	0	1	2	3	4	5	6	7
0	0	1	0	0	0	0	0	0
1	0	1	0	1	1	1	1	0
2	0	1	0	1	0	0	0	0
3	0	1	0	1	0	1	1	1
4	0	0	0	0	0	0	0	0
5	0	1	0	1	1	1	1	0
6	0	1	0	1	1	1	1	0
7	0	1	0	0	0	0	0	0

0 → can be visited
1 → wall

nr = 8
nc = 8

path
src → dest
(0,0) (7,7)

Directions
t
l
d
r

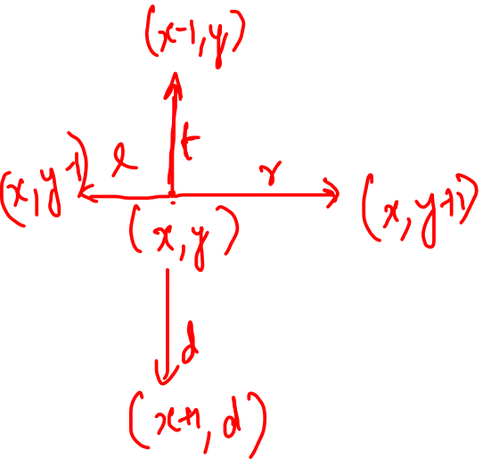
	0	1	2	3	4	5	6	7
0	0	1	0	0	0	0	0	0
1	0	1	0	1	1	1	1	0
2	0	1	0	1	0	0	0	0
3	0	1	0	1	0	1	1	1
4	0	0	0	0	0	0	0	0
5	0	1	0	1	1	1	1	0
6	0	1	0	1	1	1	1	0
7	0	1	0	0	0	0	0	0

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								

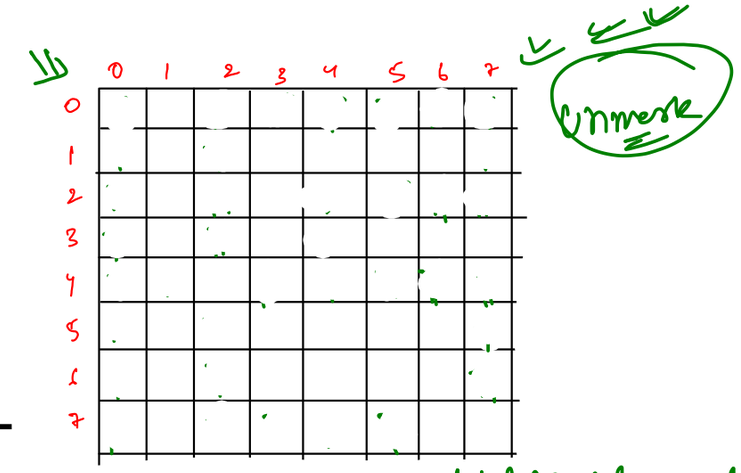
[r r r r r r p p p p]

[p p p r r r p p r r r r r p p p p]

[r r r r r p p p r r p p p p]



	0	1	2	3	4	5	6	7
0	0	1	0	0	0	0	0	0
1	0	1	0	1	1	1	1	0
2	0	1	0	1	0	0	0	0
3	0	1	0	1	0	1	1	1
4	0	0	0	0	0	0	0	0
5	0	1	0	1	1	1	1	0
6	0	1	0	1	1	1	1	0
7	0	1	0	0	0	0	0	0



(dddd r r t t t t r r r r r dd l l l l d d r r r d d d)

(dddd r r dd d r r r r r)

= (dddd r r r r r r r dd)

- out of range
- ~~Cyclic path (not allowed)~~
- → already visited block
- → wall

visited

	0	1	2	3	4	5	6	7
0	T							
1	T							
2								
3								
4								
5								
6								
7								

```java
public static void floodfill(int[][] maze , int r , int c , String psf , boolean visited[][]) {
    if(r < 0 || c < 0 || r >= maze.length || c >= maze[0].length || visited[r][c] == true || maze[r][c] == 1){
        // out of range or already visited or wall : invalid pos
        return;
    }
    if(r == maze.length-1 && c == maze[0].length-1){
        System.out.println(psf);
        return;
    }

    visited[r][c] = true; // mark

    floodfill(maze,r-1,c,psf+"t",visited); // top
    floodfill(maze,r,c-1,psf+"l",visited); // left
    floodfill(maze,r+1,c,psf+"d",visited); // bottom
    floodfill(maze,r,c+1,psf+"r",visited); // right

    visited[r][c] = false;// unmark
}
```

