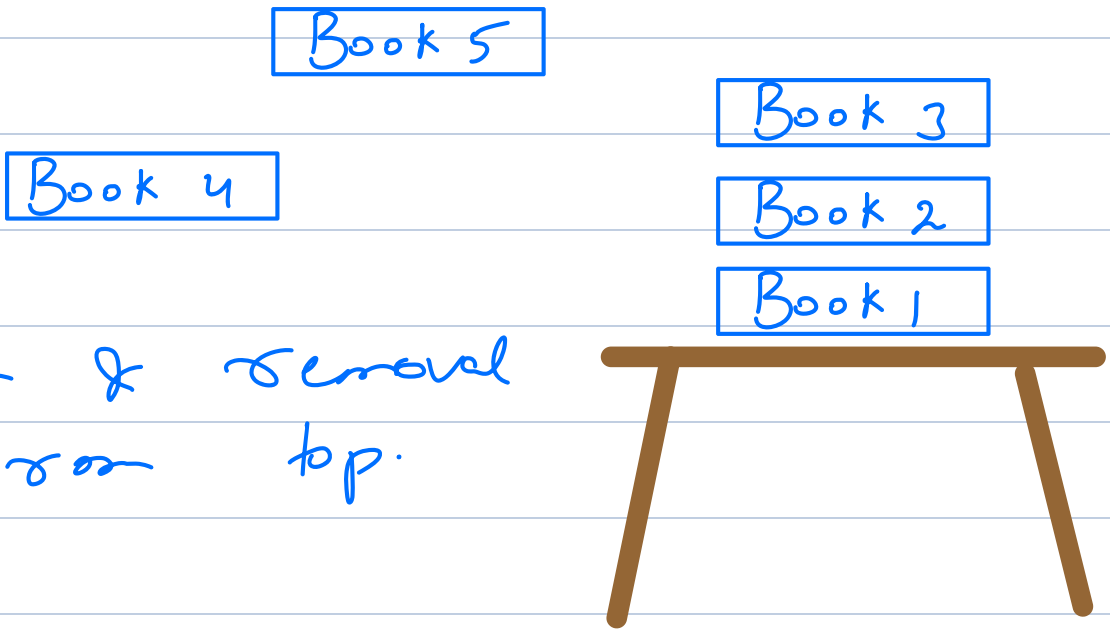
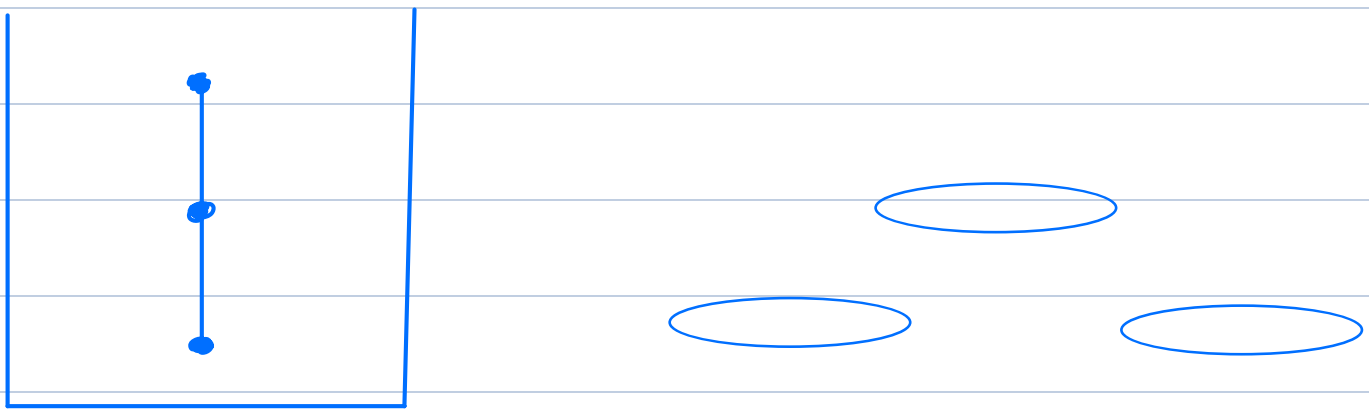
Stack.



→ Insertion & removal only from top.



Idli Maker.

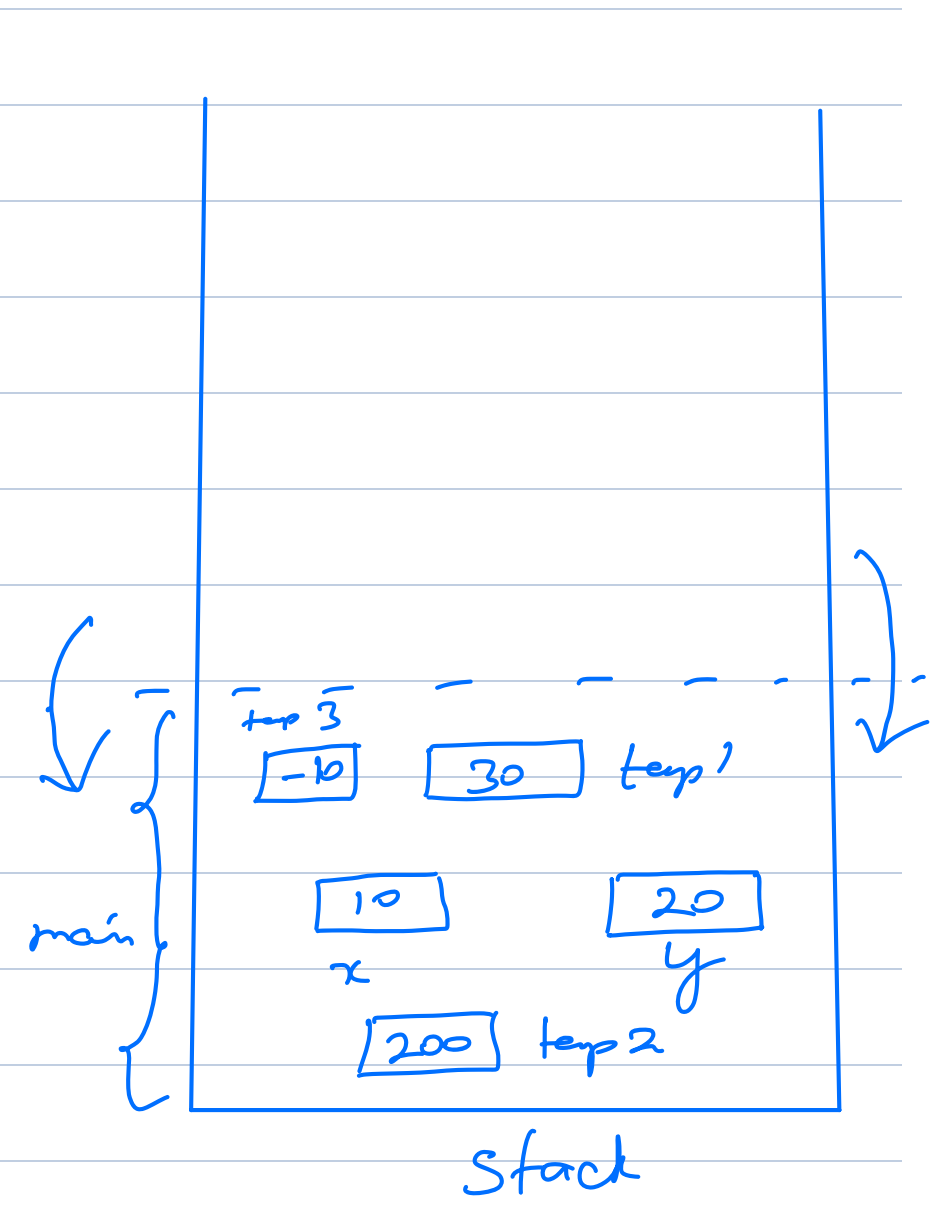
→ Last In First Out

Call Stack

o/p :- -10.

```
int add(int x, int y) {
    return x + y;
}

int product(int x, int y) {
    return x * y;
}

int subtract(int x, int y) {
    return x - y;
}

public static void main() {
    int x = 10;
    int y = 20;
    int temp1 = add(x, y);
    int temp2 = product(x, y);
    int temp3 = subtract(x, y);
    System.out.println(temp3);
}
```

