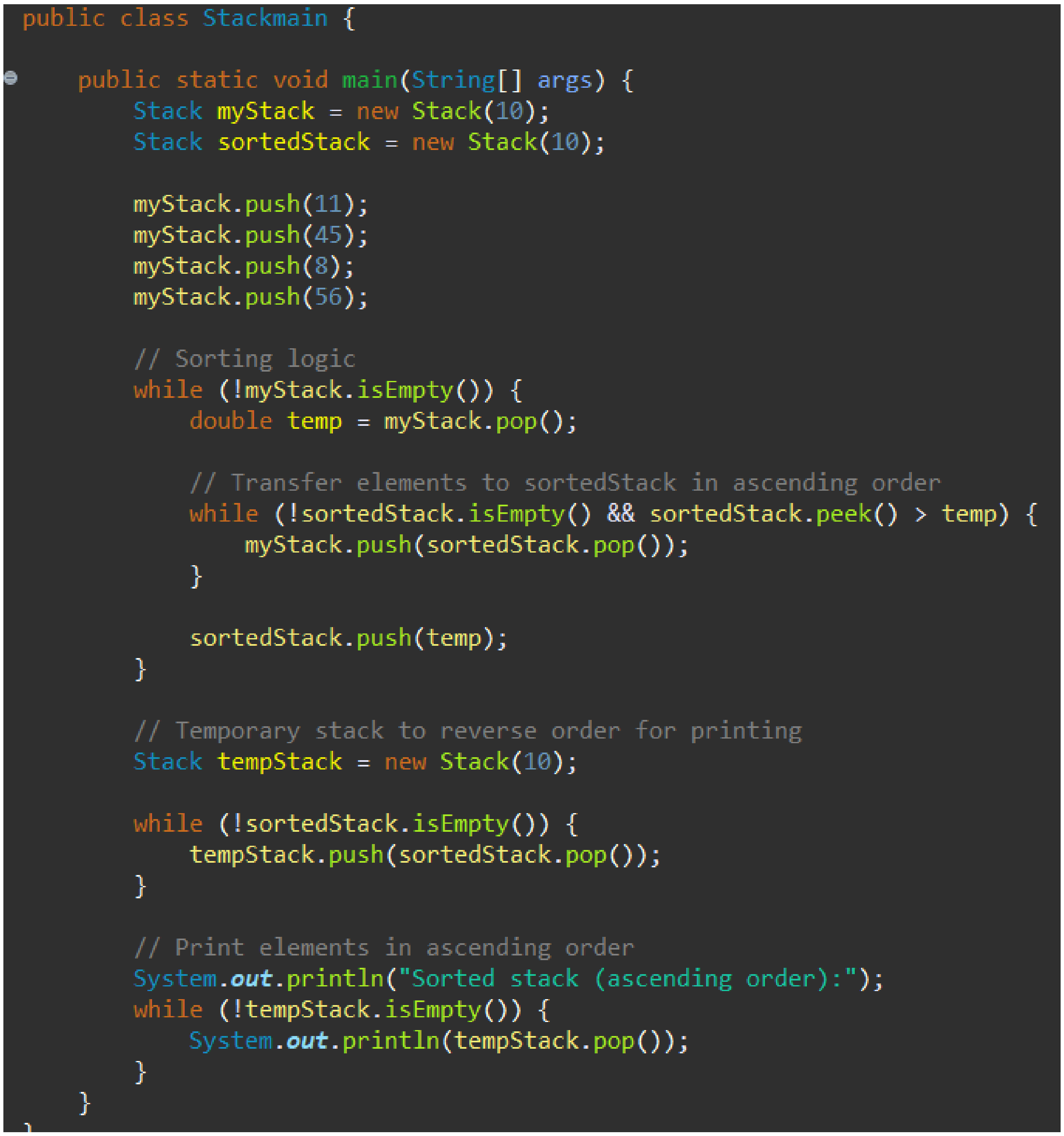
Ex:



00

Consider the StackX class is given below.

StackX
- int[] stackArray - int maxSize - int top
+void push(double j) +double pop( ) +double peek( ) +boolean isEmpty( ) +boolean isFull( )



