Classes and Objects

Reference variable

(i) arr, al, stack variable

Primitive variable

(i) int → short, long, byte

(ii) float, double

(iii) char

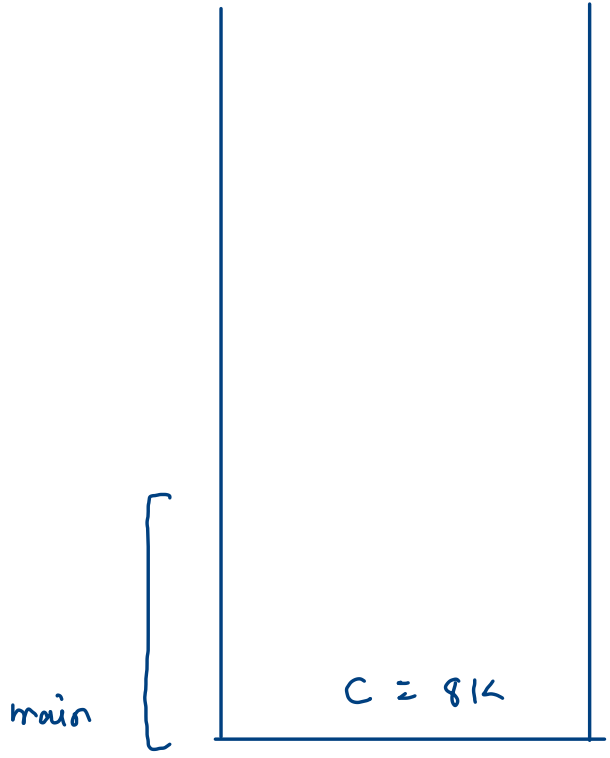
(iv) boolean

```
Car {
    String model;
    double speed;
    int id;

    check ( ) {
    }
    get ( ) {
    }
    print ( ) {
    }
}
```