	0	1	2	3	4	5	6	7
0	0 + d	1	0	0	0	0	0	0
1	0	1	0	1	1	1	1	0
2	0	1	0	1	0	0	0	0
3	0	1	0	1	0	1	1	1
4	0	0	0	0	0	0	0	0
5	0	1	0	1	1	1	1	0
6	0	1	0	1	1	1	1	0
7	0	1	0	0	0	0	0	0

5 n

10
20
30
40
50

60 → Target

arr =

10	20	30	40	50
0	1	2	3	4

Target = 60

10 + 20 + 30

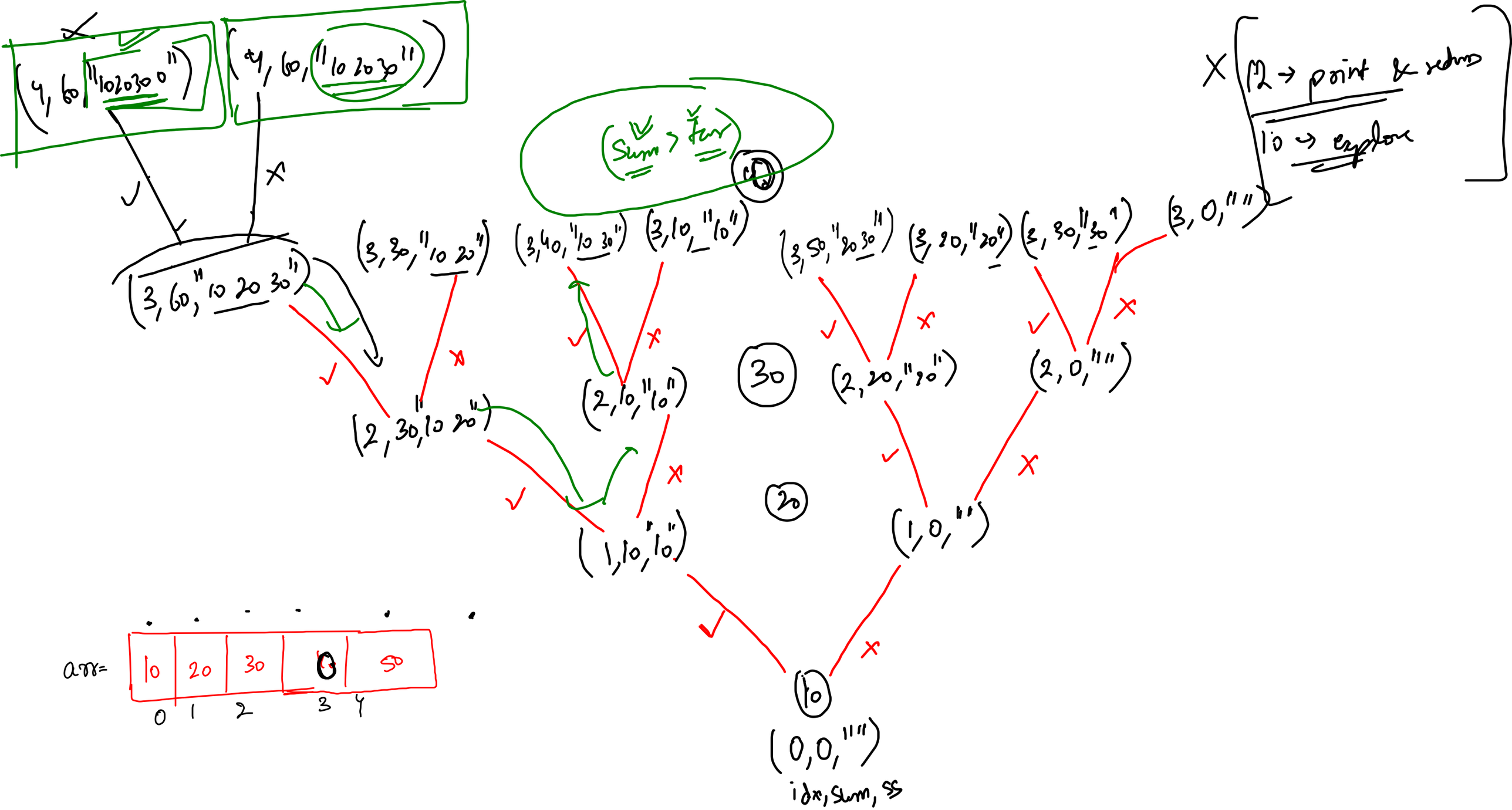