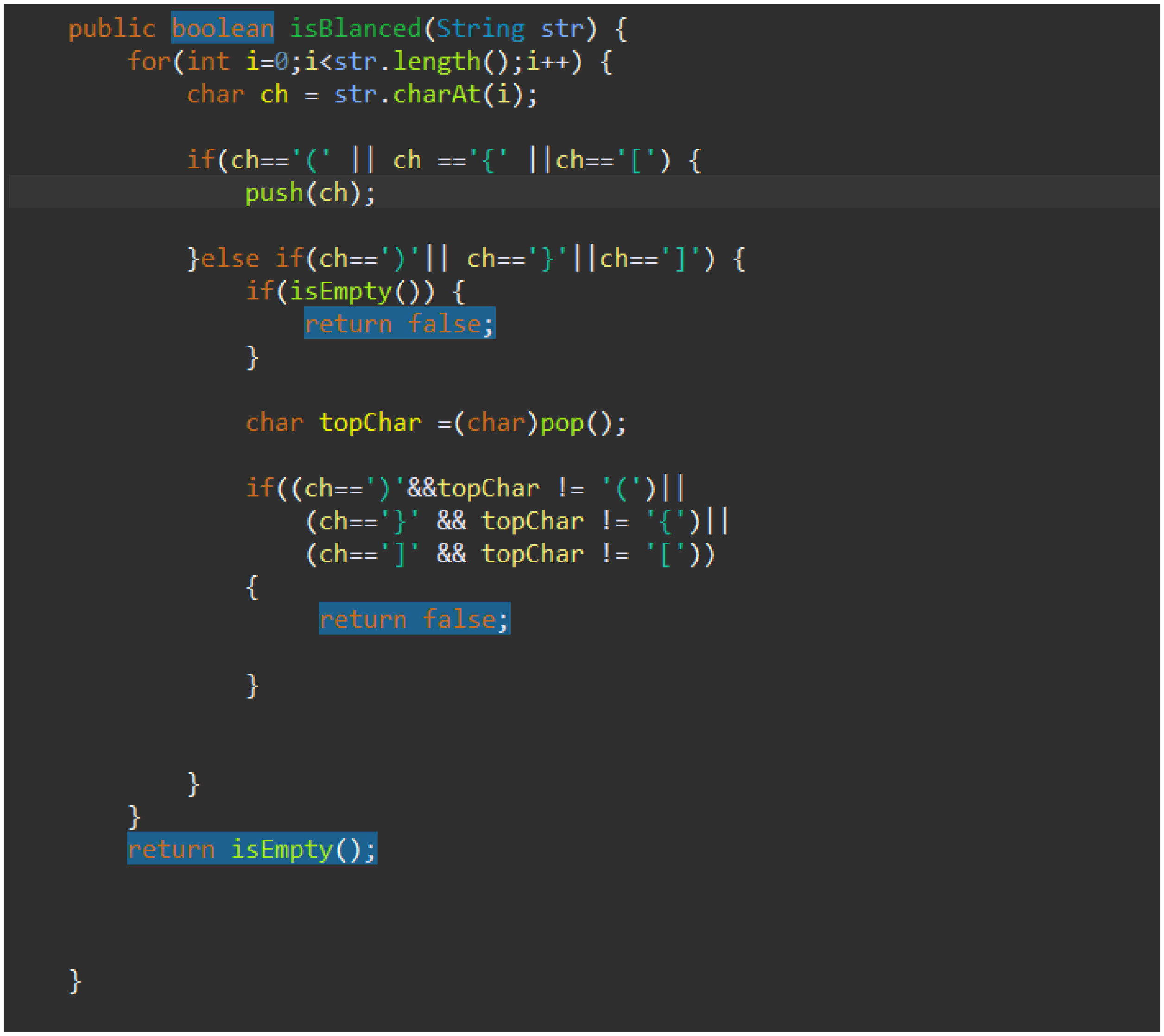
```java
public class Stackmain {

    public static void main(String[] args) {
        Stack myStack = new Stack(10);
        Stack sortedStack = new Stack(10);

        myStack.push(11);
        myStack.push(45);
        myStack.push(8);
        myStack.push(56);

        // Sorting logic
        while (!myStack.isEmpty()) {
            double temp = myStack.pop();

            // Transfer elements to sortedStack in ascending order
            while (!sortedStack.isEmpty() && sortedStack.peek() > temp) {
                myStack.push(sortedStack.pop());
            }

            sortedStack.push(temp);
        }

        // Temporary stack to reverse order for printing
        Stack tempStack = new Stack(10);

        while (!sortedStack.isEmpty()) {
            tempStack.push(sortedStack.pop());
        }

        // Print elements in ascending order
        System.out.println("Sorted stack (ascending order):");
        while (!tempStack.isEmpty()) {
            System.out.println(tempStack.pop());
        }
    }
```

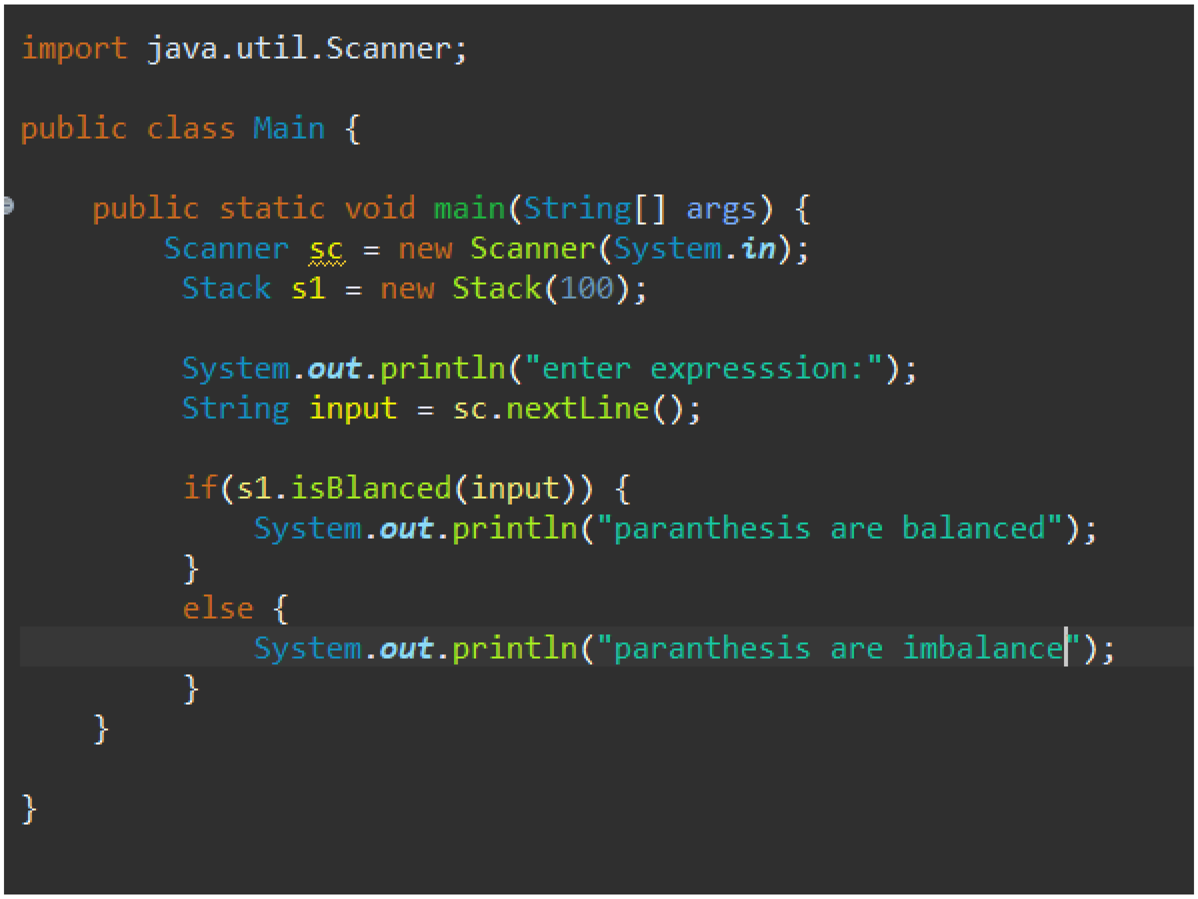
e) Assume the following StackX class is implemented. You need to introduce a new method to StackX class to check whether parentheses are balanced. (8 Marks)

Write Java code segment for ***boolean isBalanced(String str)*** to take a string input from the user. Using the stack data structure your program should return true for balanced parentheses i.e. {[()]}, and false for imbalanced parentheses i.e. {{(([]})].

