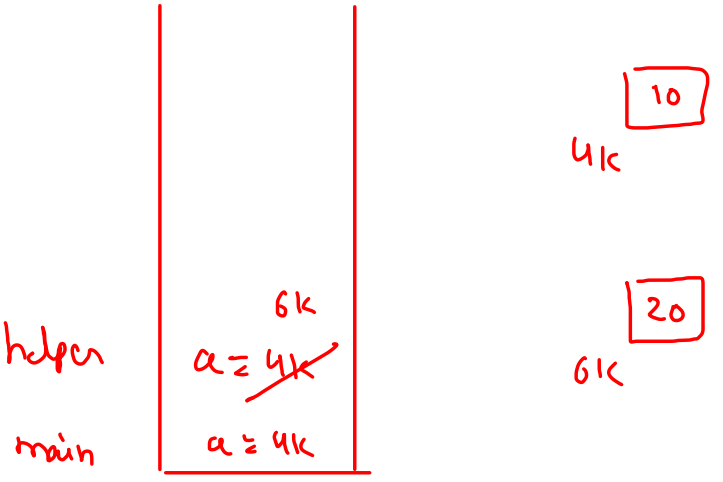
memory :

```
class Main {

    p s v main() {
        int[] a = new int[1];
        a[0] = 10;
        helper(a);
        syso(a[0]);
    }

    void helper(int[] a) {
        a = new int[1];
        a[0] = 20;
    }
}
```

copy

Java is always pass by value.



Classes and Objects

↓ data type

↓ variable

```
Student {
    data member:
        int sn;
        String name;
    member function:
        void print_details ( ) {
            syso( sn + " " + name);
        }
}
```

```
Student s1 = new Student( );
```

(i) object creation on heap.

(ii) parsing.

(iii) constructor calling.

```java
public static class Student {
    //data members
    int sn;
    String name;

    //member function
    void print_details() {
        System.out.println(sn + " " + name);
    }

    void update_name(String new_name) {
        name = new_name;
    }
}

public static void main(String[]args) {
    Student s1 = new Student();

    s1.sn = 1;
    s1.name = "abc";

    Student s2 =  new Student();

    s2.sn = 2;
    s2.name = "xyz";

    s1.print_details();
    s2.print_details();

}
```

this

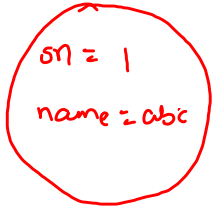
main

S2 = 8K

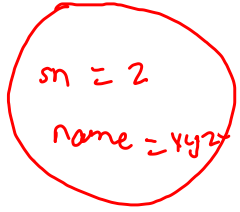
S1 = 4K

1 abc

2 xyz



4K



8K

Student func:

print_details();

update_name(new_name);

```java
public static class Student {
    //data members
    int sn;
    String name;

    //member function
    void print_details() {
        System.out.println(sn + " " + name);
    }

    void update_name(String new_name) {
        name = new_name;
    }

    //constructor

    //1. default constructor
    Student() {

    }

    //2. paremetirised constructor
    Student(int sn,String name) {
        this.sn = sn;
        this.name = name;
    }

}
```

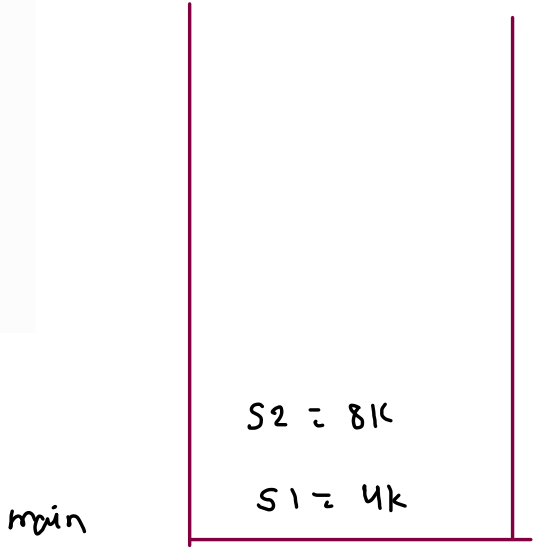
```java
public static void main(String[]args) {
    Student s1 = new Student(1,"abc");

    Student s2 =  new Student();

    s2.sn = 2;
    s2.name = "xyz";

    s1.print_details();
    s2.print_details();

}
```

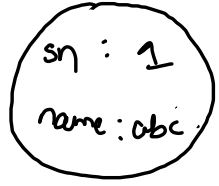
now

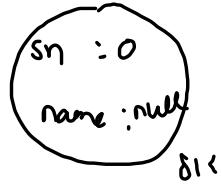
(i) object creation on heap.