20 + 40

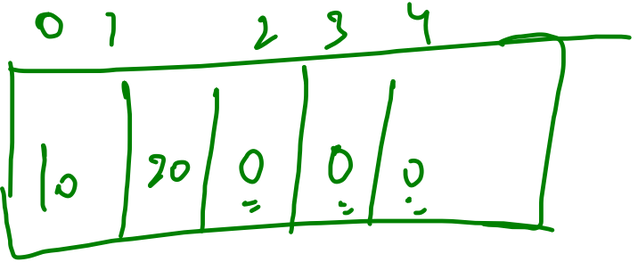
10 + 50

```
10, 20, 30, .
10, 50, .
20, 40, .
```

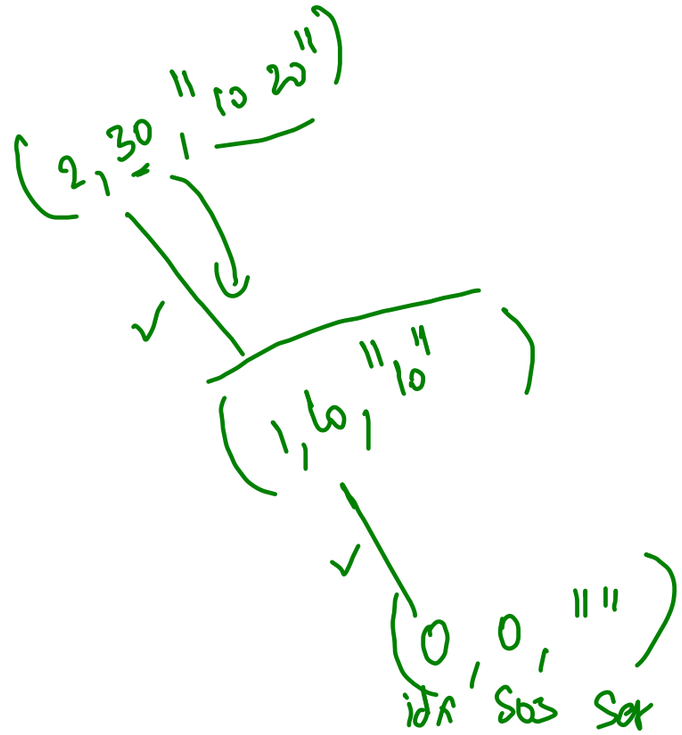
Print all Subsets whose total contribution is equal to target.

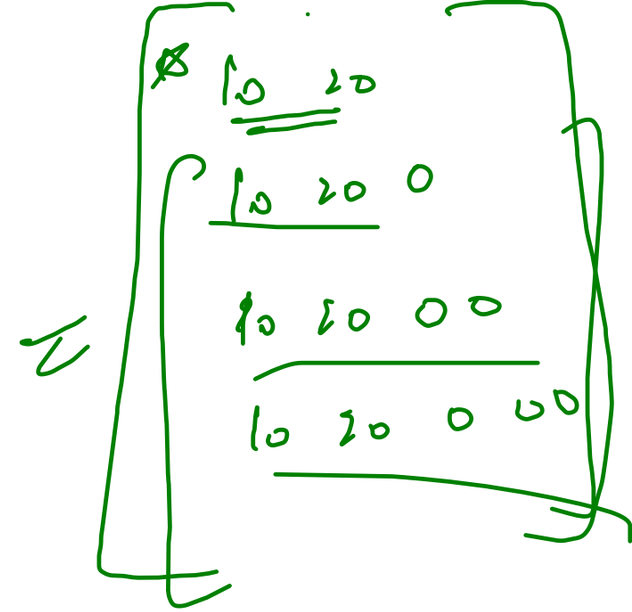
(string → subsets) == (array → subsets.)