StackX
- int[] stackArray - int maxSize - int top
+void push(char c) +double pop( ) +double peek( ) +boolean isEmpty( ) +boolean isFull( )

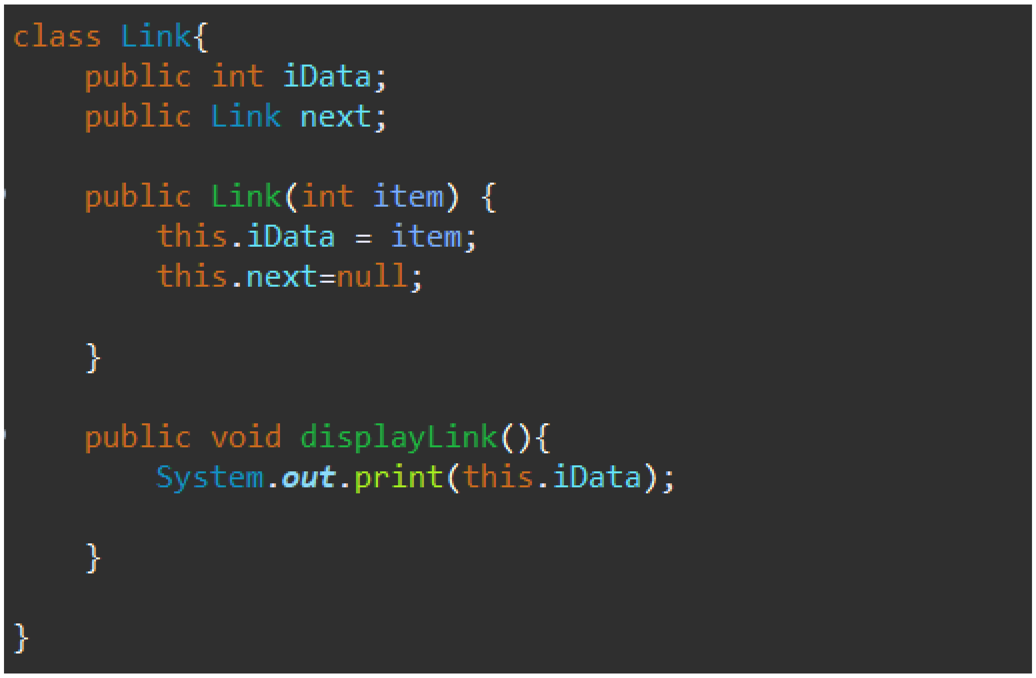
f) Write the main program to call isBalanced(String str) and display "Parentheses are balanced" for true instances and "Parentheses are imbalanced" for false instances. (3 Marks)


```java
    public boolean isBlanced(String str) {
        for(int i=0;i<str.length();i++) {
            char ch = str.charAt(i);

            if(ch=='(' || ch =='{' ||ch=='[') {
                push(ch);

            }else if(ch==')'|| ch=='}'||ch==']') {
                if(isEmpty()) {
                    return false;
                }

                char topChar =(char)pop();

                if((ch==')'&&topChar != '(')||
                    (ch=='}' && topChar != '{')||
                    (ch==']' && topChar != '['))
                {
                     return false;

                }


            }
        }
        return isEmpty();



    }
```

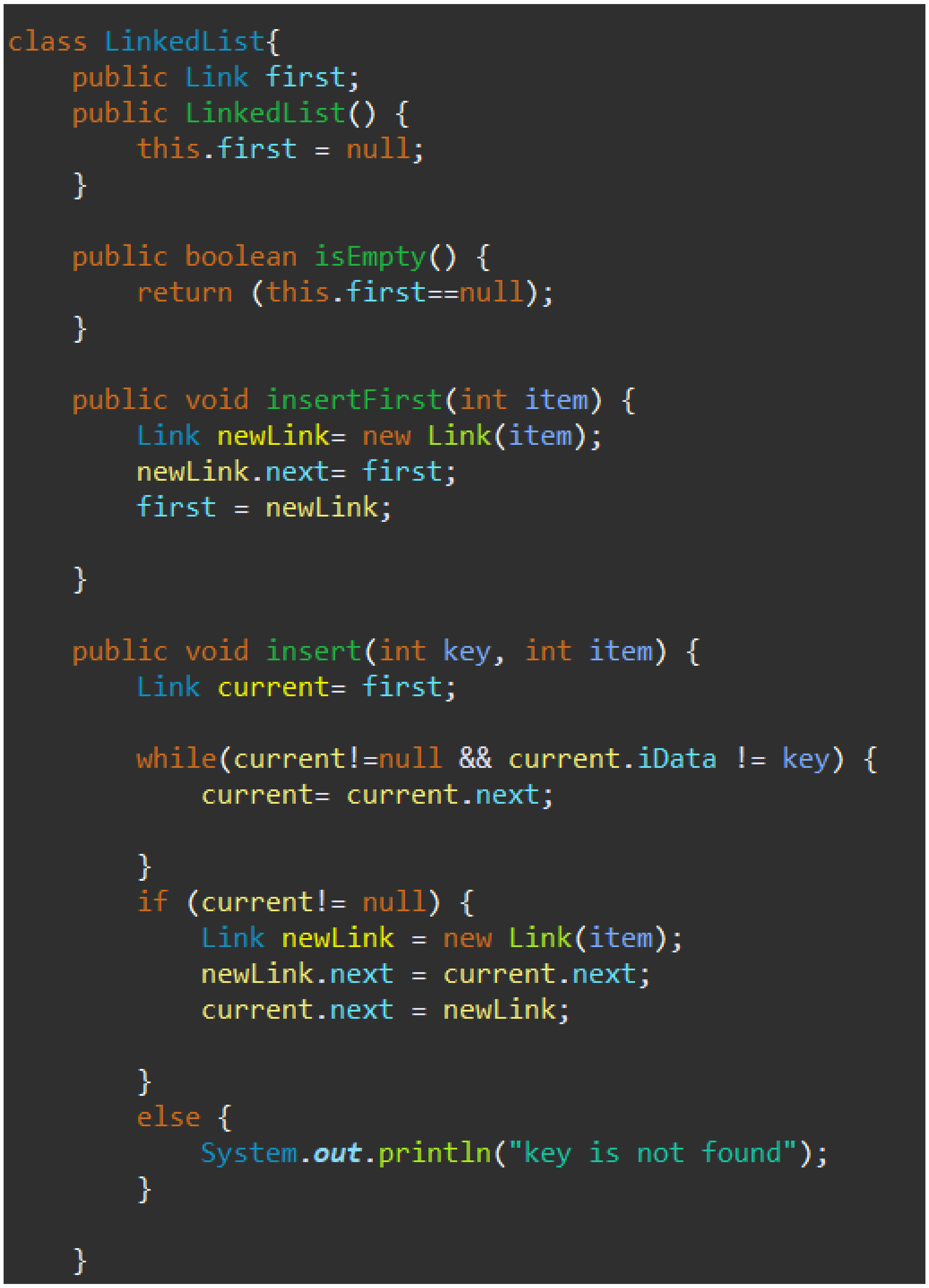


```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
         Stack s1 = new Stack(100);

         System.out.println("enter expresssion:");
         String input = sc.nextLine();

         if(s1.isBlanced(input)) {
             System.out.println("paranthesis are balanced");
         }
         else {
             System.out.println("paranthesis are imbalance");
         }
    }

}
```