(ii) parsing.

(iii) constructor calling.







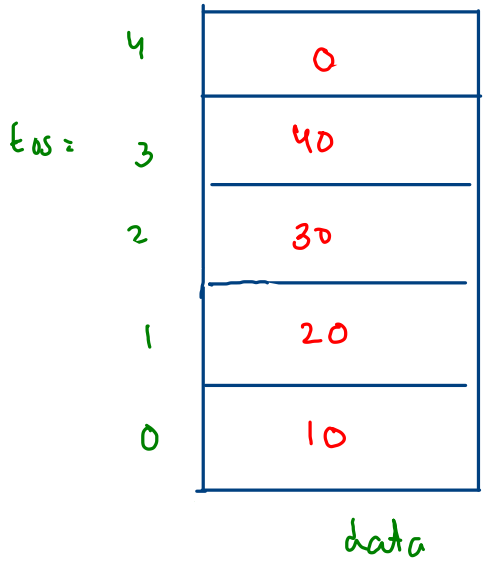

```java
//member function
void print_details() {
    System.out.println(sn + " " + name);
}

void update_name(String new_name) {
    name = new_name;
}
```

Normal Stack

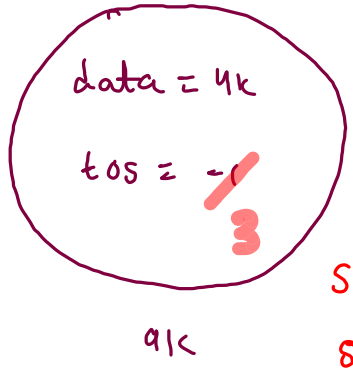
```
CustomStack st = new CustomStack(n);
```

5

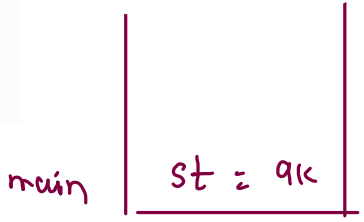
```
public static class CustomStack {
  int[] data;
  int tos;

  public CustomStack(int cap) {
    data = new int[cap];
    tos = -1;
  }

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

  void display() {
    // write ur code here
  }

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

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

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



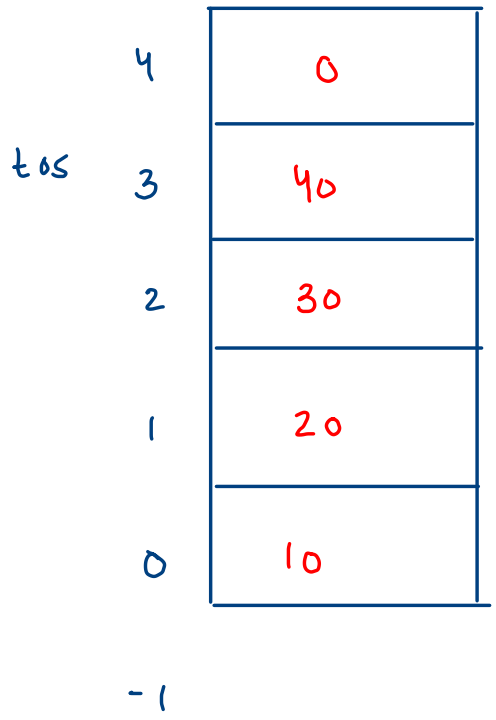
cap = 5



st.push(10);
st.push(20);
st.top() → 20
st.push(30)
st.push(40)
st.push(50)
st.push(60) → overflow
st.pop()



```java
int size() {
  return tos+1;