(4, 60, "102030 0")
(4, 60, "10 20 30")
(Sum > Tar)
40
X [12 → point & return
10 → explore]
(3, 60, "10 20 30")
(3, 30, "10 20")
(3, 40, "10 30")
(3, 10, "10")
(3, 50, "20 30")
(3, 20, "20")
(3, 30, "30")
(3, 0, "")
(2, 30, "10 20")
(2, 10, "10")
30
(2, 20, "20")
(2, 0, "")
(1, 10, "10")
20
(1, 0, "")
10
(0, 0, "")
idx, sum, ss
arr=
10
20
30
0
50
0
1
2
3
4



tar = 30






```java
public static void printTargetSumSubsets(int[] arr, int idx, String set, int sos, int tar) {
    if(idx == arr.length){
        // every element has been given a choice
        if(sos == tar){
            System.out.println(set+".");
        }
        return;
    }
    if(sos > tar){
        // invalid
        return;
    }

    printTargetSumSubsets(arr,idx+1,set+arr[idx]+", ",sos+arr[idx],tar); //  inc
    printTargetSumSubsets(arr,idx+1,set,sos,tar); //  inc
}
```

Classes & Objects

Class => structure to define Data Members for an entity
+
~~Member~~ functions

data members :- Attributes / properties

entity Explain values

⇓

class

member functions

```
public static class Person{
    data members:
        String name;
        int age;
    member functions:
        void sayHi(){
            System.out.println(name + "-->" + age);
        }
}
```

Blue print

class → object

```
Person p1 = new Person();
```

[abc [10]
def [15]]

Instances → Heap
References = Prog. Stack

1. Class
2. Objects
3. Context
4. ~~Objects~~ Ref./Instances

Objects by Jvm

```java
public class Main{

    public static class Person{
        int age;
        String name;

        void saysHi(){
            System.out.println(name+"["+age+"]");
        }
    }
    public static void main(String []args){
        Person p1 = new Person();
        p1.name = "abc";
        p1.age = 10;
        p1.saysHi();

        Person p2 = new Person();
        p2.name = "def";
        p2.age = 15;
        p2.saysHi();
    }
```

