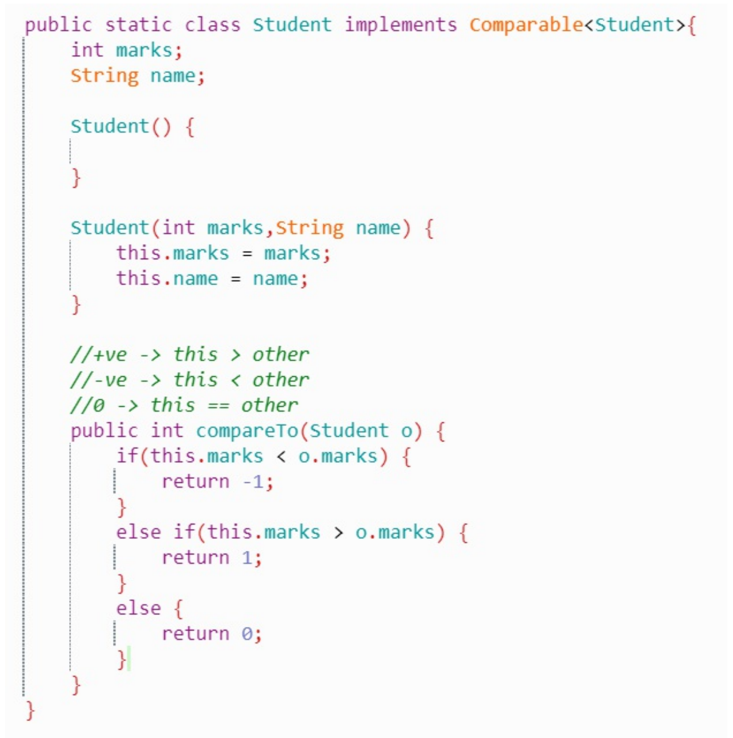
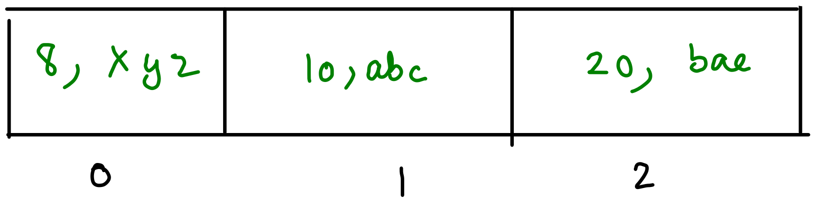


```java
public static class Student implements Comparable<Student>{
    int marks;
    String name;

    Student() {

    }

    Student(int marks,String name) {
        this.marks = marks;
        this.name = name;
    }

    //+ve -> this > other
    //-ve -> this < other
    //0 -> this == other