}

void display() {
    for(int i = tos; i >= 0;i--) {
        System.out.print(data[i] + " ");
    }
    System.out.println();
}

void push(int val) {
  if(tos == data.length - 1) {
      System.out.println("Stack overflow");

  }
  else {
      tos++;
      data[tos] = val;
  }
}

int pop() {
   if(tos == -1) {
       System.out.println("Stack underflow");
       return -1;
   }

   int val = data[tos];
   data[tos] = 0;
   tos--;

   return val;
}

int top() {
   if(tos == -1) {
       System.out.println("Stack underflow");
       return -1;
   }

   return data[tos];
}
```



st.push(10) cap = 5

st.push(20)

st.size() → 2

st.push(30)

st.top() → 30

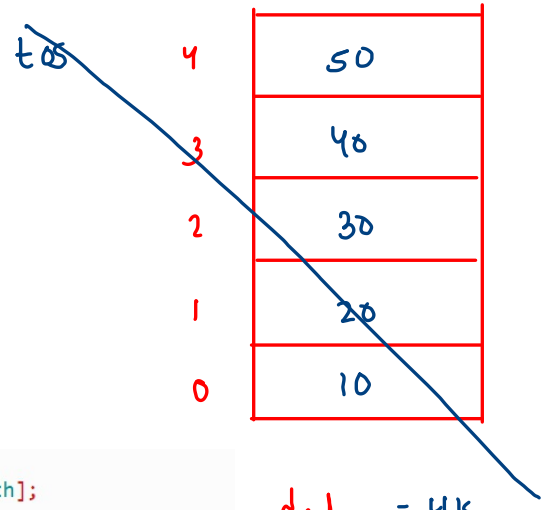
st.push(40)

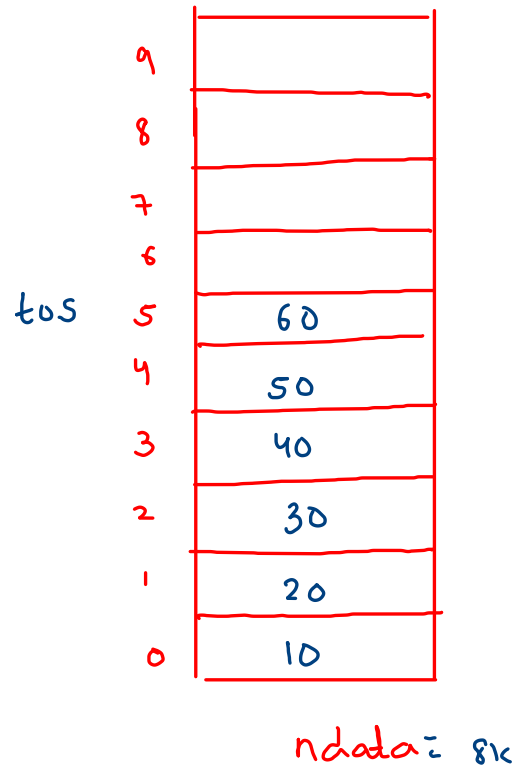
st.push(50)

st.push(60) → overflow

st.pop() → 50

Dynamic stack



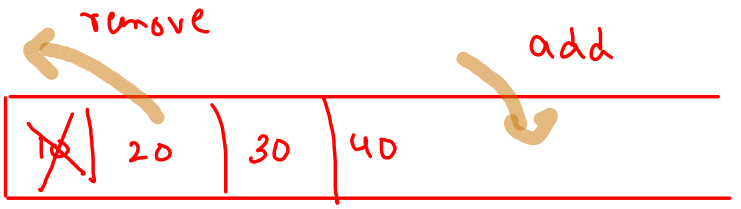



```java
private void resize() {
    int[]new_data = new int[2*data.length];