data members

member function

```
Car c = new Car();
```



(i) memory creation on heap

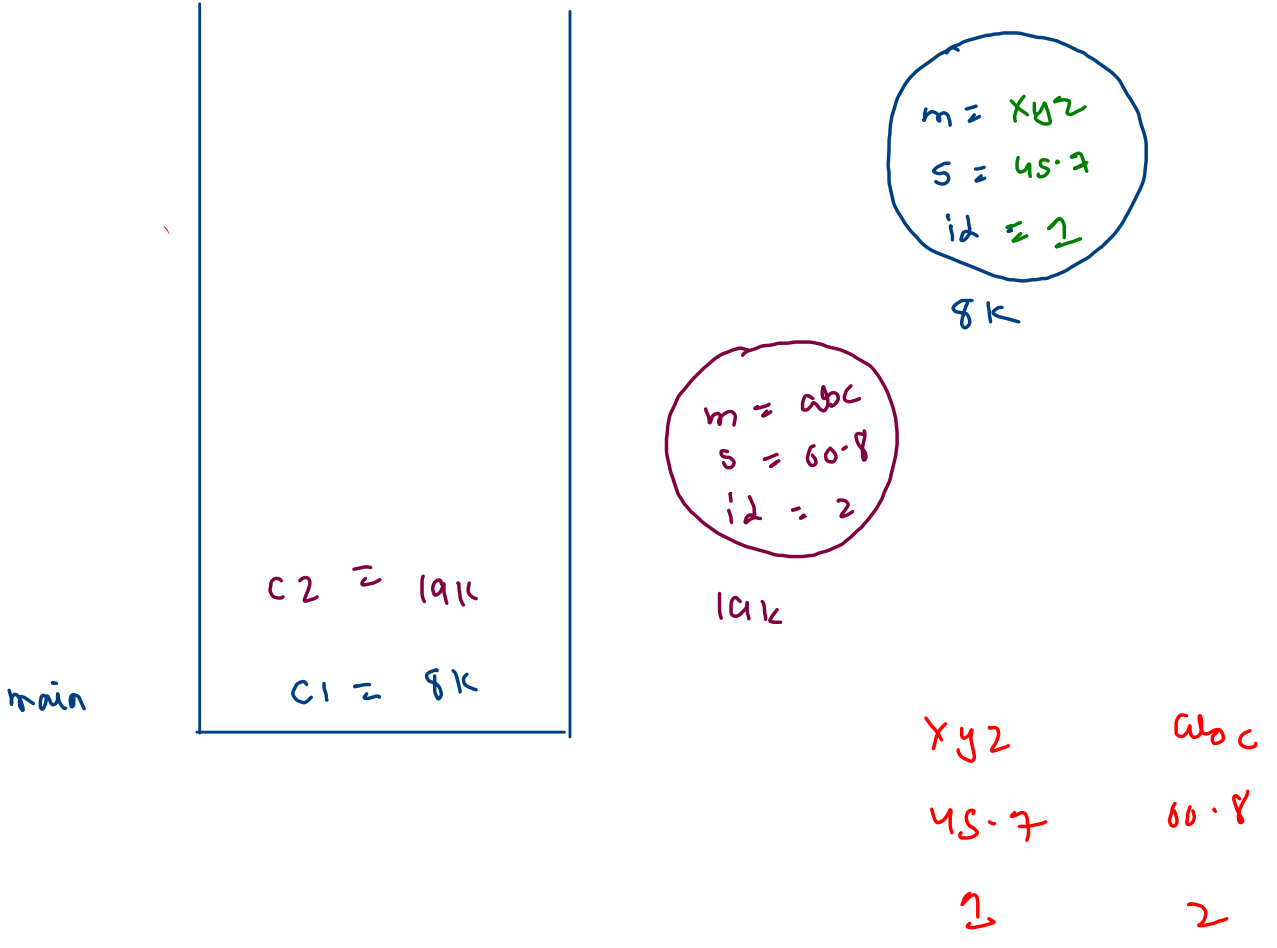
model = null
speed = 0.0
id = 0

81K

(ii) Parsing

(iii) constructor calling

```java
public static class Car {
    //data members
    String model;
    double speed;
    int id;

    //member function
    void print() {
        System.out.println("Model " + this.model);
        System.out.println("Speed " + this.speed);
        System.out.println("Id " + this.id);
    }

    boolean check() {
        if(this.speed < 50) {
            return false;
        }
        else {
            return true;
        }
    }

    Car inc_speed() {
        this.speed += 2;
        return this;
    }
}

public static void main(String[]args) {
    Car c1 = new Car();
    c1.model = "xyz";
    c1.speed = 45.7;
    c1.id = 1;

    c1.print();

    Car c2 = new Car();
    c2.model = "abc";
    c2.speed = 60.8;
    c2.id = 2;

    c2.print();

}
```



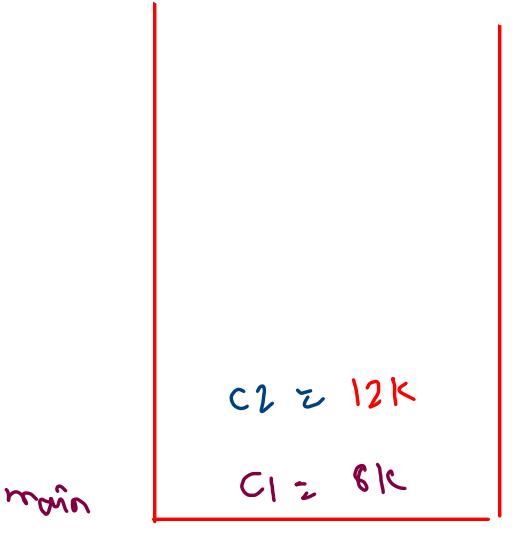
```java
public static class Car {
    //data members
    String model;
    double speed;
    int id;

    //member function
    void print() {
        System.out.println("Model " + this.model);
        System.out.println("Speed " + this.speed);
        System.out.println("Id " + this.id);
    }

    boolean check() {
        if(this.speed < 50) {
            return false;
        }
        else {
            return true;
        }
    }

    Car inc_speed() {
        this.speed += 2;
        return this;
    }

    //constructors
    //default constructor
    Car() {

    }

    //parameterised constructor
    Car(int model,int speed,int id) {
        this.model = model;
        this.speed = speed;
        this.id = id;
    }
}

public static void main(String[]args) {
    Car c1 = new Car();
    c1.model = "xyz";
    c1.speed = 45.7;
    c1.id = 1;

    c1.print();

    Car c2 = new Car("abc",60.8,2);

    c2.print();
```