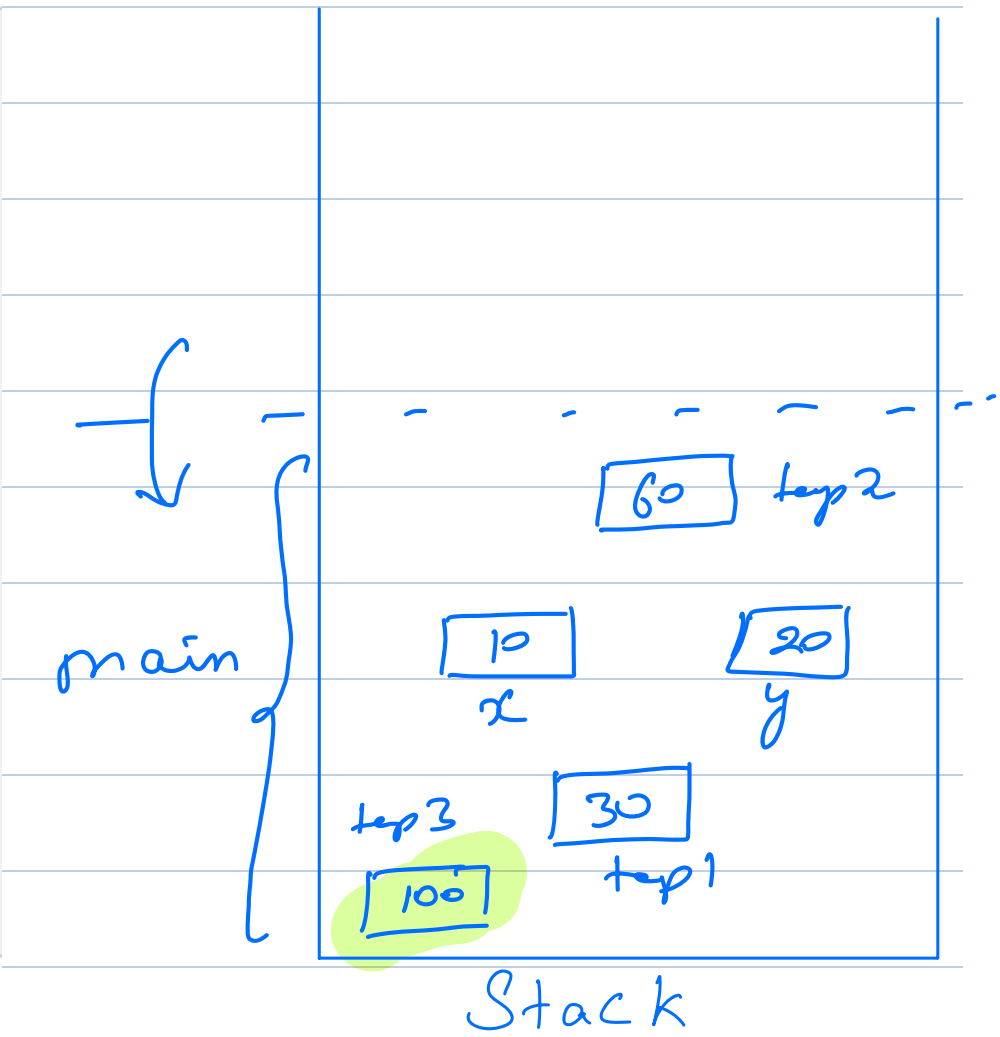
-10.



o/p → 100.

(2)

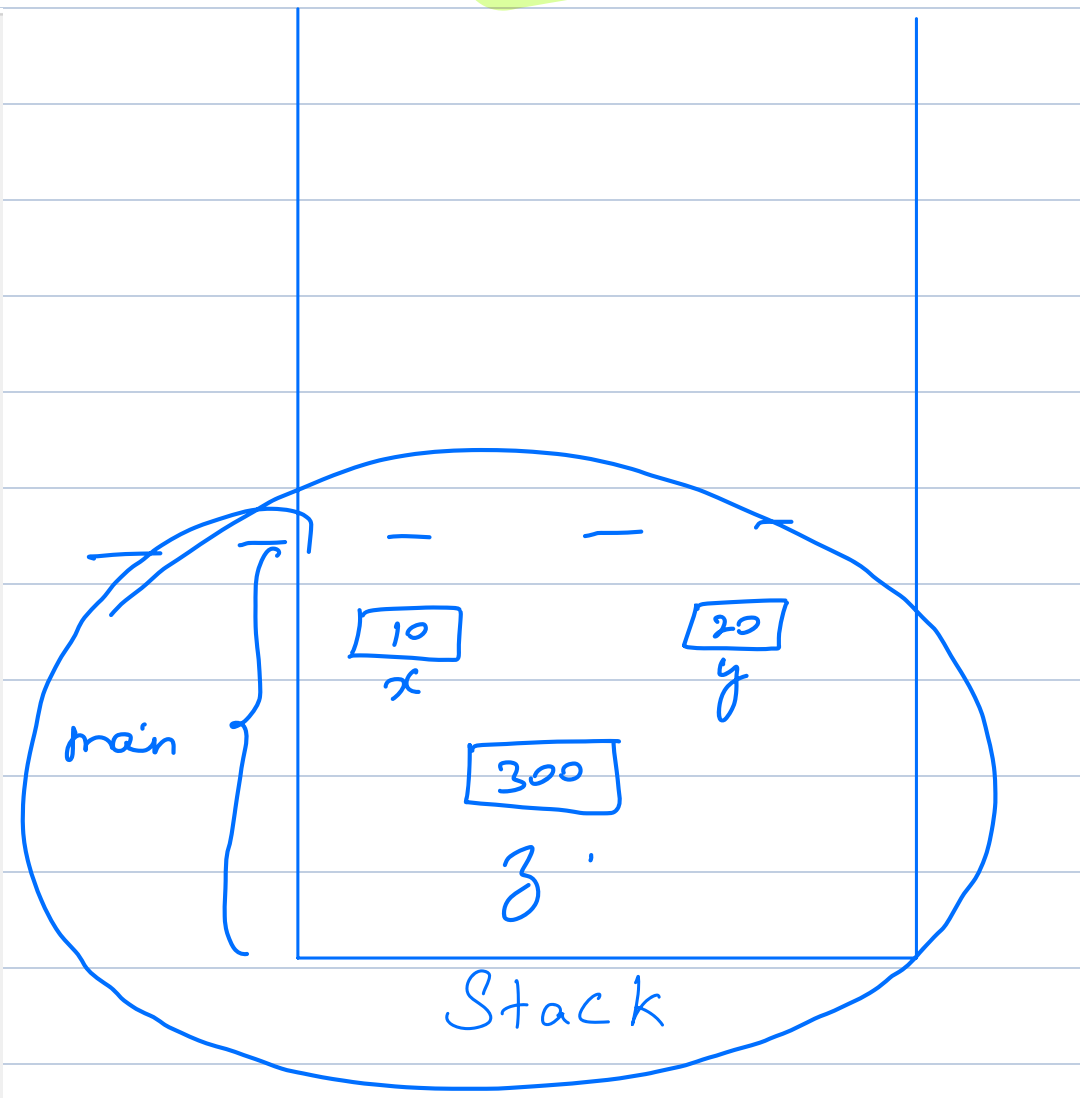
```
int add(int x, int y) {
    return x + y;
}

public static void main() {
    int x = 10;
    int y = 20;
    int temp1 = add(x, y);
    int temp2 = add(temp1, 30);
    int temp3 = add(temp2, 40);
    System.out.println(temp3);
}
```



3

o/p → 300
Hello
300

```
int add(int x, int y) {
    return x + y;
}

static int fun(int a, int b) {
    int sum = add(a, b);
    int ans = sum * 10;
    return ans;
}
static void extra(int w){
    System.out.println("Hello");
    System.out.println(w);
}
public static void main() {
    int x = 10;
    int y = 20;
    int z = fun(x, y);
    System.out.println(z);
    extra(z);
}
```