    public int compareTo(Student o) {
        if(this.marks < o.marks) {
            return -1;
        }
        else if(this.marks > o.marks) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```

Arrays.sort(arr);



arr[j].compareTo(arr[j+1]);

Merge Overlapping Interval

T : nlogn

27	30
22	28
5	12
1	8
14	19
25	27

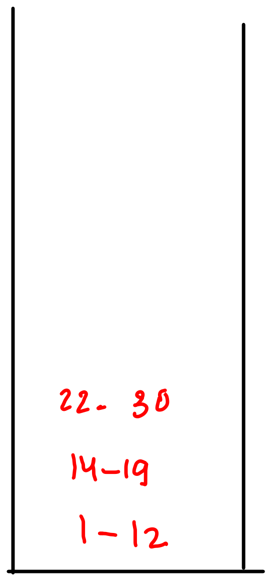
1	12
14	19
22	30

25	27
5	12
26	30
22	28
1	8
14	19

$\xrightarrow{\text{sort}}$

1	8
5	12
14	19
22	28
25	27
26	30

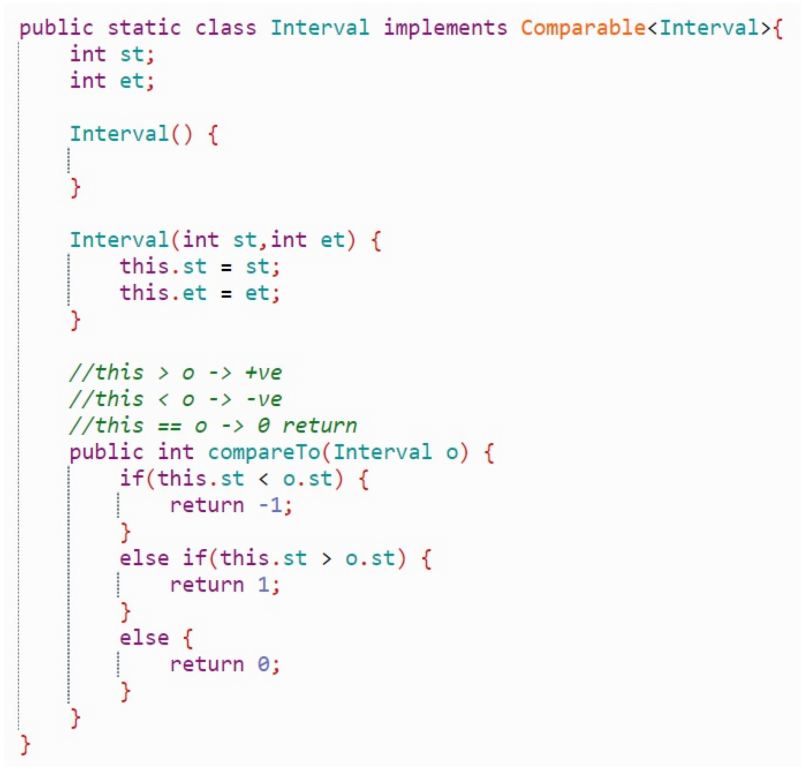
✓



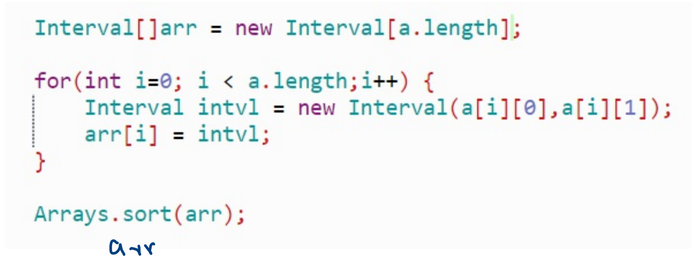
1-12

14-19

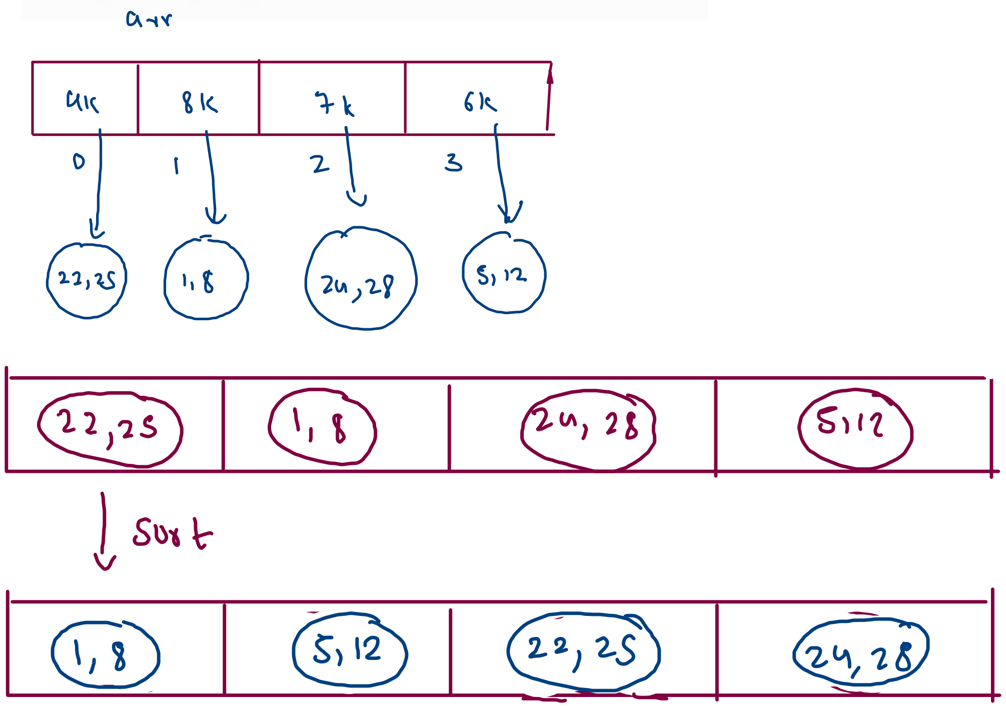
22-30



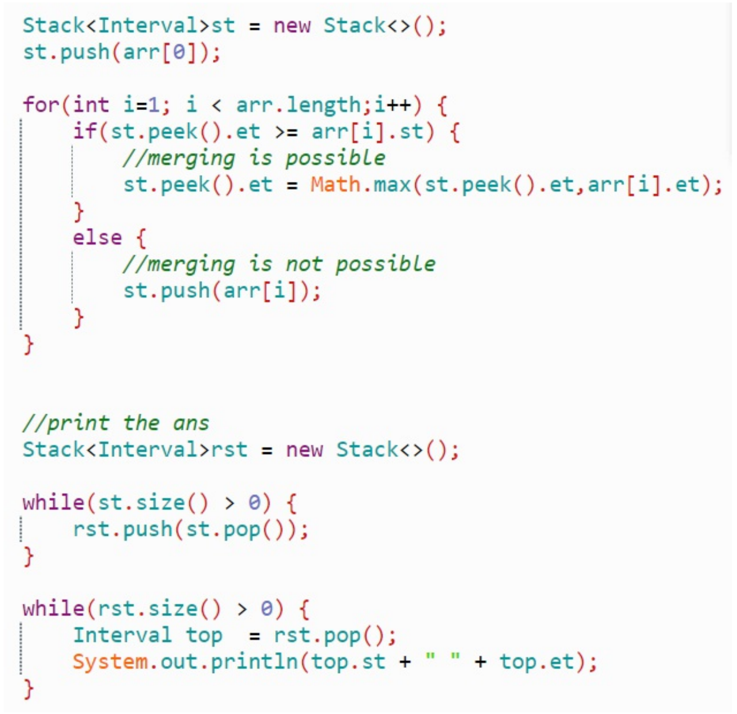
```java
public static class Interval implements Comparable<Interval>{
    int st;
    int et;

