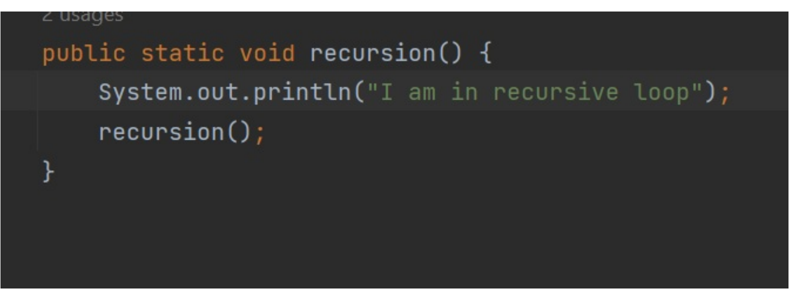
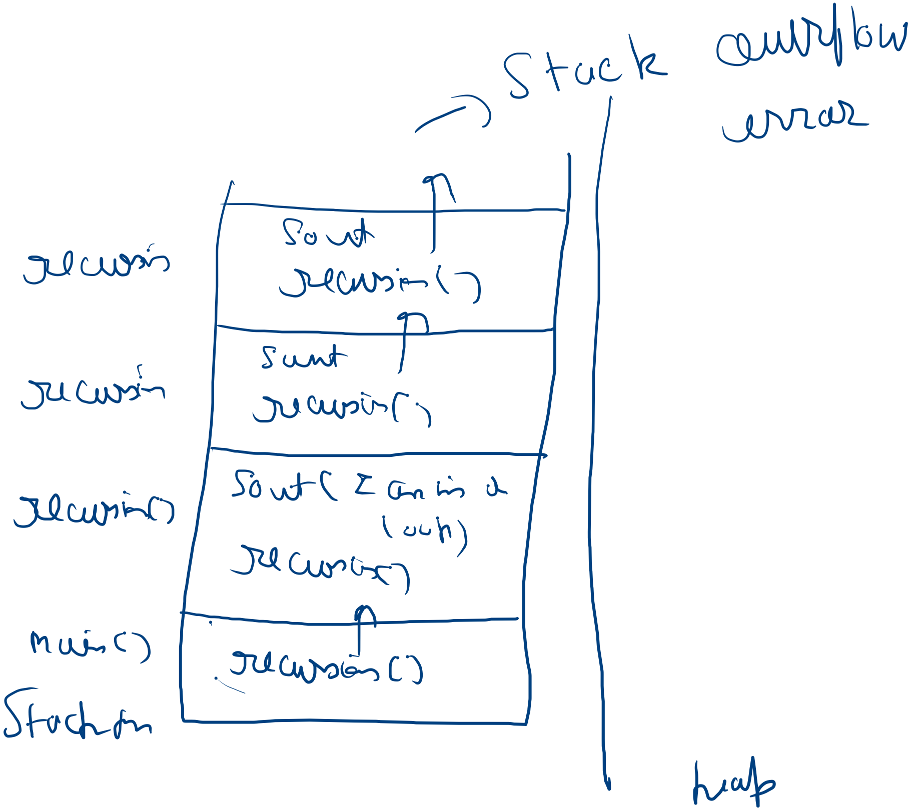
Recursion

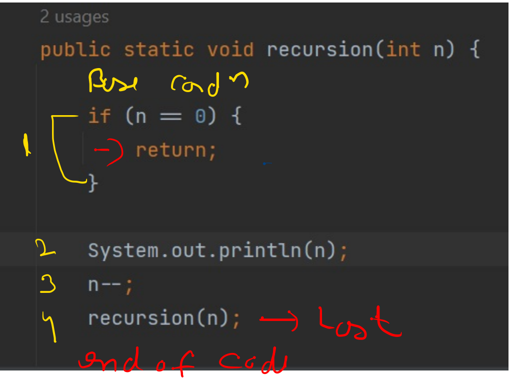
→ A fn calling itself



```
2 usages
public static void recursion() {
    System.out.println("I am in recursive loop");
    recursion();
}
```