Types of Memory in Java

(1) Stack → (1) Function Calls
(2) Primitive data type
(3) Reference variables of my container

(2) Heap → Actual container of that reference Variable is stored.
Array, Arraylist, Objects.
(Actual instance)

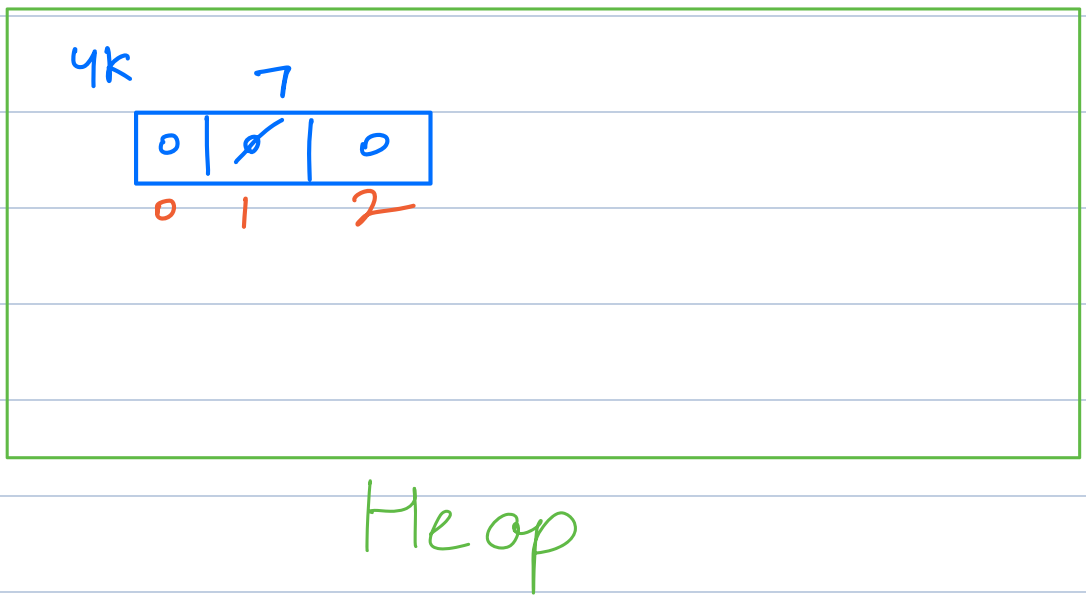
*(a + 2*4).

o/p

4K.
0

Ex - 1.

```
public static void main() {
    int x = 10;
    int[] ar = new int[3];
    System.out.println(ar); // #ad1
    System.out.println(ar[2]); // 0
    ar[1] = 7;
}
```

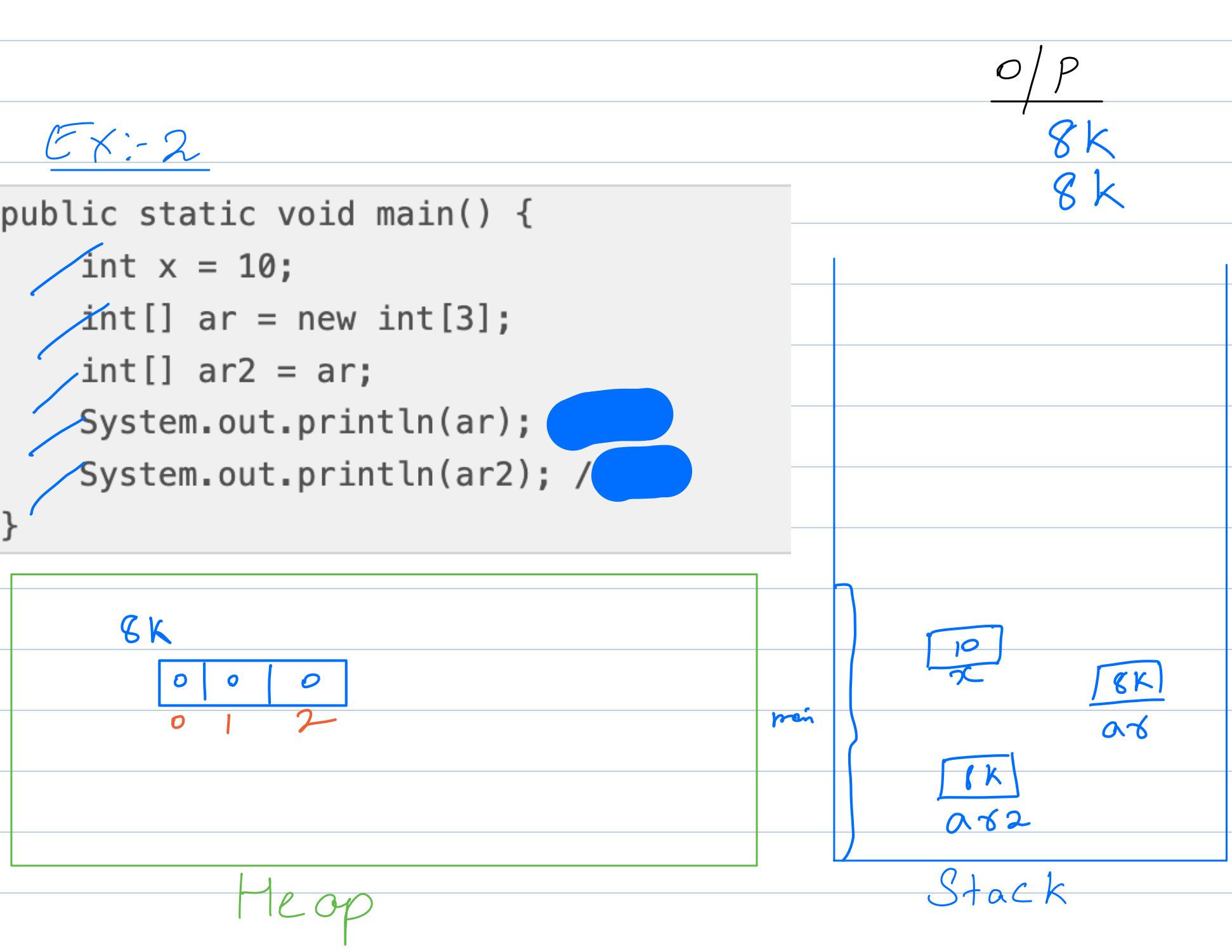


o/p

8k
8k

Ex:- 2

```
public static void main() {
    int x = 10;
    int[] ar = new int[3];
    int[] ar2 = ar;
    System.out.println(ar);
    System.out.println(ar2); /
}
```

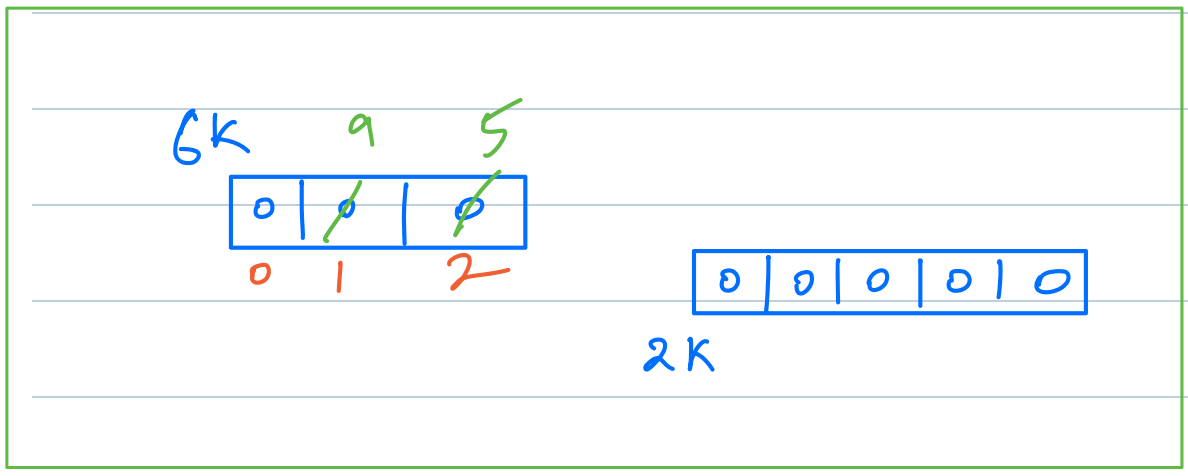


Ex:- 3.