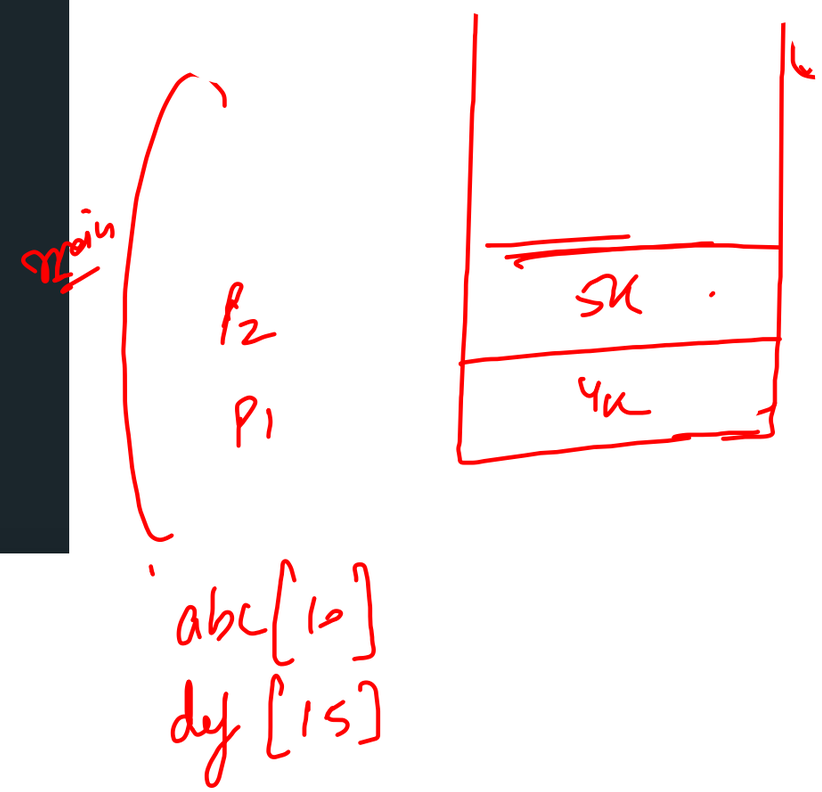
main

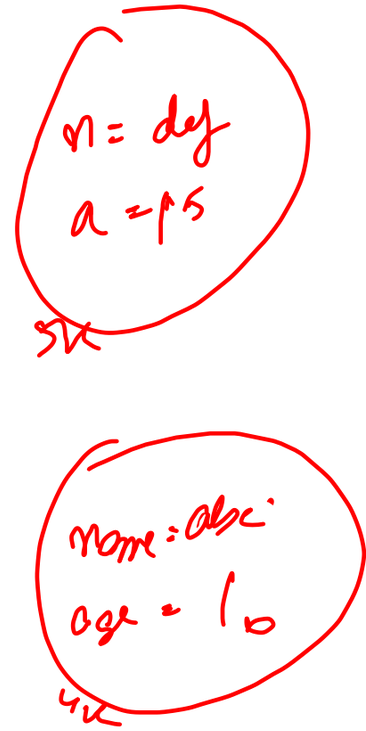
p2	5k
p1	4k

age = 10
name = abc
4k

age = 15
name = def
5k

```java
public static void swap(Person p1 , Person p2){
    Person temp = p1;
    p1 = p2;
    p2 = temp;
}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    swap(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```





```java
public static void swap1(Person p1 , Person p2){
    int age = p1.age;
    p1.age = p2.age;
    p2.age = age;

    String name = p1.name;
    p1.name = p2.name;
    p2.name = name;
}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    swap1(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```

abc[10]
def[15]

def[15]
abc[10]

swap1(4k, 5k)

Swapping done

name	"abc"
age	10
P_2	5k
P_1	4k

main

P_2	5k
P_1	4k

name = ~~abc~~ def