    Interval() {

    }

    Interval(int st,int et) {
        this.st = st;
        this.et = et;
    }

    //this > o -> +ve
    //this < o -> -ve
    //this == o -> 0 return
    public int compareTo(Interval o) {
        if(this.st < o.st) {
            return -1;
        }
        else if(this.st > o.st) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```



```java
Interval[]arr = new Interval[a.length];

for(int i=0; i < a.length;i++) {
    Interval intvl = new Interval(a[i][0],a[i][1]);
    arr[i] = intvl;
}

Arrays.sort(arr);
```



n = 4



```java
Stack<Interval>st = new Stack<>();
st.push(arr[0]);

for(int i=1; i < arr.length;i++) {
    if(st.peek().et >= arr[i].st) {
        //merging is possible
        st.peek().et = Math.max(st.peek().et,arr[i].et);
    }
    else {
        //merging is not possible
        st.push(arr[i]);
    }
}


//print the ans
Stack<Interval>rst = new Stack<>();

while(st.size() > 0) {
    rst.push(st.pop());
}

while(rst.size() > 0) {
    Interval top  = rst.pop();
    System.out.println(top.st + " " + top.et);
}
```

1 - 8

5 - 12

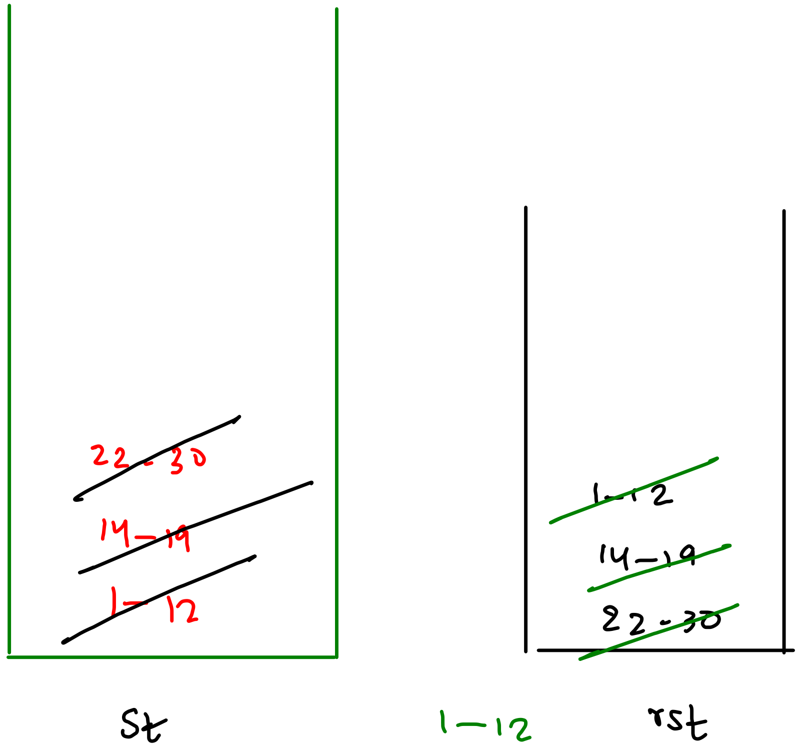
14 - 19

22 - 28

25 - 27

26 - 30

i



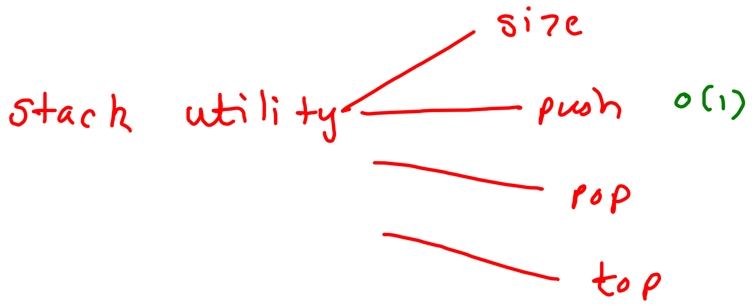
1-12

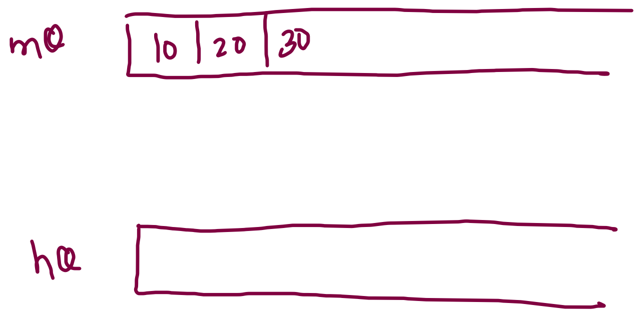
14-19

22-30

Queue To Stack Adapter - Push Efficient

```java
public static class QueueToStackAdapter {
  Queue<Integer> mainQ;
  Queue<Integer> helperQ;

  public QueueToStackAdapter() {
    mainQ = new ArrayDeque<>();
    helperQ = new ArrayDeque<>();
  }

  int size() {
    // write your code here
  }

  void push(int val) {
    // write your code here
  }

  int pop() {
    // write your code here
  }

  int top() {
    // write your code here
  }
}
```





st.push(10);