    //copy content from old data array to new data array
    for(int i=0; i < data.length;i++) {
        new_data[i] = data[i];
    }

    data = new_data;
}
// change the code of this function according to question
void push(int val) {
  if (tos == data.length - 1) {
    resize();
    push(val);
  } else {
    tos++;
    data[tos] = val;
  }
}
```

st.push(60);

Queue

FIFO



q.add (10)

q.add (20)

q.add (30)

q.peek () -> 10

q.add (40)

q.remove () -> 10

Count binary strings

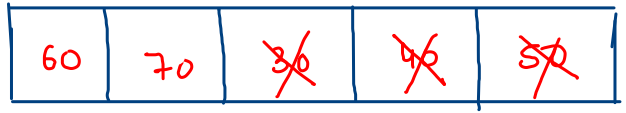
$n = 3$

↳ 3 length binary strings with no consecutive 0's.

Normal Queue

cap = 5

```
public static class CustomQueue {
  int[] data;
  int front;
  int size;

  public CustomQueue(int cap) {
    data = new int[cap];
    front = 0;
    size = 0;
  }

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

  void display() {
    // write ur code here
  }

  void add(int val) {
    // write ur code here
  }

  int remove() {
    // write ur code here
  }

  int peek() {
    // write ur code here
  }
}
```



f

s = 2

Efficient use of space

- add() → rear : (front + size) % data.len
- remove() → f = (f + 1) % data.len

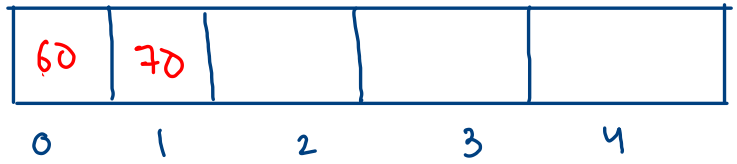
q.add (10)
q.add (20)
q.add (30)
q.remove()
q.add (40)
q.remove()
q.add (50)
q.add (60)
q.add (70)
q.add (80)

q.remove()
q.remove()
q.remove()

```
void display() {
  for(int i=0; i < size;i++) {
      int idx = (i + front) % data.length;
      System.out.print(data[idx] + " ");
  }
  System.out.println();
}

void add(int val) {
   if(size == data.length) {
       System.out.println("Queue overflow");
   }
   else {
       int rear = (front + size) % data.length;
       data[rear] = val;
       size++;
   }

}
int remove() {
   if(size == 0) {
       System.out.println("Queue underflow");
       return -1;
   }
   else {
       int val = data[front];
       data[front] = 0;
       front = (front + 1) % data.length;
       size--;

       return val;
   }
}

int peek() {
   if(size == 0) {
       System.out.println("Queue underflow");
       return -1;
   }
   else {
       return data[front];
   }
}
```

cap = 5

30 40 50 60



f

s = 2

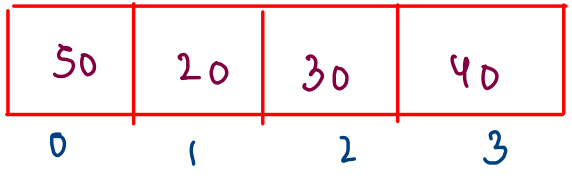
- q.add (10) q.remove()
- q.add (20) q.remove()
- q.add (30) q.remove()
- q.remove()
- q.add (40)
- q.remove()
- q.add (50)
- q.add (60)
- q.add (70)
- q.add (80) -> 0

```
private void resize() {
    int[]new_data = new int[2*data.length];

    for(int i=0; i < data.length;i++) {
        int idx = (i + front) % data.length;

        new_data[i] = data[idx];
    }

    data = new_data;
    front = 0;
}
```

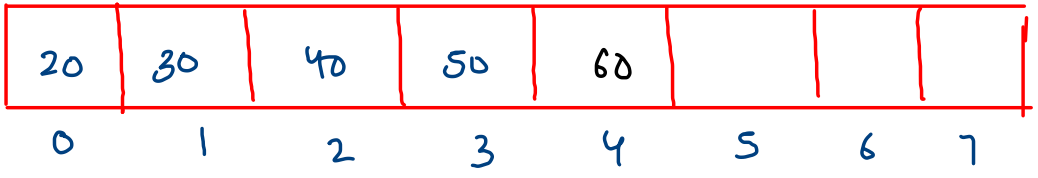
q.add(60)

cap = 4

data



new_data



s = 4

s = 5

i	idx
0	1
1	2
2	3
3	0

idx -> (i + 1) % 4