age = ~~10~~ 15

4k

n = ~~def~~ abc
a = ~~15~~ 10

5k

Question 1]

Pen paper

08:40

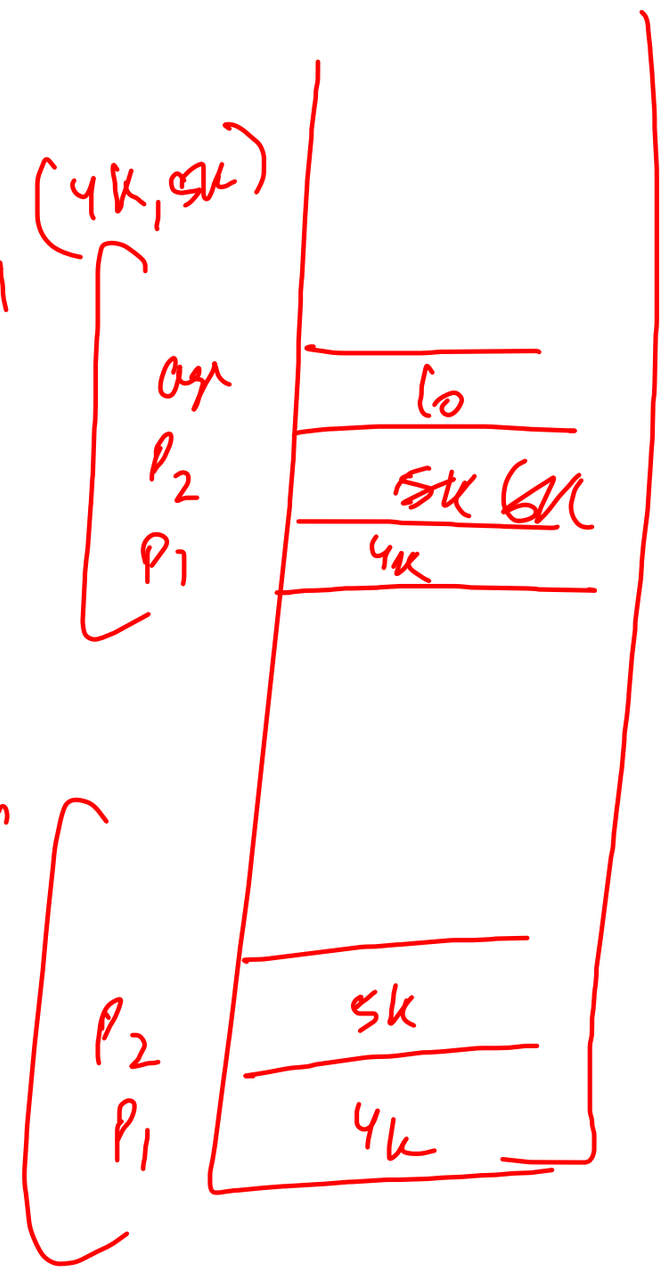
```java
public static void game1(Person p1 , Person p2){
    p2 = new Person();
    int age = p1.age;
    p1.age = p2.age;
    p2.age = age;

    p1 = new Person();
    String name = p1.name;
    p1.name = p2.name;
    p2.name = name;
}

public static void game2(Person p1 , Person p2){}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    // swap1(p1,p2);
    game1(p1,p2);
    // game2(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```

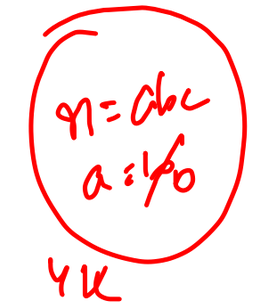
Question 2]

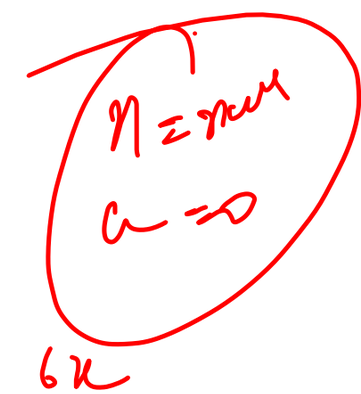
```java
public static void game2(Person p1 , Person p2){
    p2 = new Person();
    p1 = new Person();
    int age = p1.age;
    p1.age = p2.age;
    p2.age = age;