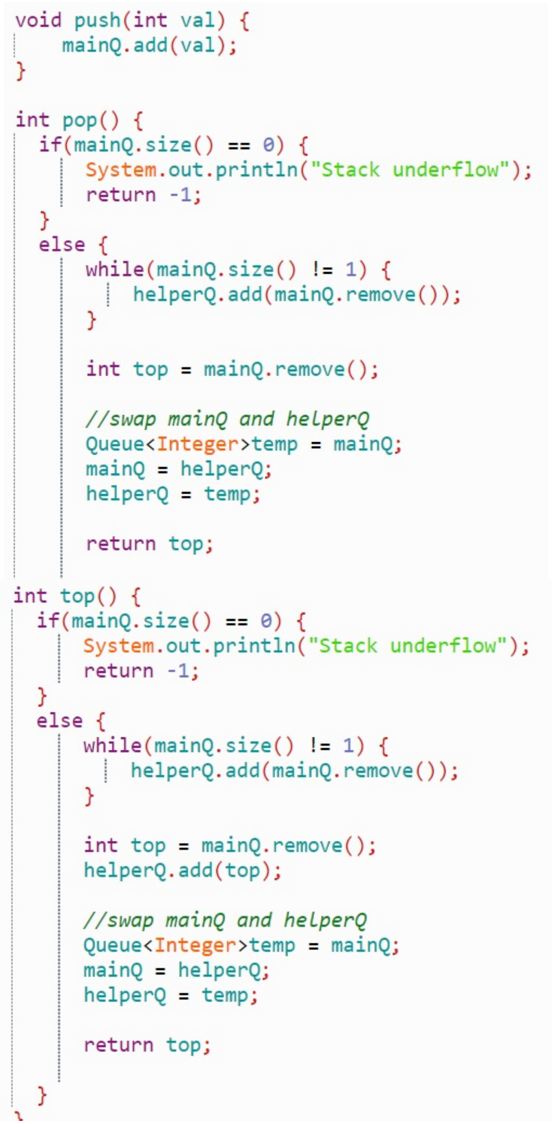
st.push(20);

st.push(30);