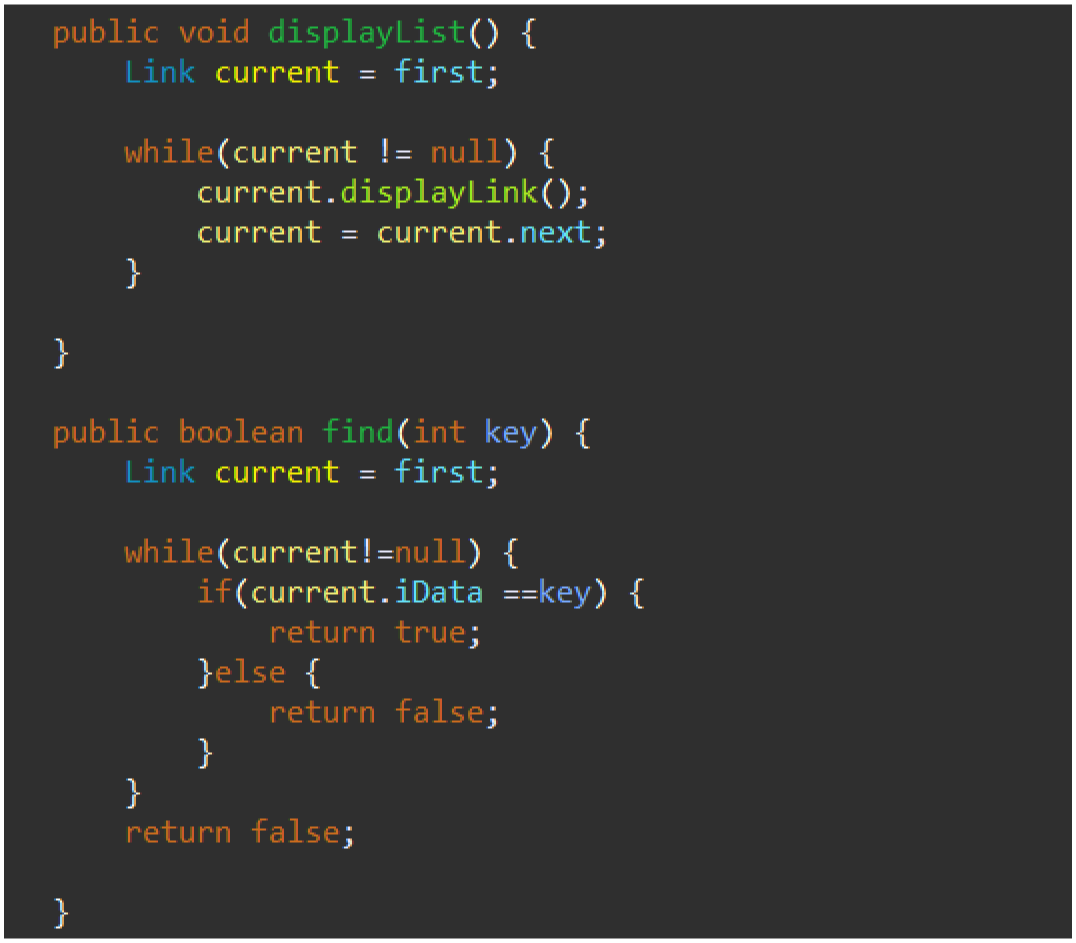
Linked list coding



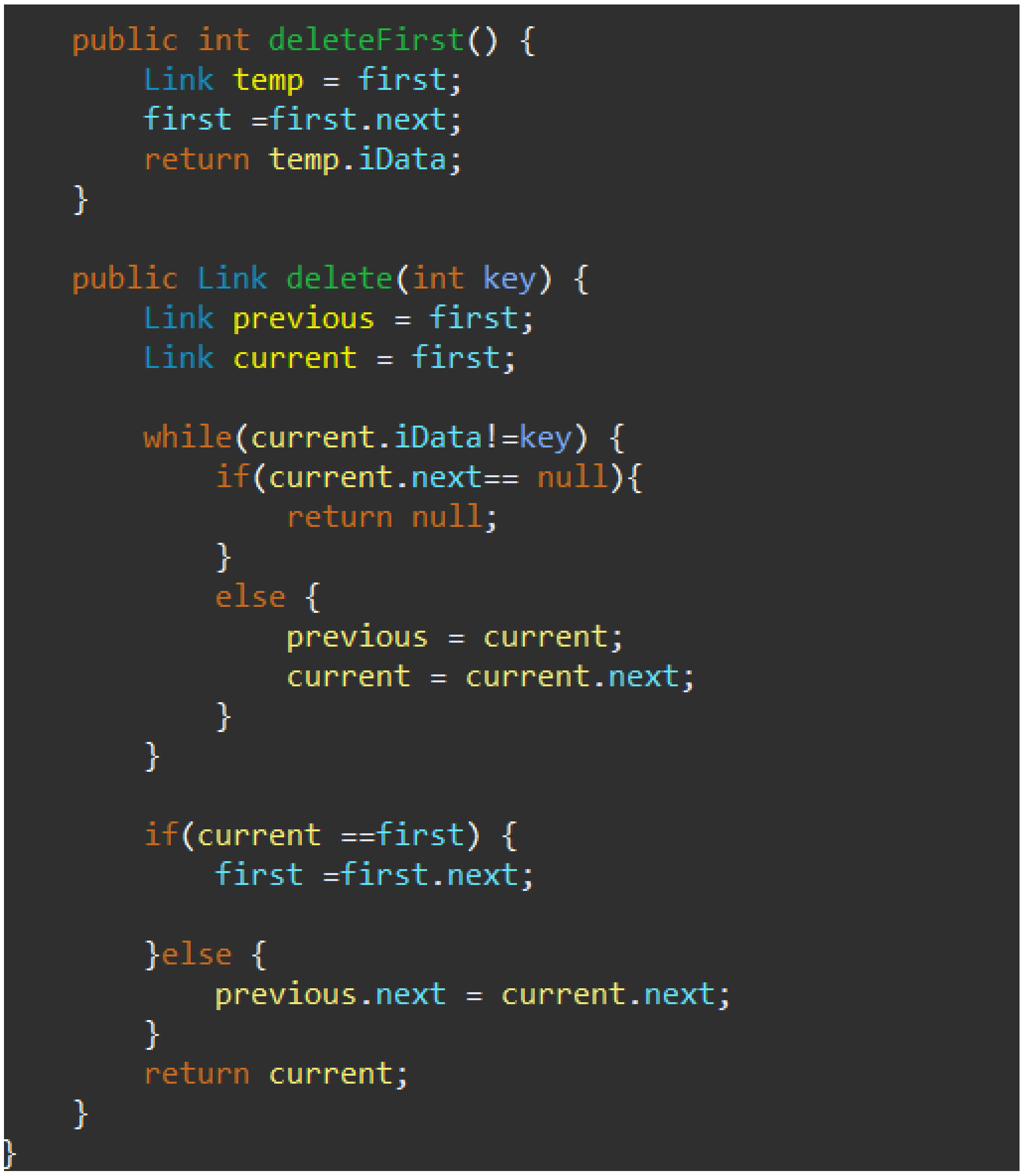
```
class Link{
    public int iData;
    public Link next;

    public Link(int item) {
        this.iData = item;
        this.next=null;

    }

    public void displayLink(){
        System.out.print(this.iData);

    }

}
```