    String name = p1.name;
    p1.name = p2.name;
    p2.name = name;
}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    // swap1(p1,p2);
    // game1(p1,p2);
    game2(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```

```java
public static void game1(Person p1 , Person p2){
    p2 = new Person();
    int age = p1.age;
    p1.age = p2.age;
    p2.age = age;

    p1 = new Person();
    String name = p1.name;
    p1.name = p2.name;
    p2.name = name;
}

public static void game2(Person p1 , Person p2){}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    // swap1(p1,p2);
    game1(p1,p2);
    // game2(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```

abc[10]
def[15]







```java
public static void game2(Person p1 , Person p2){
    p2 = new Person();
    p1 = new Person();
    int age = p1.age;
    p1.age = p2.age;
    p2.age = age;

    String name = p1.name;
    p1.name = p2.name;
    p2.name = name;
}
public static void main(String []args){
    Person p1 = new Person();
    p1.name = "abc";
    p1.age = 10;
    // p1.saysHi();

    Person p2 = new Person();
    p2.name = "def";
    p2.age = 15;
    // p2.saysHi();

    p1.saysHi();
    p2.saysHi();
    // swap1(p1,p2);
    // game1(p1,p2);
    game2(p1,p2);
    p1.saysHi();
    p2.saysHi();
}
```

abc[10]
def[15]

abc[10]
def[15]

game2 (4K, 5K)

P2
P1

5K 6K
4K 7K

3.5

main

P2
P1

5K
4K

n: null
a: 0
7K

n = null
a = 0
6K

n = def
a = 15
5K

n = abc
a = 10
4K

Constructors

- functions with special name i.e. same as class name
- Default Constructor

if no constructor is provided by the user then java language does the work for you i.e. java will provide an empty default constructor.

objectCreation

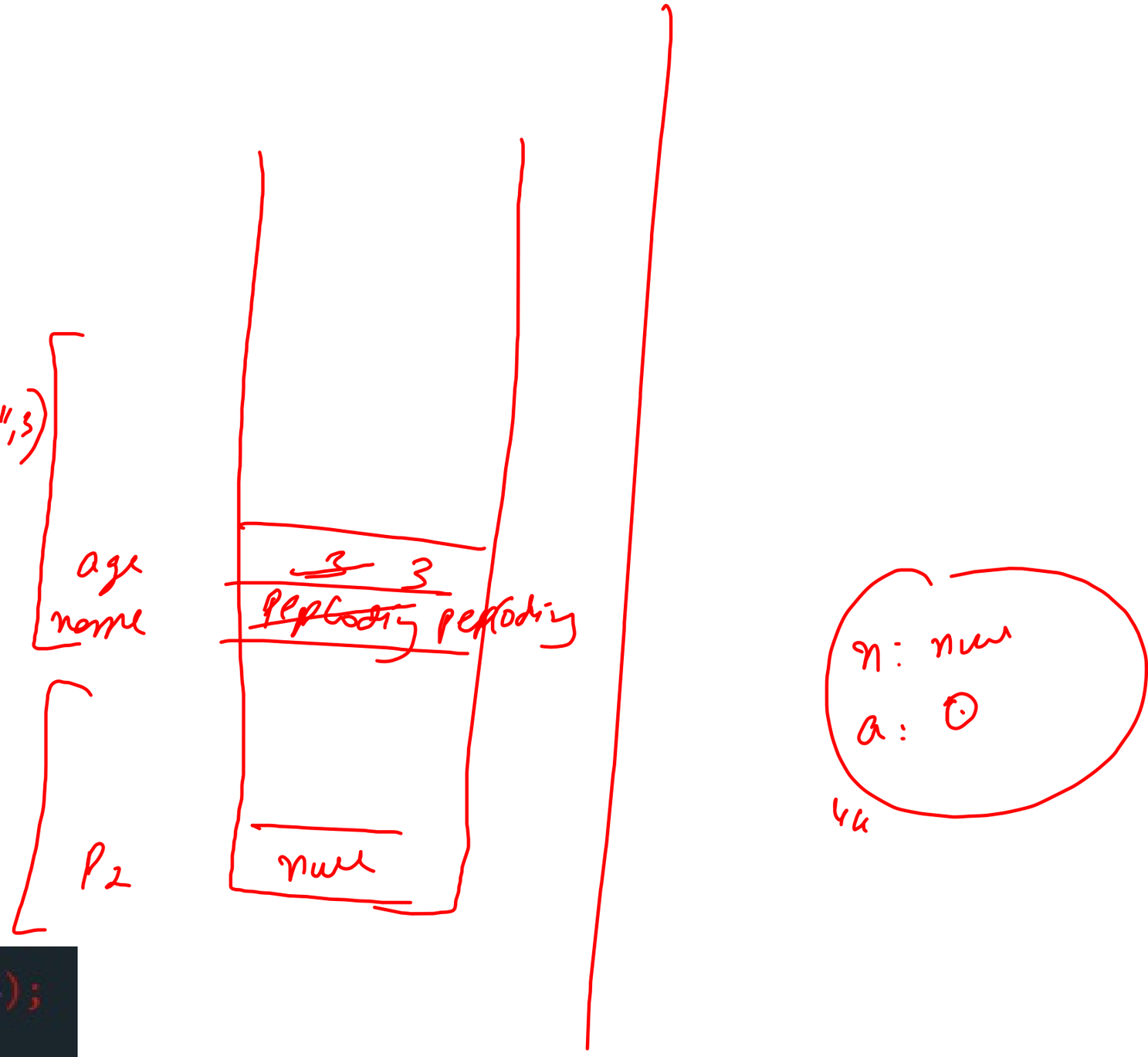
- Memory allocation
- Parsing (Default values)
- Constructor Invoke

```java
public static class Person{
    int age; = 15
    String name; = "0"

    /*
    Person(){
        // this is provided by java lang. incase user forgets.
    }
    */

    Person(){
        System.out.println("Welcome users");
        name = "NA";
        age = -1;
    }

    Person(String n , int a){
        name = n;
        age = a;
    }

    void saysHi(){
        System.out.println(name+"["+age+"]");
    }
}
```

```java
public static class Person{
    int age;
    String name;

    /*
    Person(){
        // this is provided by java lang. incase user
    }
    */