O/P

6K
2K

```
public static void main() {
    int[] ar = new int[3];
    System.out.println(ar);
    ar[1] = 9;
    ar[2] = 5;

    ar = new int[5];
    System.out.println(ar);
}
```

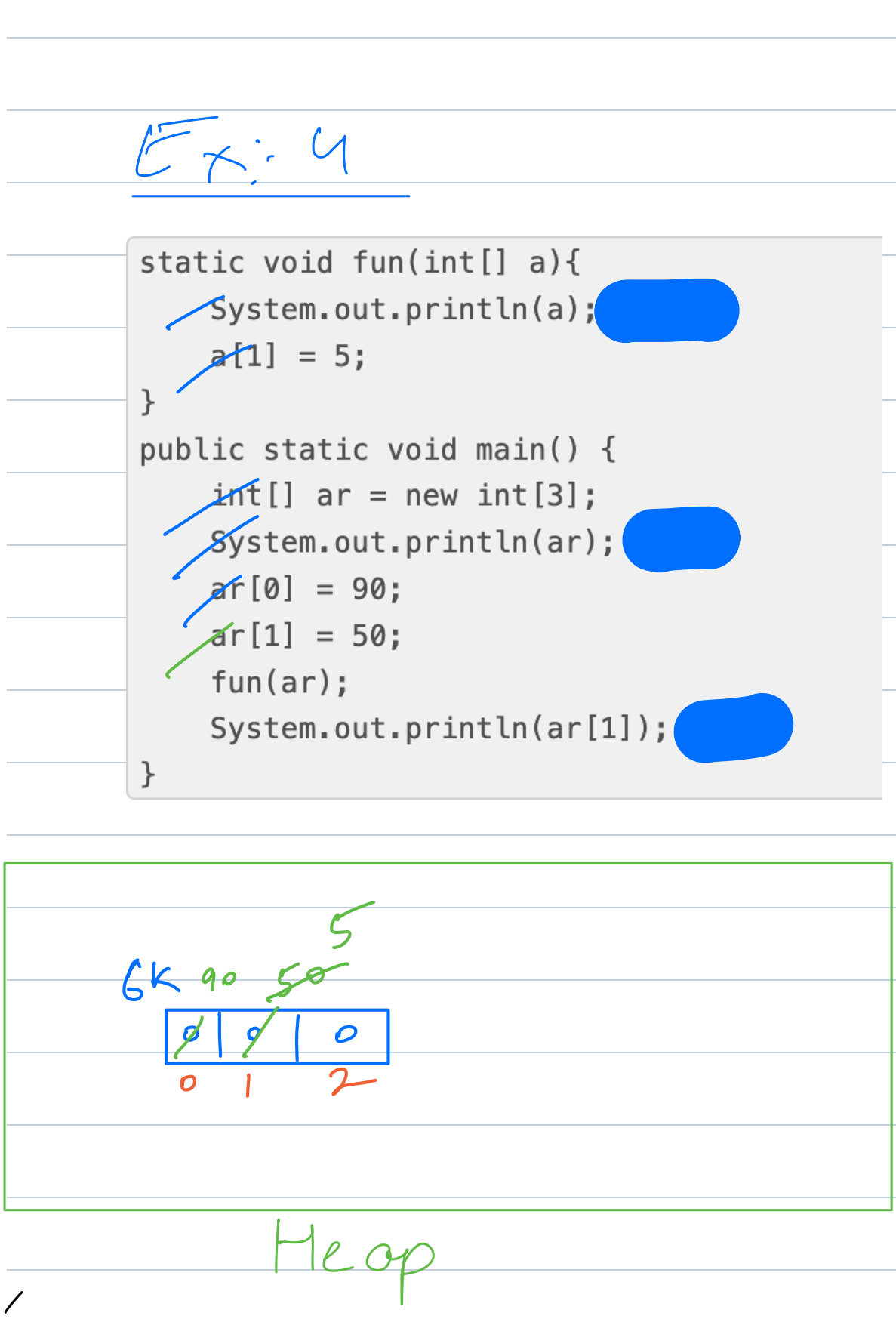


Ex: 4

O/P

6K
6K
5

```
static void fun(int[] a){
    System.out.println(a);
    a[1] = 5;
}
public static void main() {
    int[] ar = new int[3];
    System.out.println(ar);
    ar[0] = 90;
    ar[1] = 50;
    fun(ar);
    System.out.println(ar[1]);
}
```



Ex:- 5

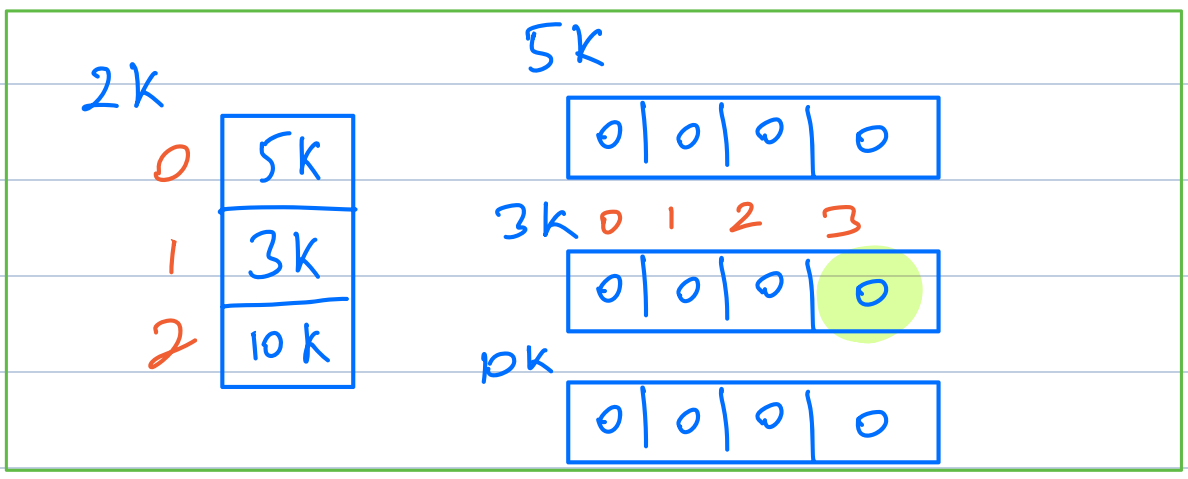
```
public static void main() {
    float y = 7.84f;
    int[][] mat = new int[3][4];
    System.out.println(mat); //
    System.out.println(mat[1]); //
    System.out.println(mat[1][3]); //
}
```