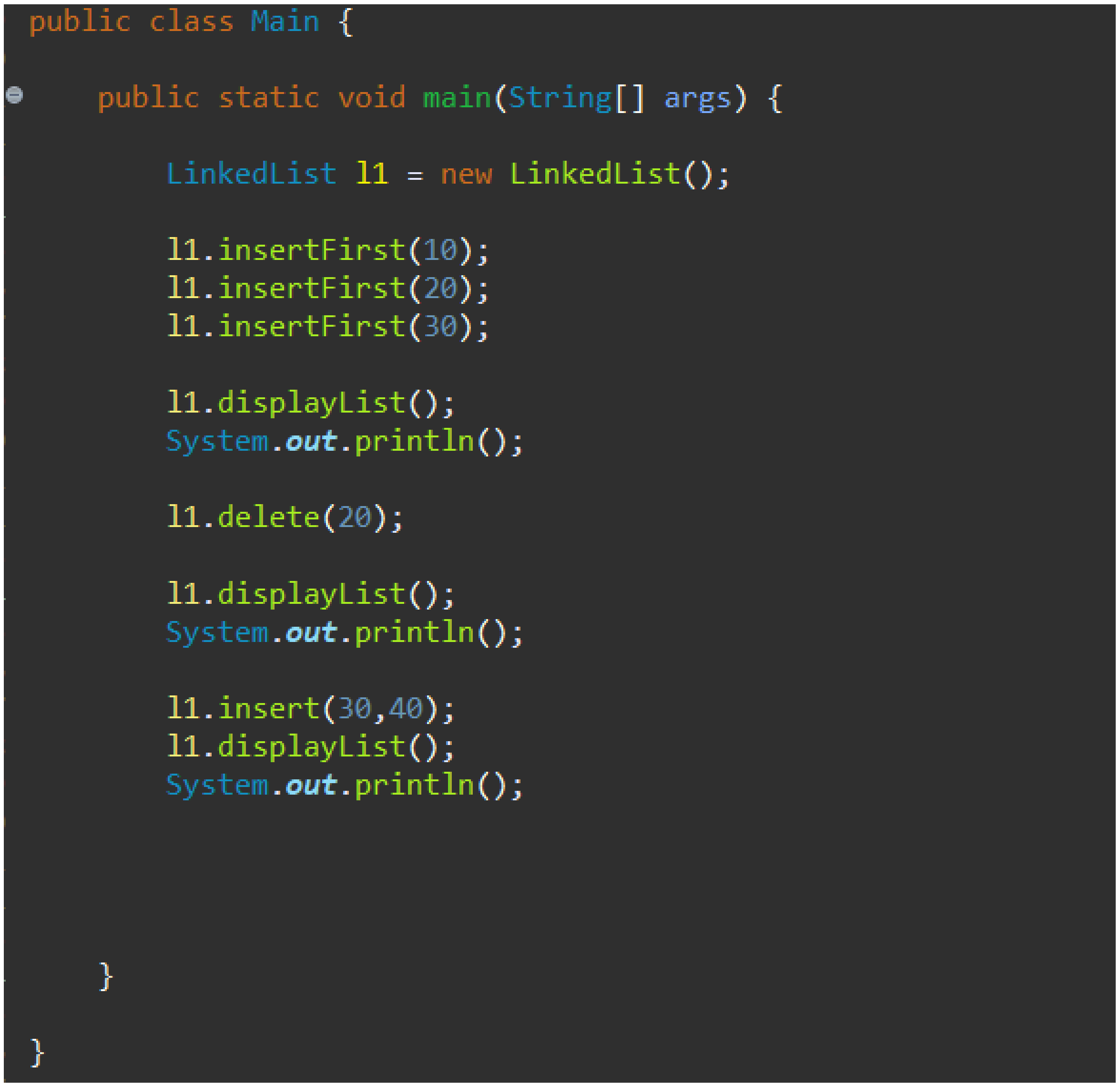
```
class LinkedList{
    public Link first;
    public LinkedList() {
        this.first = null;
    }

    public boolean isEmpty() {
        return (this.first==null);
    }

    public void insertFirst(int item) {
        Link newLink= new Link(item);
        newLink.next= first;
        first = newLink;

    }

    public void insert(int key, int item) {
        Link current= first;

        while(current!=null && current.iData != key) {
            current= current.next;

        }
        if (current!= null) {
            Link newLink = new Link(item);
            newLink.next = current.next;
            current.next = newLink;

        }
        else {
            System.out.println("key is not found");
        }

    }
```



```
public void displayList() {
    Link current = first;

    while(current != null) {
        current.displayLink();
        current = current.next;
    }

}

public boolean find(int key) {
    Link current = first;

    while(current!=null) {
        if(current.iData ==key) {
            return true;
        }else {
            return false;
        }
    }
    return false;

}
```



```
    public int deleteFirst() {
        Link temp = first;
        first =first.next;
        return temp.iData;
    }