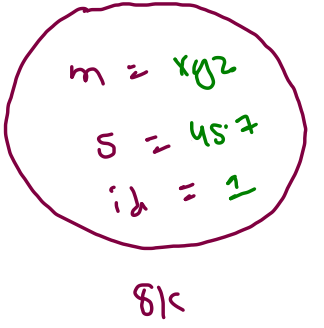
sole 0.103sec 11768kb Accepted

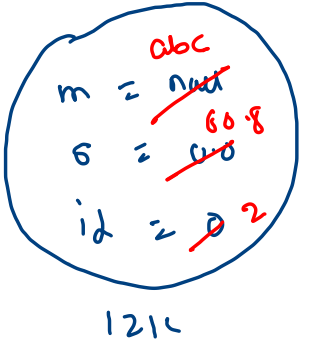
(i) memory creation on heap

(ii) parsing

(iii) constructor calling

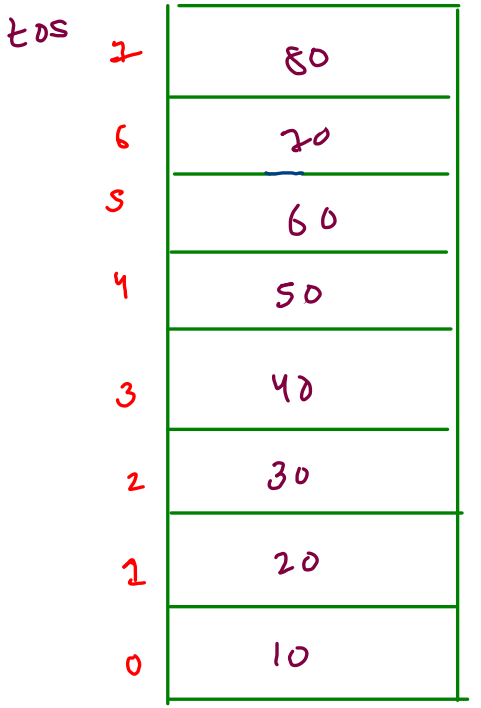






```java
public static class CustomStack {
  int[] data;
  int tos;

  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
  }
```

```java
public static void main(String[] args) throws Exception {
  BufferedReader br = new BufferedReader(new InputStreamReader(S
  int n = Integer.parseInt(br.readLine());
  CustomStack st = new CustomStack(n);

  String str = br.readLine();
  while(str.equals("quit") == false){
    if(str.startsWith("push")){
      int val = Integer.parseInt(str.split(" ")[1]);
      st.push(val);
    } else if(str.startsWith("pop")){
      int val = st.pop();
      if(val != -1){
        System.out.println(val);
      }
    } else if(str.startsWith("top")){
      int val = st.top();
      if(val != -1){
        System.out.println(val);
      }
    } else if(str.startsWith("size")){
      System.out.println(st.size());
    } else if(str.startsWith("display")){
      st.display();
    }
    str = br.readLine();
```



data = 21k

tos = -1

15k

st = 15k .

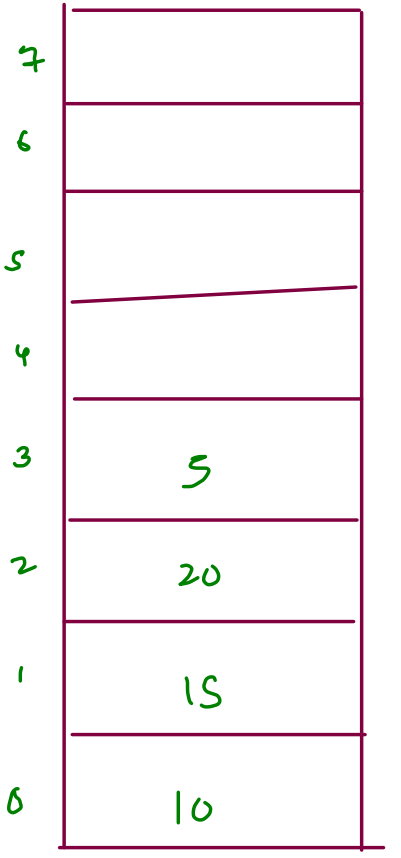
st.push(50)

```java
void push(int val) {
  if (tos == data.length - 1) {
    System.out.println("Stack overflow");
  } else {
    tos++;
    data[tos] = val;
  }
}

int pop() {
```