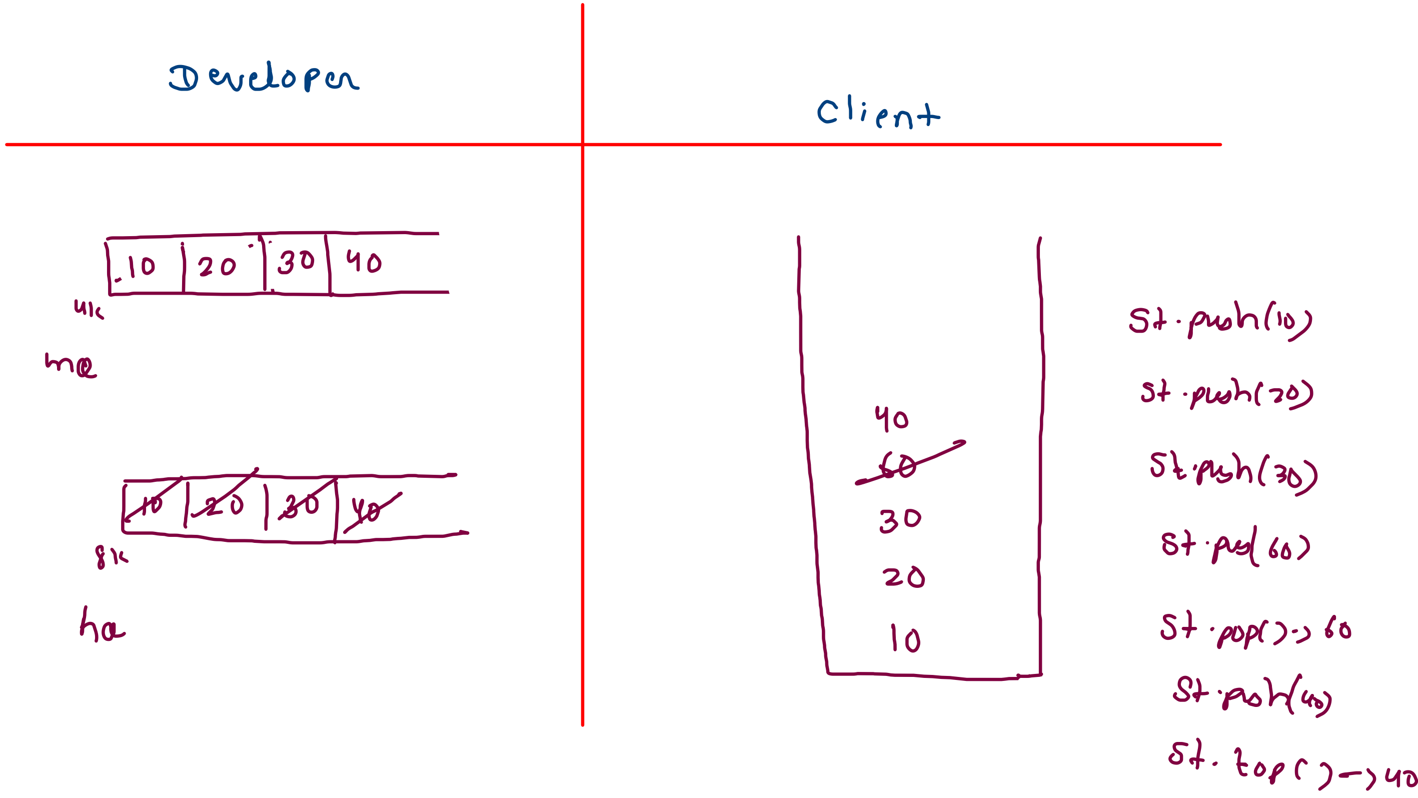
st.top() -> 30

st.push(40);

st.pop() -> 40

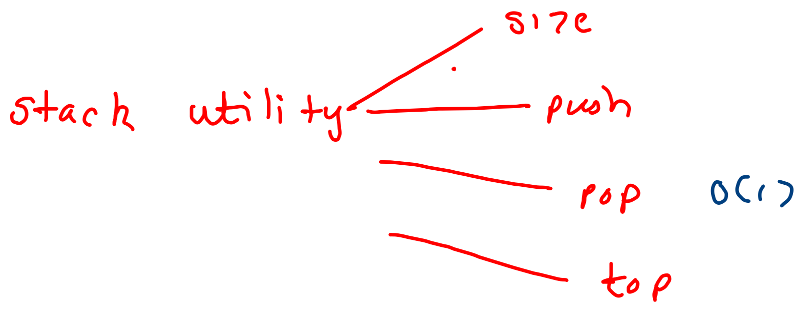


```java
void push(int val) {
    mainQ.add(val);
}

int pop() {
  if(mainQ.size() == 0) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      while(mainQ.size() != 1) {
          helperQ.add(mainQ.remove());
      }

      int top = mainQ.remove();

      //swap mainQ and helperQ
      Queue<Integer>temp = mainQ;
      mainQ = helperQ;
      helperQ = temp;

      return top;
int top() {
  if(mainQ.size() == 0) {
      System.out.println("Stack underflow");
      return -1;
  }
  else {
      while(mainQ.size() != 1) {
          helperQ.add(mainQ.remove());
      }

      int top = mainQ.remove();
      helperQ.add(top);

      //swap mainQ and helperQ
      Queue<Integer>temp = mainQ;
      mainQ = helperQ;
      helperQ = temp;

      return top;

  }
}
```



```
St.push(10)
St.push(20)
St.push(30)
St.push(60)
St.pop()→60
St.push(40)
St.top()→40
```