    public Link delete(int key) {
        Link previous = first;
        Link current = first;

        while(current.iData!=key) {
            if(current.next== null){
                return null;
            }
            else {
                previous = current;
                current = current.next;
            }
        }

        if(current ==first) {
            first =first.next;

        }else {
            previous.next = current.next;
        }
        return current;
    }
}
```



```java
public class Main {

    public static void main(String[] args) {

        LinkedList l1 = new LinkedList();

        l1.insertFirst(10);
        l1.insertFirst(20);
        l1.insertFirst(30);

        l1.displayList();
        System.out.println();

        l1.delete(20);

        l1.displayList();
        System.out.println();

        l1.insert(30,40);
        l1.displayList();
        System.out.println();




    }

}
```

b) In a hospital management system, patient beds are stored in a link list. patientBed class and linkList classes are given below.
'Vacant' attribute of patientBed class will be 0 when the patient is admitted to the hospital and assigned the bed. It will be 1 when the patient is discharged.

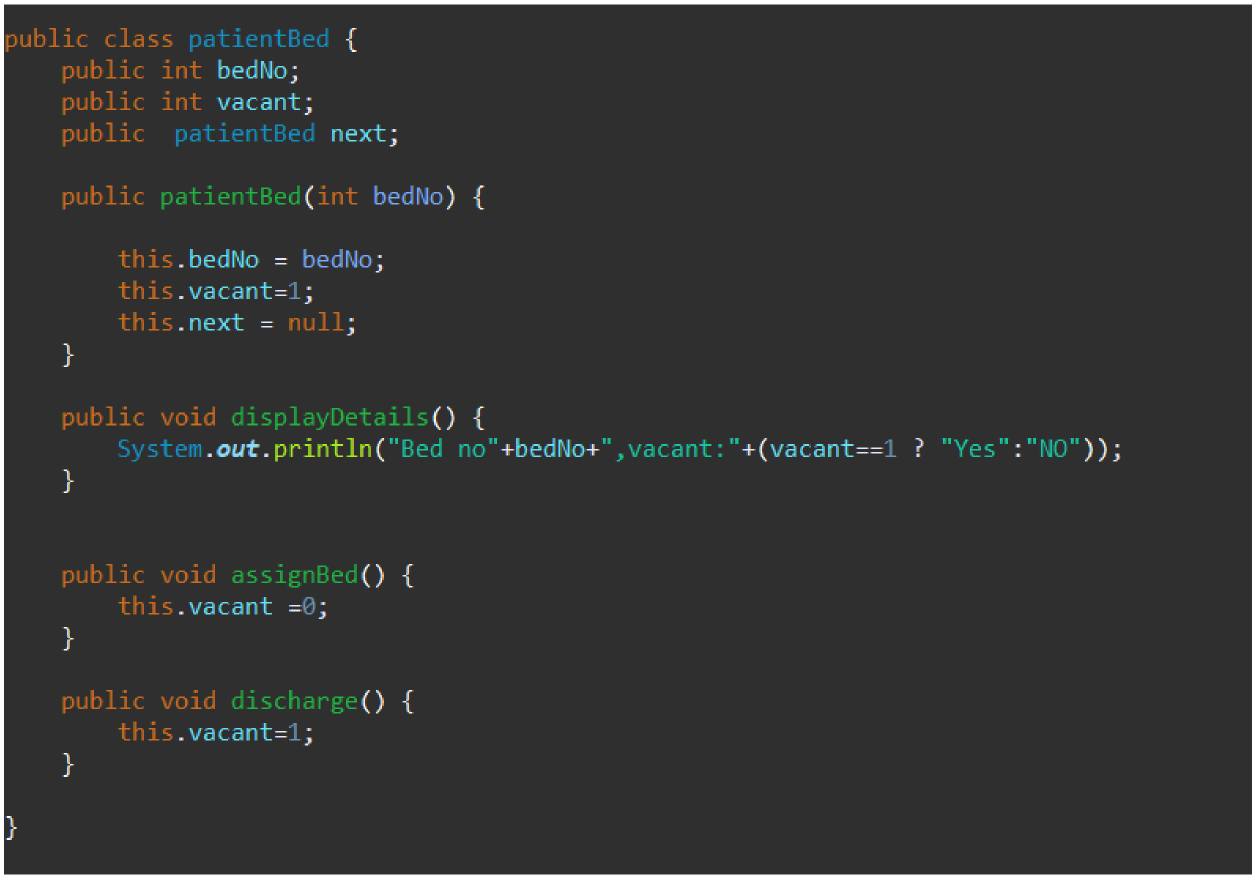
Assume the two classes have already been implemented.