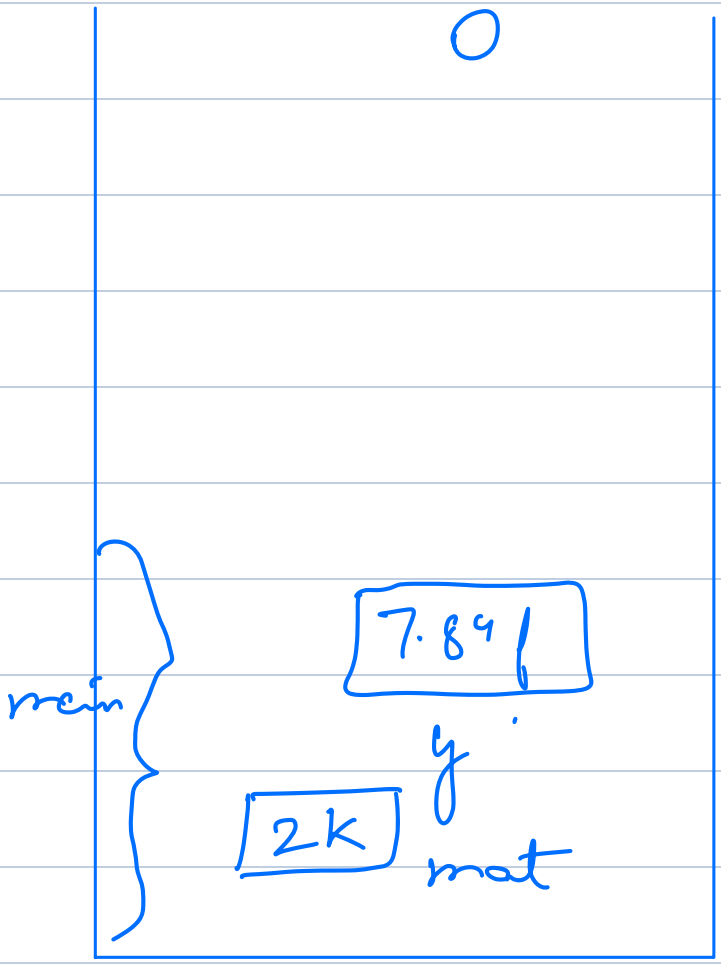
o/p

2k

3k

0





Ex:- 6

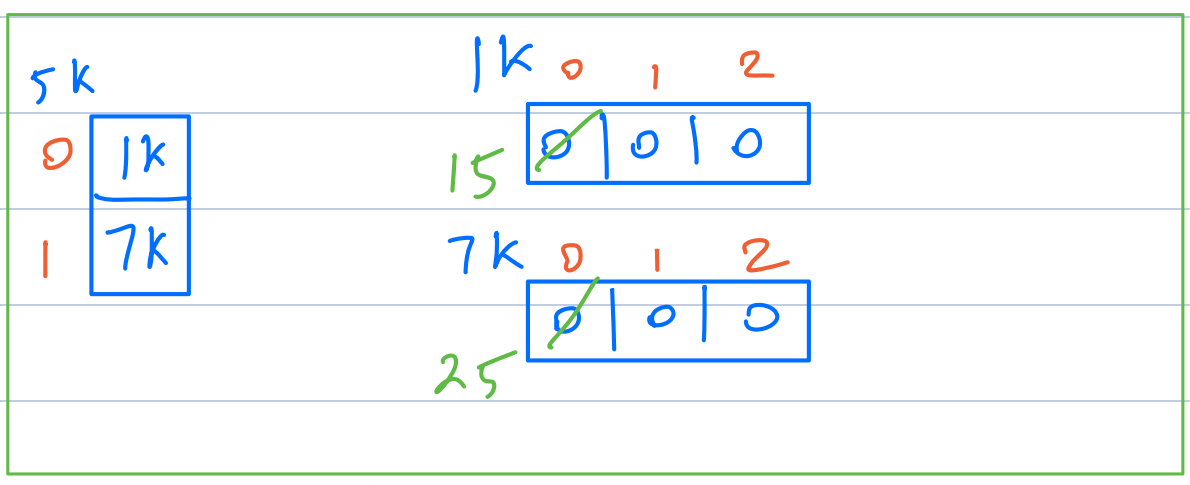
```
static void sum(int[][] mat){
    System.out.println(mat); //
    System.out.println(mat[0][0] + mat[1][0]); //
}
public static void main() {
    int[][] mat = new int[2][3];
    mat[0][0] = 15;
    mat[1][0] = 25;
    sum(mat);
}
```

15 + 25

o/p

5k

40



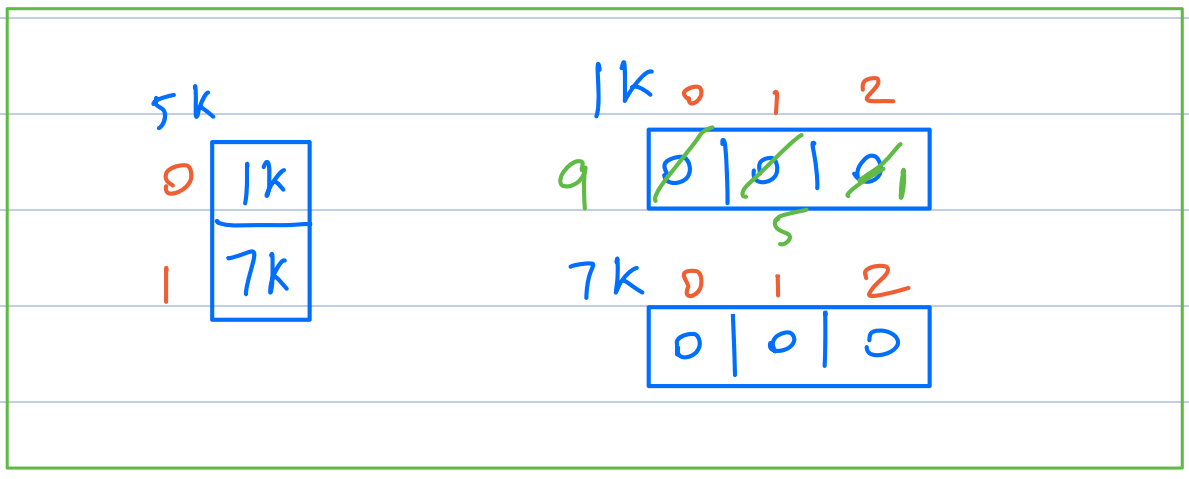
Ex: 7

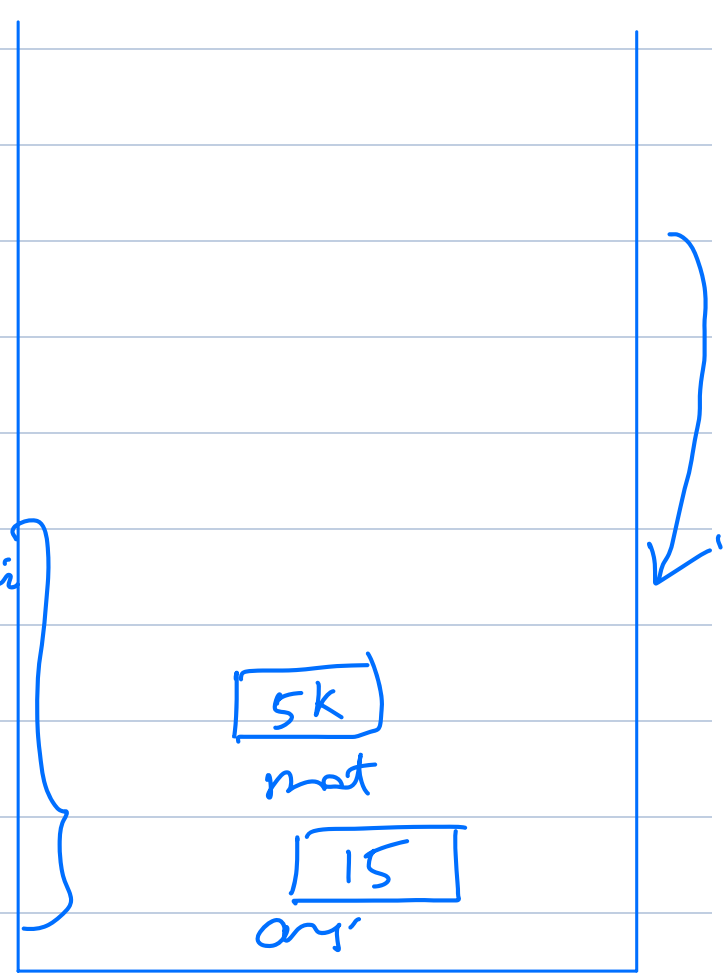
```
static int sumOfRow(int[] arr){
    System.out.println(arr); //
    int sum = 0;
    for (int i = 0; i < arr.length; i++){
        sum = sum + arr[i];
    }
    return sum;
}

public static void main() {
    int[][] mat = new int[2][3];
    mat[0][0] = 9;
    mat[0][1] = 5;
    mat[0][2] = 1;
    int ans = sumOfRow(mat[0]); //
    System.out.println(ans); //
}
```

O / P

1K

15.