Queue To Stack Adapter - Pop Efficient



St.push(10)

St.push(20)

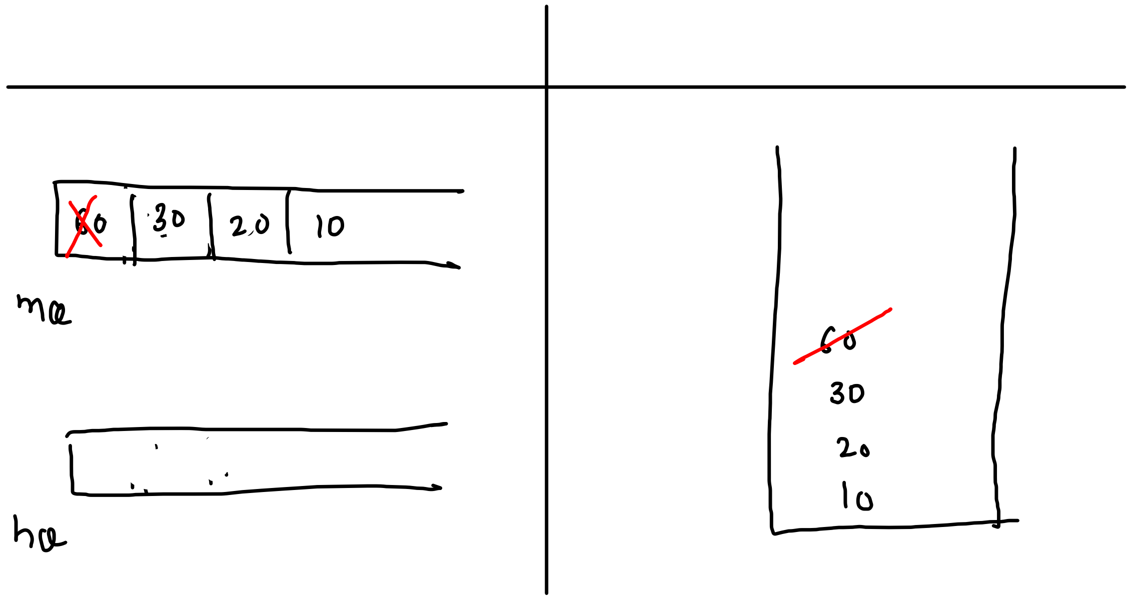
St.push(30)

St.push(60)

St.pop() -> 60

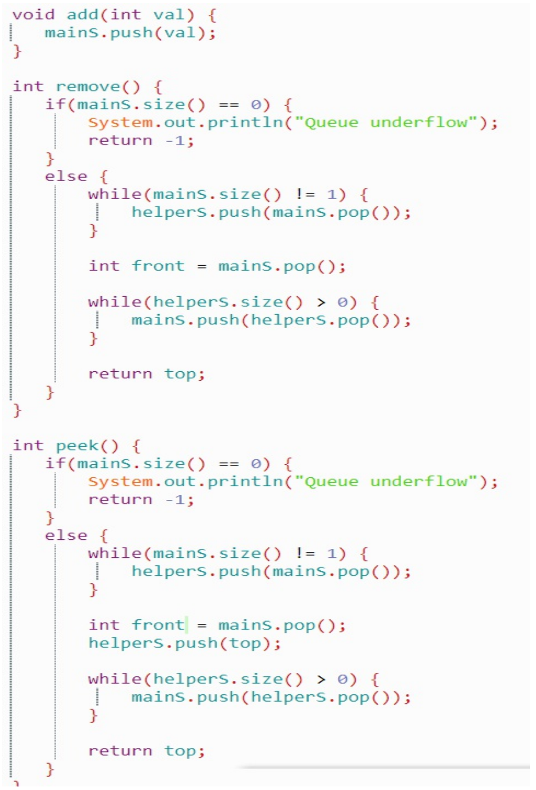
St.push(40)