data = 191k

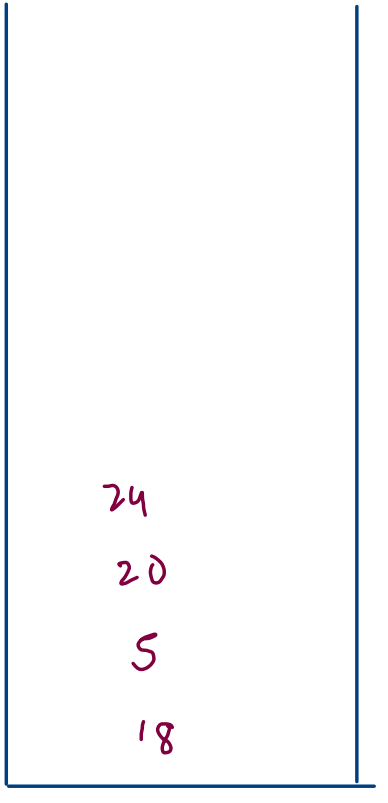
tos = 3

151k

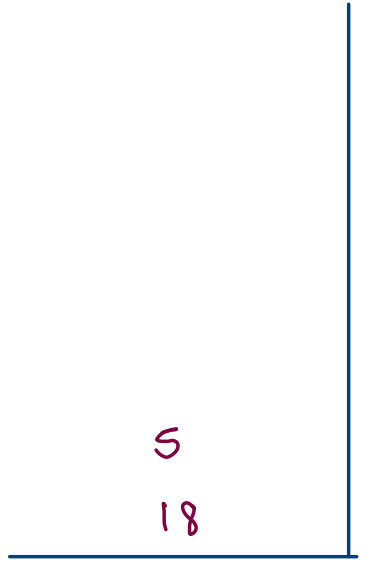


```java
void push(int val) {
    if(allData.size() == 0) {
        allData.push(val);
        minData.push(val);
    }
    else {
        allData.push(val);
        if(val <= minData.peek()) {
            minData.push(val);
        }
    }
}

int pop() {
    if(allData.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }
    else {
        int val = allData.pop();

        if(val == minData.peek()) {
            minData.pop();
        }

        return val;
    }
}

int top() {
    if(allData.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }
    return allData.peek();
}

int min(){
    if(allData.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }
    return minData.peek();
}
```

18 5 20 24 2 19 6 2 15 ~~2~~



all Data



min Data

18 20 5 24 2 19 6 2 15 1

min = 2

-1
24
-8
20
18

(i) val + (val - omin)

val - omin = k (k < 0) (val < omin)

$\boxed{\text{val} > \text{val} + k}$

(ii) st.peek() -> fake value

real value -> min

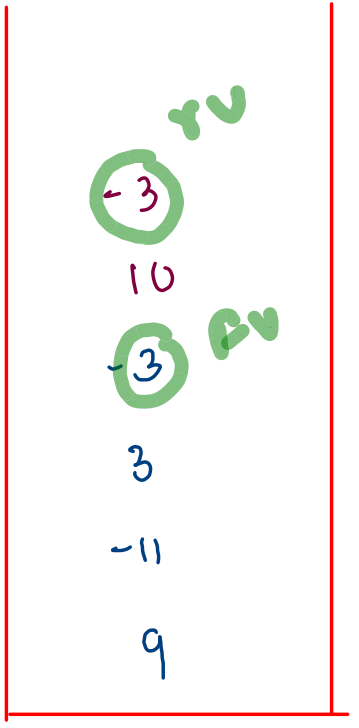
rval = min

omin = rval + rval - st.peek()

val + (val - omin) = -8

5 + (5 - omin) = -8

omin = 18



val - min

min = -5

encoded = -5 - (-2)

= -3

val - min < 0

val > val - min

(wrong in -ve numbers)

9

-2

3

-5

10

-3

normal queue

cap = 8

size

add

remove

peek

110	120			70	80	90	100
0	1	2	3	4	5	6	7

F

i -> j to F+S

4 -> 4

5 -> 5

6 -> 6

7 -> 7

8 -> 0

9 -> 1

Size = 6

remove

$F = (F+1) \% cap$

add

$r = (F + size) \% cap$

$= (2 + 7) \% 8 = 1$

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

```java
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 = 8

90	100						
0	1	2	3	4	5	6	7

f

size = 3

f+s =<10

cap = 4

50	60	30	40
0	1	2	3
		F	

30	40	50	60				
0	1	2	3	4	5	6	7
F							

size = 4

i	nd	od
0	0	2
1	1	3
2	2	0
3	3	1

```
for( int i=0; i<cap; i++) {

    ndata[i] = data [(i+front)%cap];

}
```