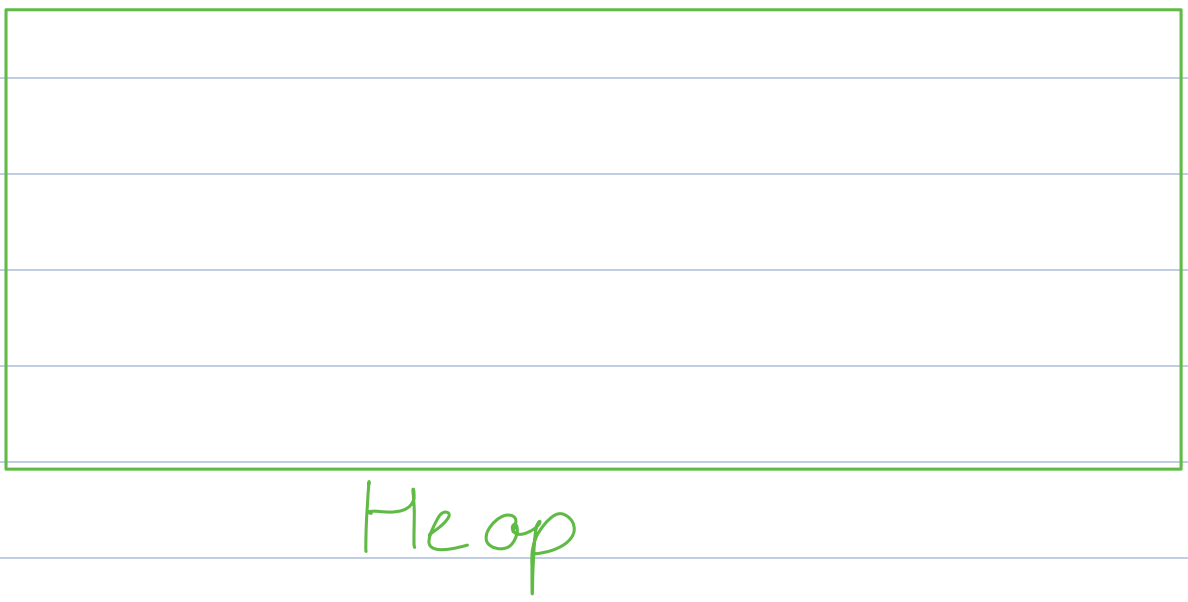
8:50.

Quizzes

o/p — 7 10 .

Q-1.

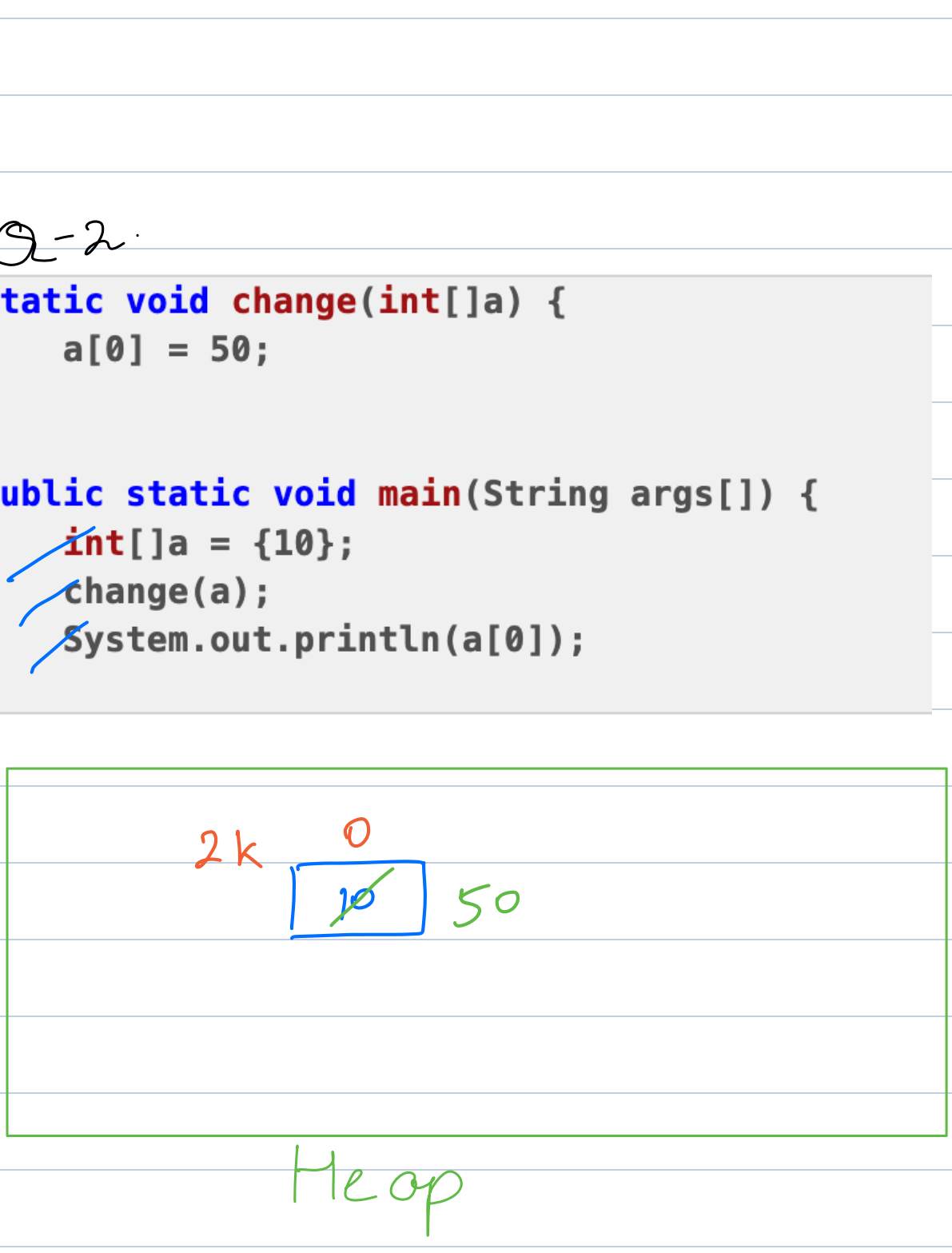
```
static void change(int a) {
    a = 50;
}

public static void main(String args[]) {
    int a = 10;
    change(a);
    System.out.println(a);
}
```



o/p — 50

Q-2.