patientBed
int bedNo int vacant patientBed next
patientBed (int bedNum) void displayDetails() void assignBed() void discharge()

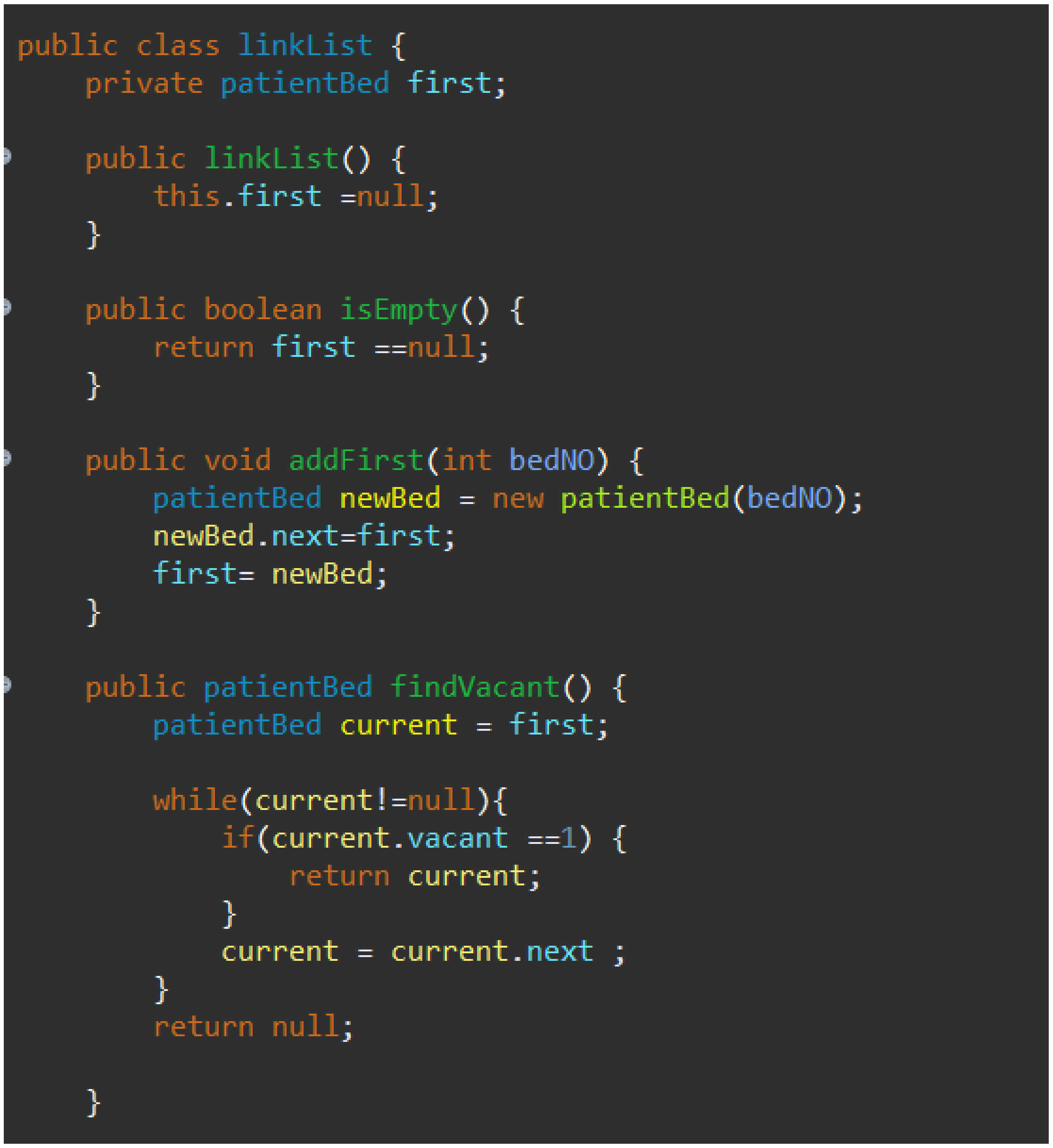
linkList
patientBed first;
void linkList() boolean isEmpty() void addFirst(int bedNo) patientBed findVacant()

i) You need to modify the linkList class by adding another method called countVacantBeds() in the linkList class. This method returns the number of vacant patient beds. Implement the method. (5 marks)

