



Today's agenda

↳ object memory

↳ Constructors (copy constructors)

↳ Access modifier

↳ inheritance



AlgoPrep



// Test 2

Java: Pass by value always

```
main() {
```

```
    int x = 10;
```

```
    int y = 20;
```

```
    fun(x, y);
```

```
    print(x + " " + y); // 10 20
```

```
}
```

```
fun(int x, int y) {
```

```
    x = 40;
```

```
    y = 50;
```

```
main {
```

fun {
 x = 40
 y = 20
}

main {
 x = 10
 y = 20
}

Stack

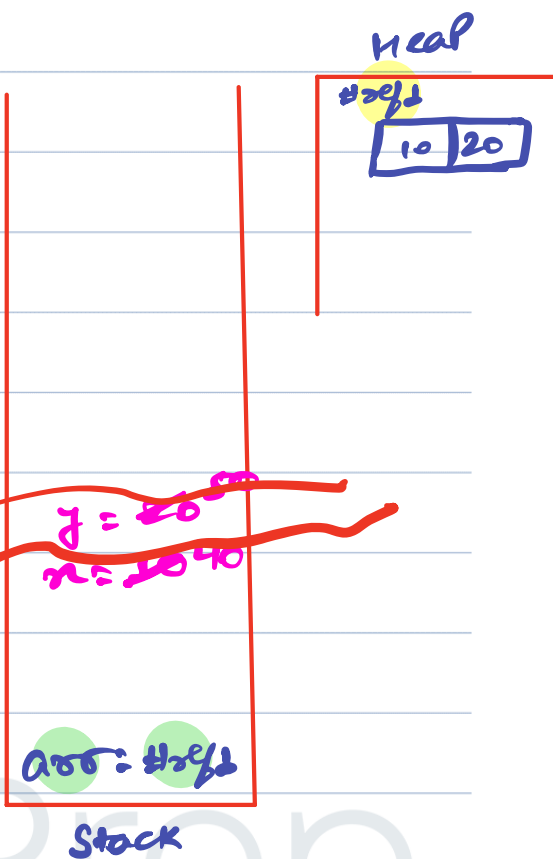
```
}
```



// Test 2

```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arr = {10,20};
8         fun(arr[0], arr[1]);
9
10        System.out.println(arr[0]+" "+arr[1]); → 10 20
11    }
12
13    public static void fun(int x, int y) {
14        x = 40;
15        y = 50;
16    }
17 }
```

fun
main





11+es+3

heap

#29/2

10	20
40	50

```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arr = {10,20};
8
9         fun(arr);
10
11         System.out.println(arr[0]+" "+arr[1]);
12     }
13
14     public static void fun(int[] arr) {
15         arr[0] = 40;
16         arr[1] = 50;
17     }
18 }
19
20
21
22
```

fun { arr: 10 20 }
main { arr: 40 50 }

Stack



AlgoPrep

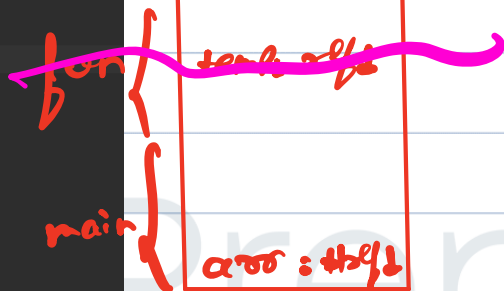


heap



// test 4

```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arr = {10,20};
8         fun(arr);
9
10        System.out.println(arr[0]+" "+arr[1]); 40 50
11    }
12
13    public static void fun(int[] temp) {
14        temp[0] = 40;
15        temp[1] = 50;
16    }
17 }
18
19
20
21
22
```





Heap

// test 4

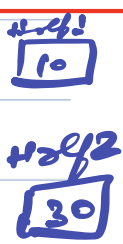
```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arr1 = {10};
8         int[] arr2 = {30};
9
10        fun(arr1, arr2);
11
12        System.out.println(arr1[0] + " " + arr2[0]); → 10 30
13    }
14
15    public static void fun(int[] arr1, int[] arr2) {
16        int[] temp = arr1;
17        arr1 = arr2;
18        arr2 = temp;
19    }
20
21 }
22
23 }
24
```

fun
main

~~temp: #x/1~~
~~arr2: #x/2~~
~~arr1: #x/1~~

arr2: #x/2
arr1: #x/1

Stack





Heap

arr1: 30

arr2: 10

```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arr1 = {10};
8         int[] arr2 = {30};
9
10        fun(arr1, arr2);
11
12        System.out.println(arr1[0] + " " + arr2[0]);
13    }
14
15    public static void fun(int[] temp1, int[] temp2) {
16        int t = temp1[0];
17        temp1[0] = temp2[0];
18        temp2[0] = t;
19    }
20
21 }
22
23 }
24
```