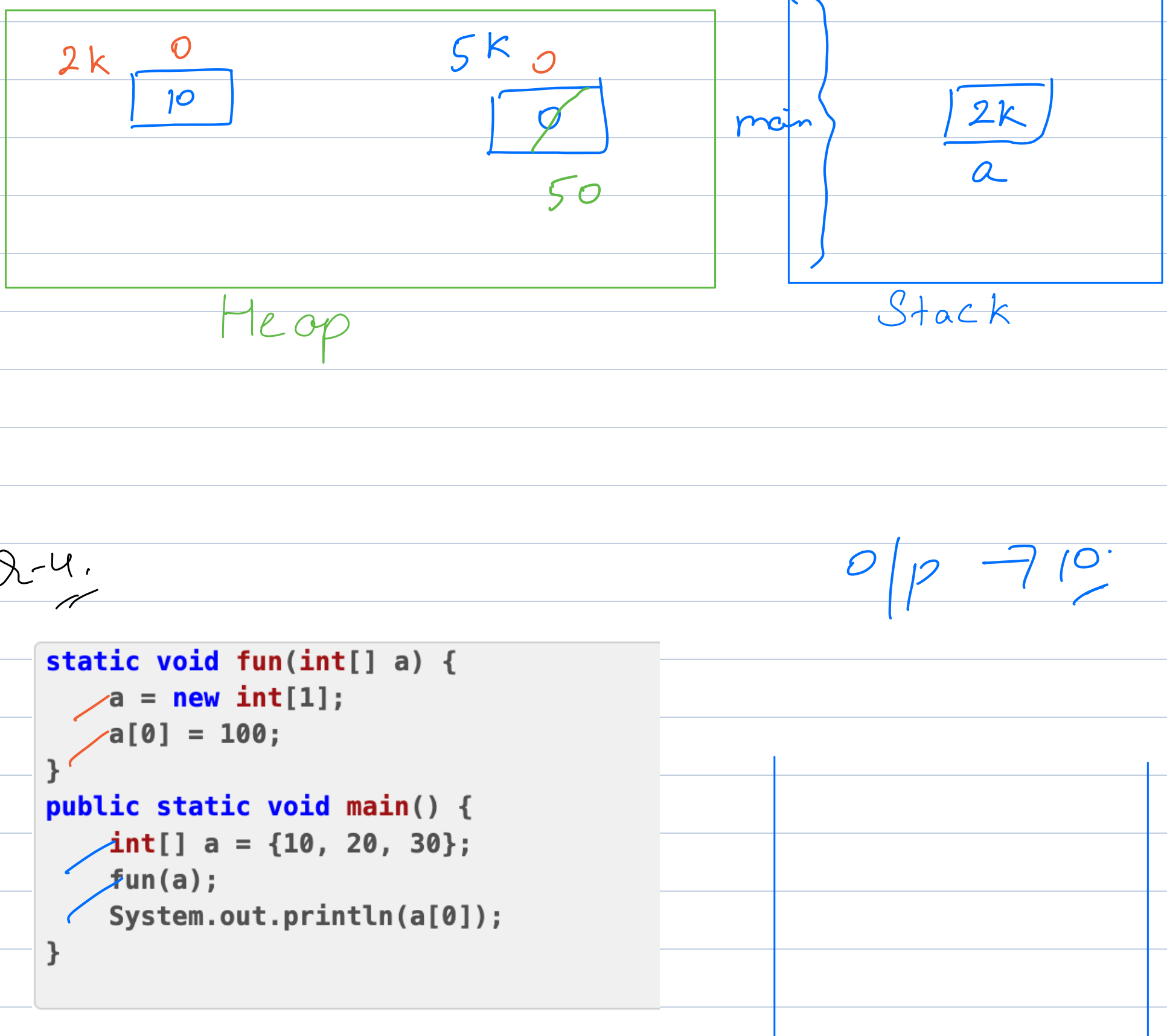
```
static void change(int[]a) {
    a[0] = 50;
}

public static void main(String args[]) {
    int[]a = {10};
    change(a);
    System.out.println(a[0]);
}
```



Q-3.

o/p - 10:

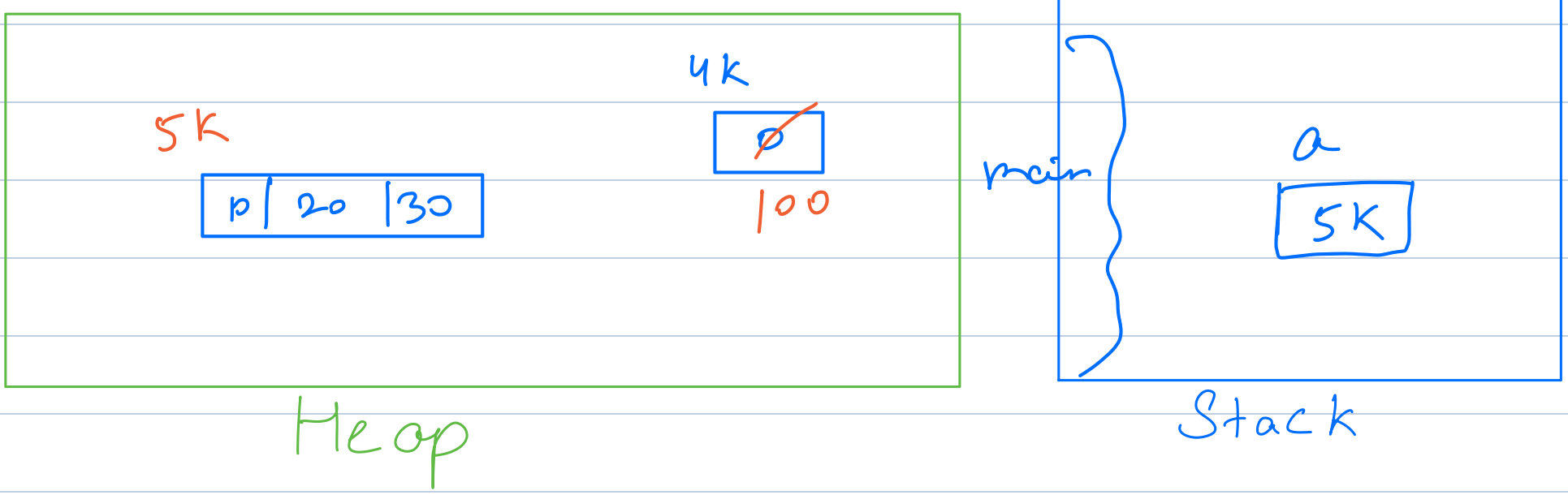
```
static void test(int[]a) {
    a = new int[1];
    a[0] = 50;
}

public static void main(String args[]) {
    int[]a = {10};
    test(a);
    System.out.println(a[0]);
}
```



Q-4.

o/p - 710:

```
static void fun(int[] a) {
    a = new int[1];
    a[0] = 100;
}
public static void main() {
    int[] a = {10, 20, 30};
    fun(a);
    System.out.println(a[0]);
}
```