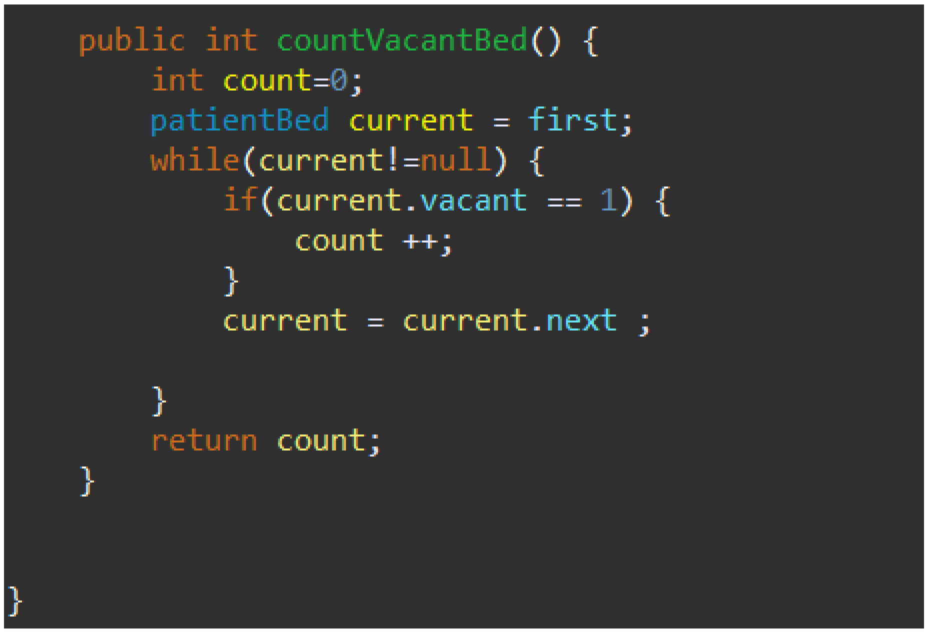
ii) findVacant() method in the linkList class returns the first vacant patientBed. If there are no vacant beds, it will return NULL. Write code segment in your main application to find a vacant bed and assign the bed. Display the assigned bedNo. If there are no vacant beds in the hospital display the message "Patient beds are not available". (5 marks)



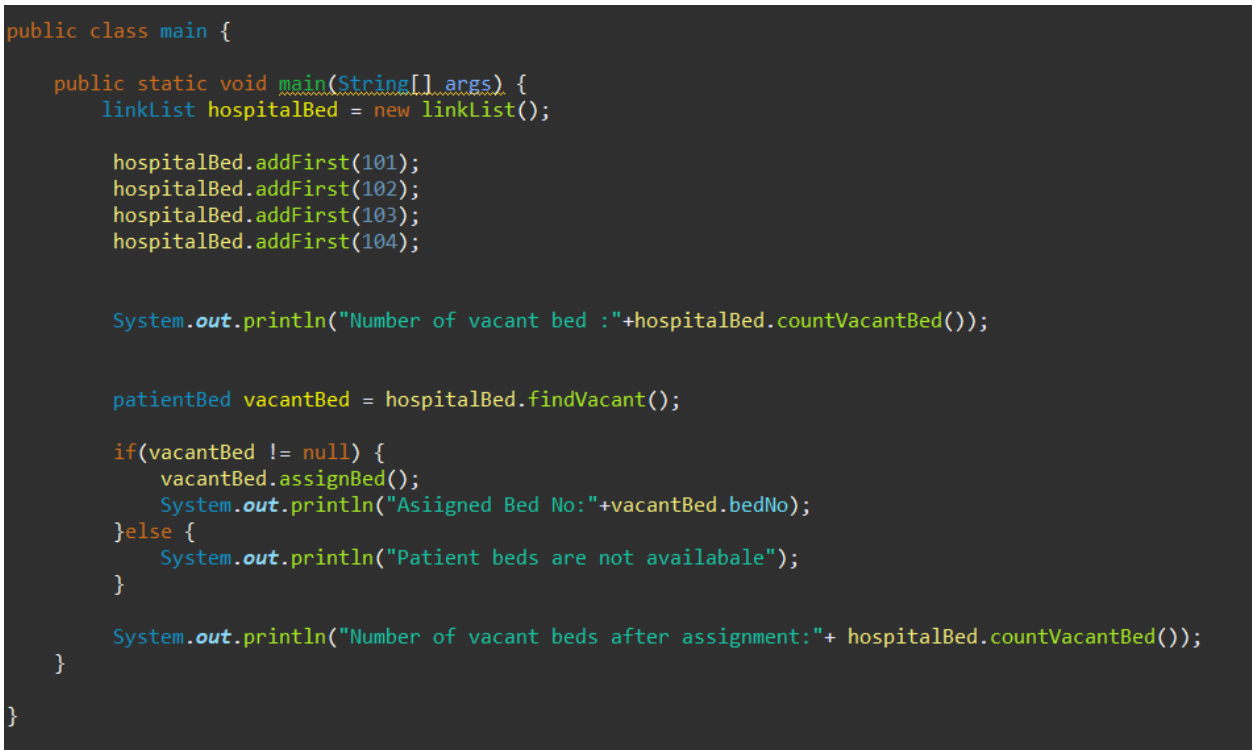
```java
public class patientBed {
    public int bedNo;
    public int vacant;
    public  patientBed next;

    public patientBed(int bedNo) {

        this.bedNo = bedNo;
        this.vacant=1;
        this.next = null;
    }

    public void displayDetails() {
        System.out.println("Bed no"+bedNo+",vacant:"+(vacant==1 ? "Yes":"NO"));
    }


    public void assignBed() {
        this.vacant =0;
    }

    public void discharge() {
        this.vacant=1;
    }

}
```



```java
public class linkList {
    private patientBed first;

    public linkList() {
        this.first =null;
    }

    public boolean isEmpty() {
        return first ==null;
    }

    public void addFirst(int bedNO) {
        patientBed newBed = new patientBed(bedNO);
        newBed.next=first;
        first= newBed;
    }

    public patientBed findVacant() {
        patientBed current = first;

        while(current!=null){
            if(current.vacant ==1) {
                return current;
            }
            current = current.next ;
        }
        return null;

    }
```



```
    public int countVacantBed() {
        int count=0;
        patientBed current = first;
        while(current!=null) {
            if(current.vacant == 1) {
                count ++;
            }
            current = current.next ;

        }
        return count;
    }


}
```



```
public class main {

    public static void main(String[] args) {
        linkList hospitalBed = new linkList();

         hospitalBed.addFirst(101);
         hospitalBed.addFirst(102);
         hospitalBed.addFirst(103);
         hospitalBed.addFirst(104);


         System.out.println("Number of vacant bed :"+hospitalBed.countVacantBed());


         patientBed vacantBed = hospitalBed.findVacant();

         if(vacantBed != null) {
             vacantBed.assignBed();
             System.out.println("Asiigned Bed No:"+vacantBed.bedNo);
         }else {
             System.out.println("Patient beds are not availabale");
         }

         System.out.println("Number of vacant beds after assignment:"+ hospitalBed.countVacantBed());
    }

}
```