30 10

fun
main

t = 10
temp2 = 10
temp1 = 30
arr2 = 10
arr1 = 30

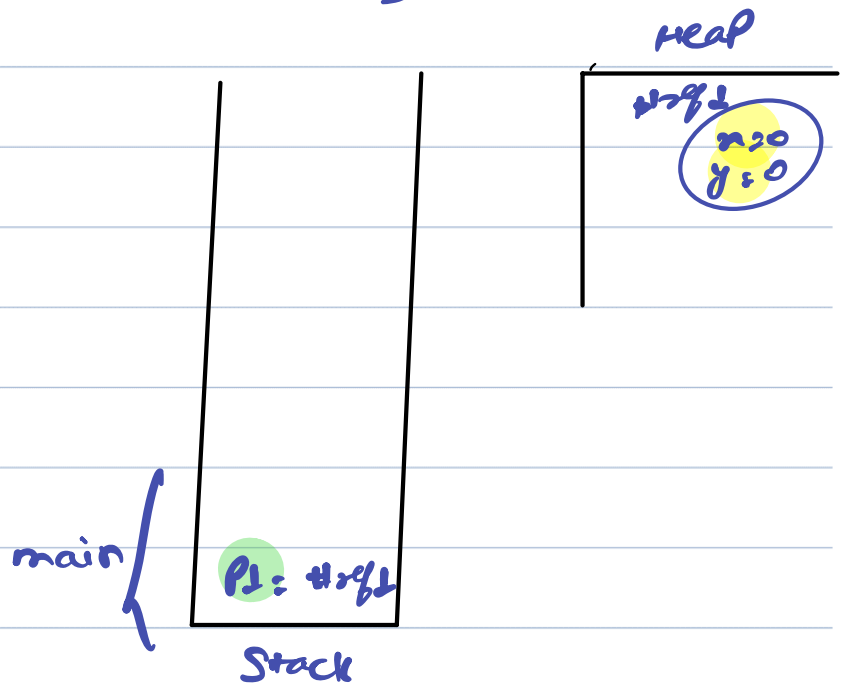


Class Pair {

int x;
int y;

}

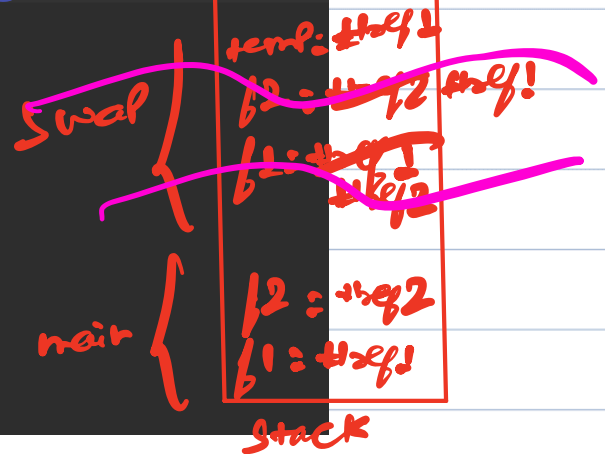
Pair p1 = new Pair();



11+28+6

AlgoPrep

```
1 package Student;
2
3 public class student {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Fun f1 = new Fun(10,20);
8         Fun f2 = new Fun(30,40);
9
10        Swap(f1,f2);
11
12        System.out.println(f1.x + " " + f2.x); 10 30
13    }
14
15    public static void Swap(Fun f1, Fun f2) {
16        Fun temp = f1;
17        f1 = f2;
18        f2 = temp;
19    }
20
21
22
23 }
24
```



Break till 9:15 PM



→ Access modifiers

Private default Public ~~Protected~~

```
class BankAccount {  
    int accountno;  
}
```

Private
default
Public

within the
class

✓
✓
✓

across class
but within
same package

xx
✓
✓

across class
and across
package

xx
xx
✓



* Copy Constructors

```
class Fun {  
    private int x;  
    int y;
```

```
    Fun (fun f) {  
        x = f.x;  
        y = f.y;  
    }  
}
```

```
void value-of-x () {  
    s.o.p (x);  
}
```

Fun f1 = new Fun ();

f1.x = 10;

f1.y = 20;

Fun f2 = new Fun (f1);

↑
Deep copy

1st ref:
x: 10
y: 20

2nd ref:
x: 10
y: 20

main

f2 = 1st ref

f1 = 1st ref

→ shallow copy



```
class Jun {  
    private int x;  
    int y;  
    Jun (Jun j) {  
        x = j.x;  
        y = j.y;  
    }  
}
```

Jun j1 = new Jun();

j1.x = 10;

j1.y = 20;

Jun j2 = j1;

```
void value-of-x () {  
    s.o.p (x);  
}
```

ref:
x = 10
y = 20

main

j2 = this
j1 = this