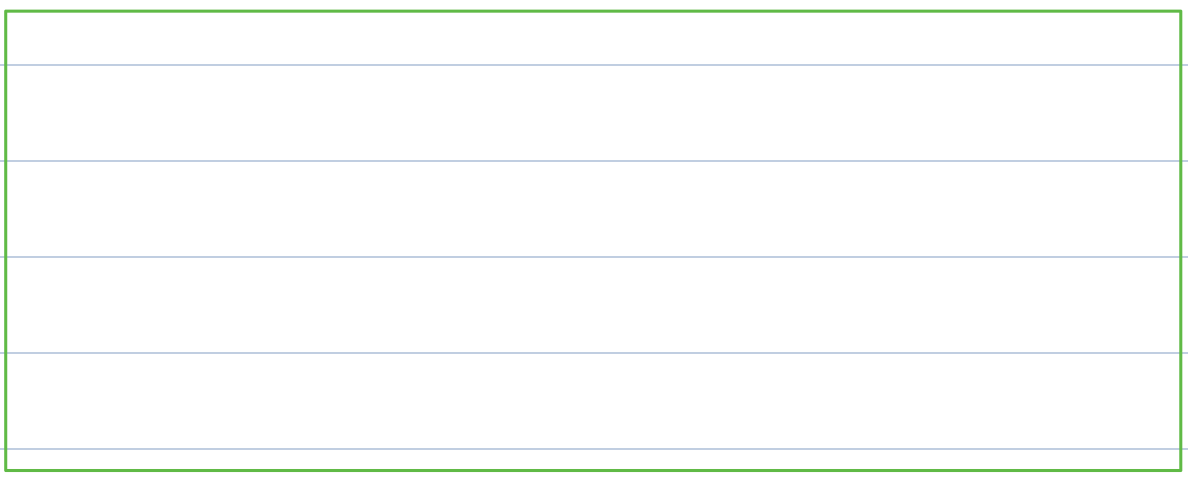


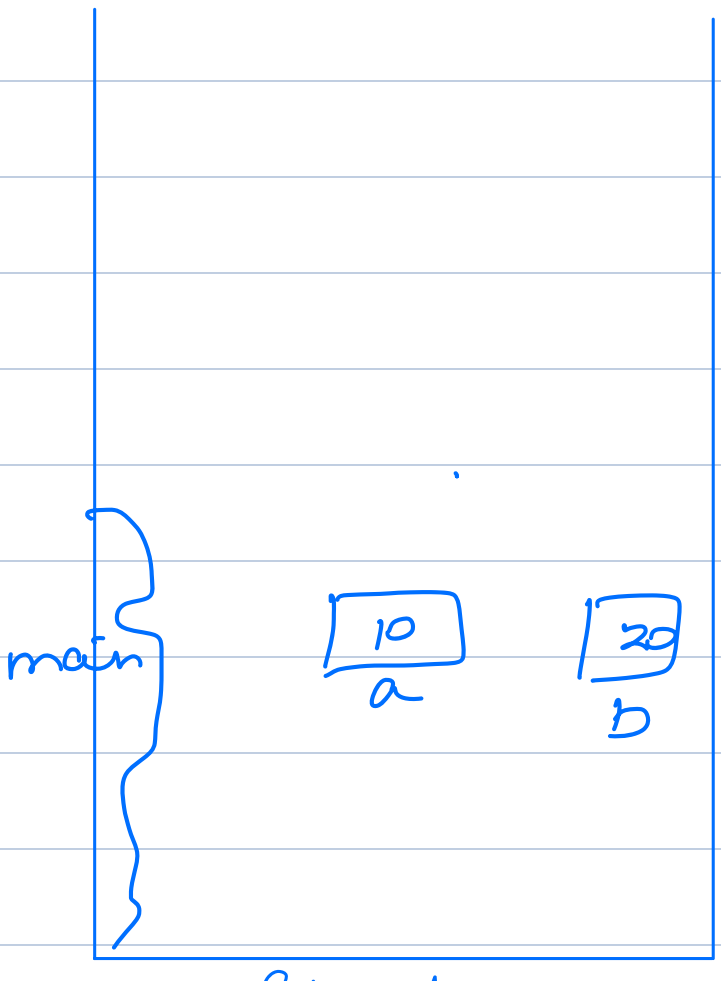
Q-5

O/P

10 20

```java
static void swap(int a,int b) {
    int temp = a;
    a = b;
    b = temp;
}

public static void main(String args[]) {
    int a = 10;
    int b = 20;
    swap(a,b);
    System.out.println(a + " " + b);
}
```



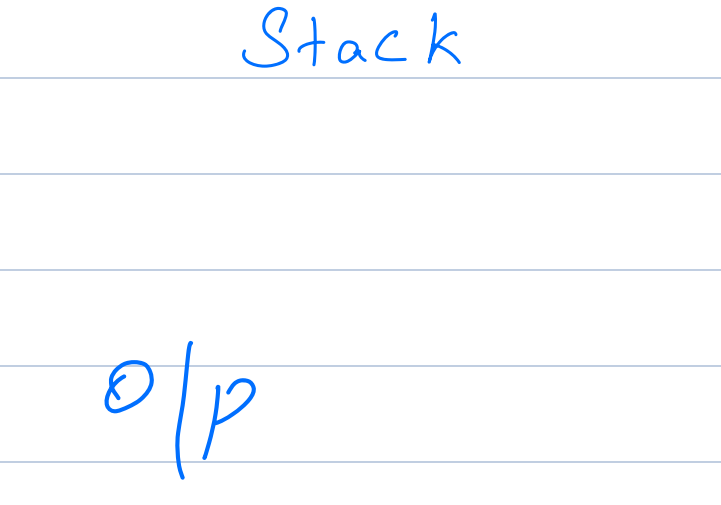


O/P

20 10

Q-6.

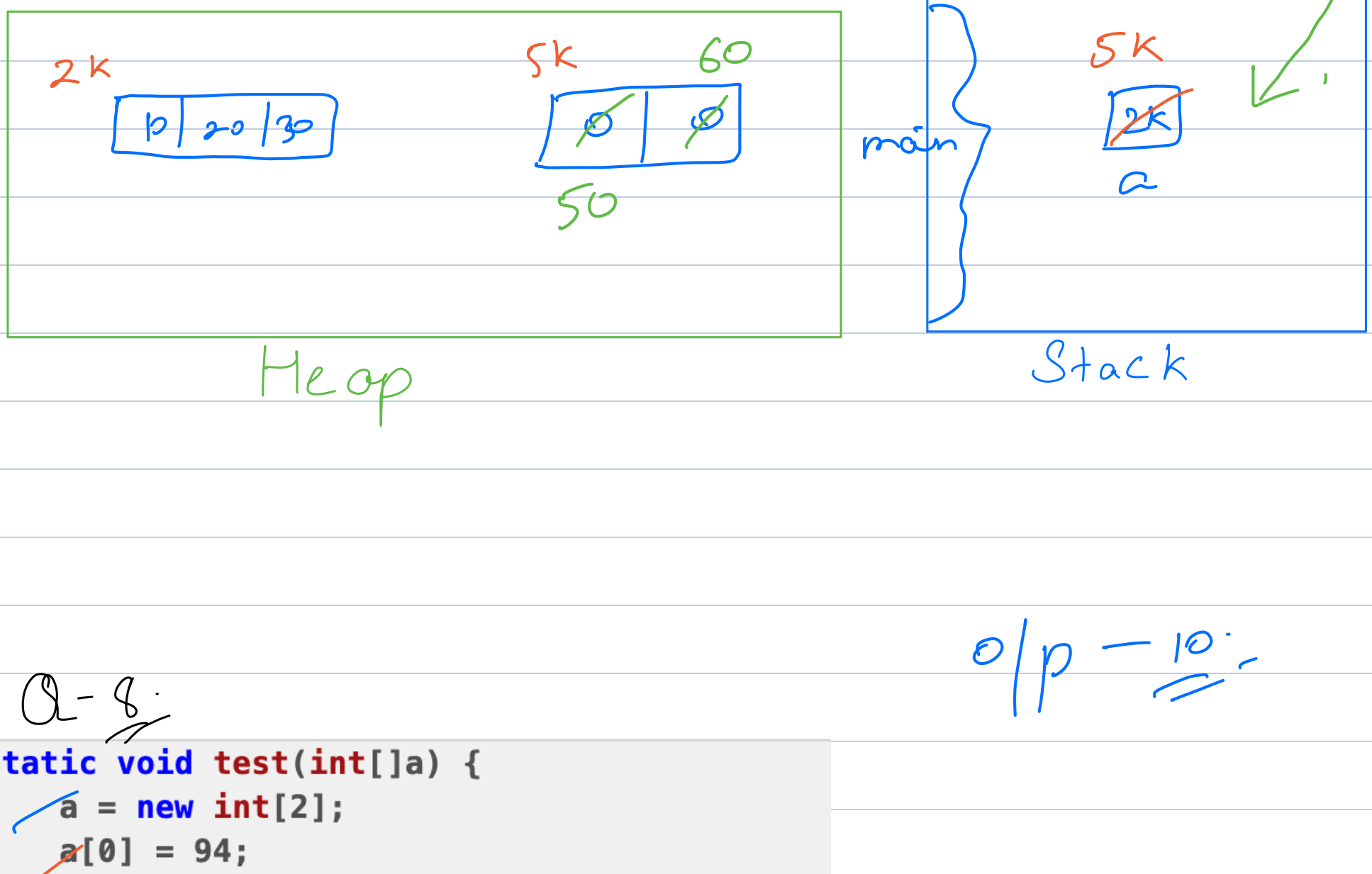
```java
static void swap(int[]a,int[]b) {
    int temp = a[0];
    a[0] = b[0];
    b[0] = temp;
}

public static void main(String args[]) {
    int[]a = {10};
    int[]b = {20};
    swap(a,b);
    System.out.println(a[0] + " " + b[0]);
}
```



Q-7.

o/p - 50.

```
static int[] fun(int[]a) {
    a = new int[2];
    a[0] = 50; a[1] = 60;
     return a;
}

public static void main(String args[]) {
    int[]a = {10,20,30};
    a = fun(a);
    System.out.println(a[0]);
}
```



o/p - 10.

Q-8.

```
static void test(int[]a) {
    a = new int[2];
    a[0] = 94;
}

public static void main(String args[]) {
    int[]a = {10,20,30};
    test(a);
    System.out.println(a[0]);
}
```

Next Class

The next class will be on "Sorting basics".

- **Data Organization** : Sorting helps in organizing data in a specific order, making it easier to search, retrieve, and analyze. For example, alphabetical sorting is commonly used for lists of names or words.
- **Search Algorithms** : Many search algorithms, such as binary search, rely on sorted data to efficiently locate a specific element. Sorting facilitates faster searching by reducing the search space.
- **Ranking and Statistics** : Sorting is used to determine the rank or order of elements based on certain criteria. This is valuable in applications such as sports rankings or financial analysis.