St.top() -> 40

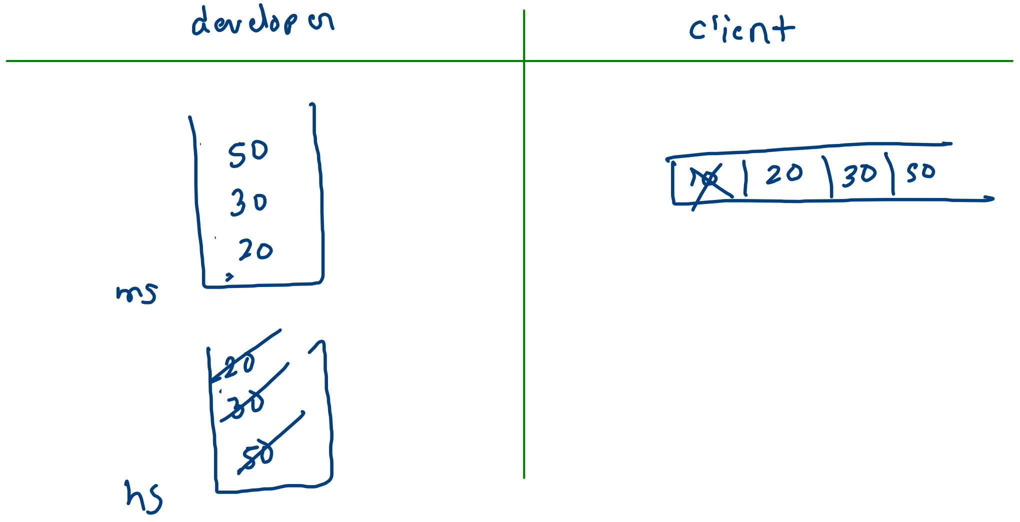


Stack To Queue Adapter - Add Efficient

q
- size
- add O(1)
- remove O(n)
- peek O(n)



```java
void add(int val) {
    mainS.push(val);
}

int remove() {
    if(mainS.size() == 0) {
        System.out.println("Queue underflow");
        return -1;
    }
    else {
        while(mainS.size() != 1) {
            helperS.push(mainS.pop());
        }

        int front = mainS.pop();

        while(helperS.size() > 0) {
            mainS.push(helperS.pop());
        }

        return top;
    }
}

int peek() {
    if(mainS.size() == 0) {
        System.out.println("Queue underflow");
        return -1;
    }
    else {
        while(mainS.size() != 1) {
            helperS.push(mainS.pop());
        }

        int front = mainS.pop();
        helperS.push(top);

        while(helperS.size() > 0) {
            mainS.push(helperS.pop());
        }

        return top;
    }
```



q.add(10)

q.add(20)

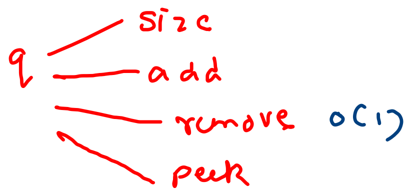
q.add(30)

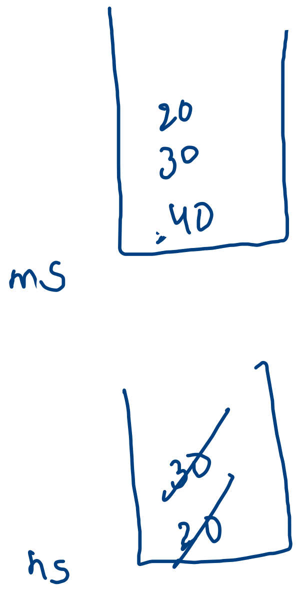
q. Peek()->10

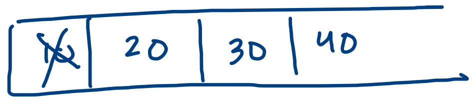
q.add(50)

q.remove()->10

Stack To Queue Adapter - Remove Efficient







q. add(10)

q.add (20)

q-add (30)

q. remove() → 10

q. add(40)