    Person(){

    }

    Person(String name , int age){
        name = name;
        age = age;
    }

    void saysHi(){
        System.out.println(name+"["+age+"]");
    }
}
```

person("pepcoding",3)

main



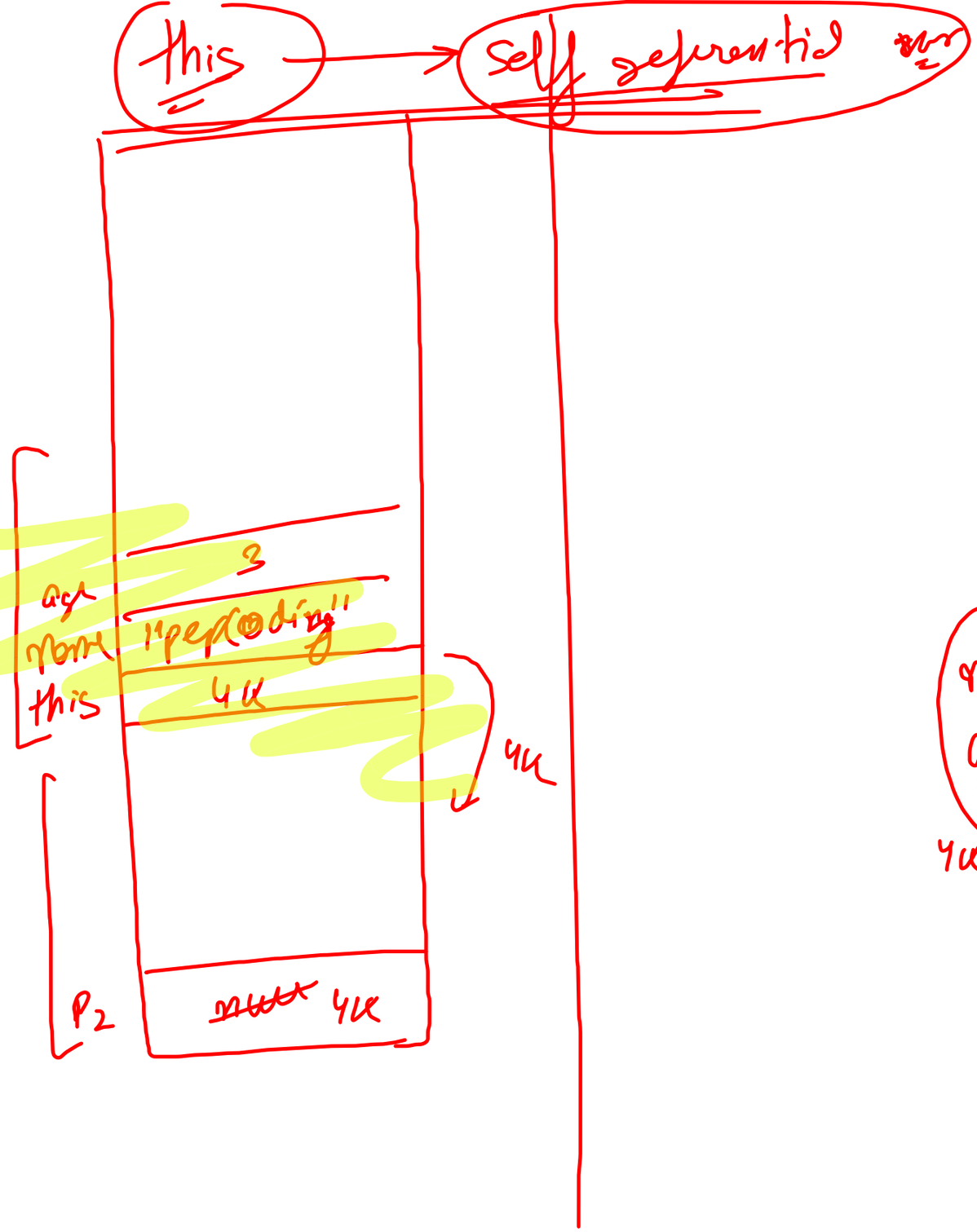

```java
Person p2 = new Person("pepcoding",3);
p2.saysHi();
```

```java
public static class Person{
    int age;
    String name;

    /*
    Person(){
        // this is provided by java lang. incase
    }
    */

    Person(){

    }

    Person(String name , int age){
        this.name = name;
        this.age = age;
    }

    void saysHi(){
        System.out.println(name+"["+age+"]");
    }
}
```

Person("pepcoding", 3)

main

```java
Person p2 = new Person("pepcoding",3);
p2.saysHi();
```



name = pepcoding
age = 3
4K