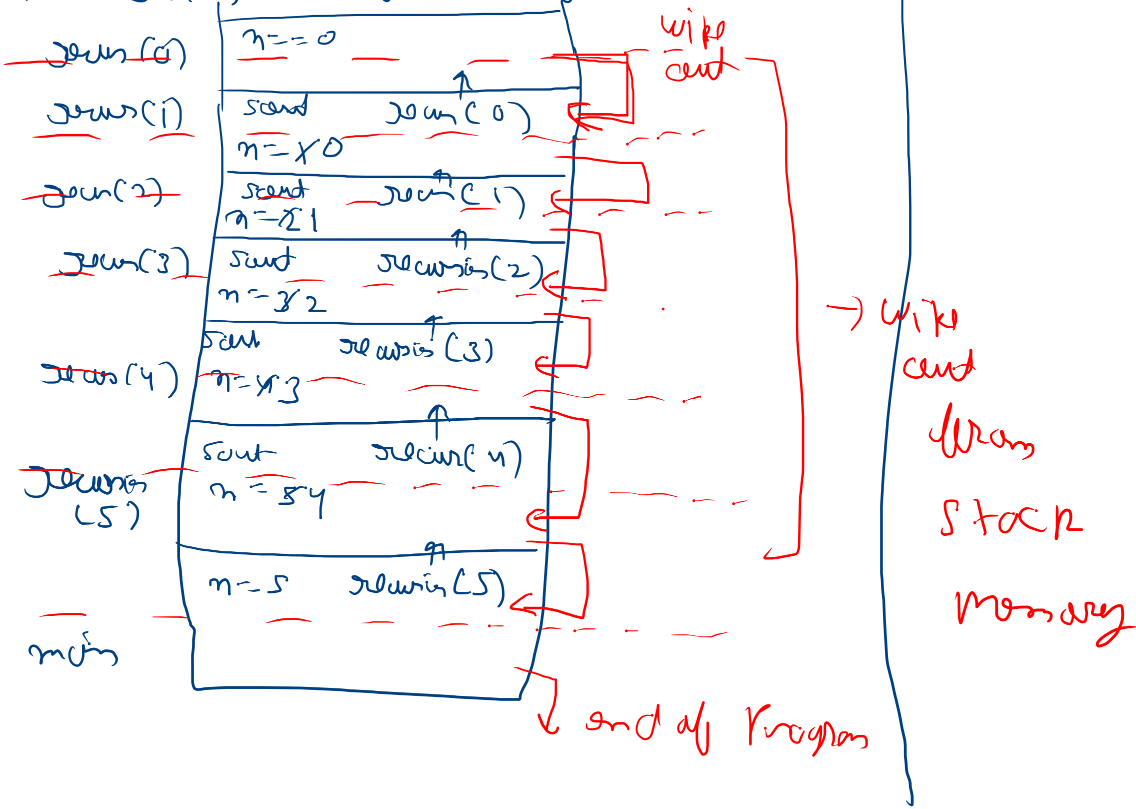


1) Base condition



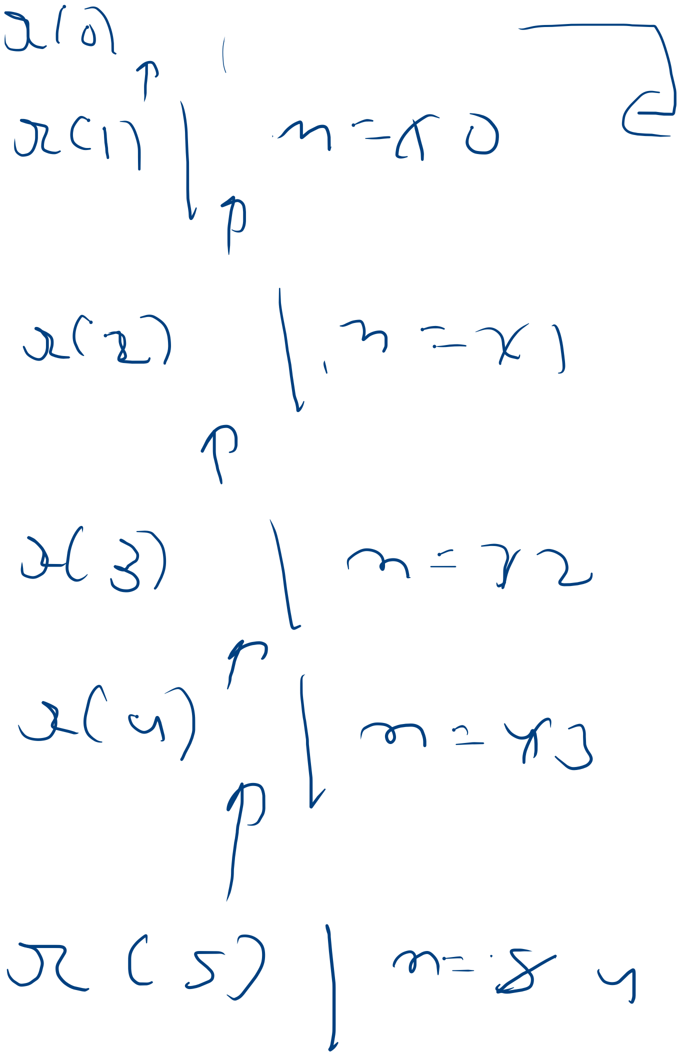
```
2 usages
public static void recursion(int n) {
    Base cond^n
1   if (n == 0) {
     → return;
    }

2   System.out.println(n);
3   n--;
4   recursion(n);  → Last
end of code
```

→ Recursion end condition



5
4
3
2
1

Recursive tree



~~r(0) → 1, 2~~

~~r(1) → 1, 2, 3, 4, 5,~~ 6 | n = ~~1~~ 0

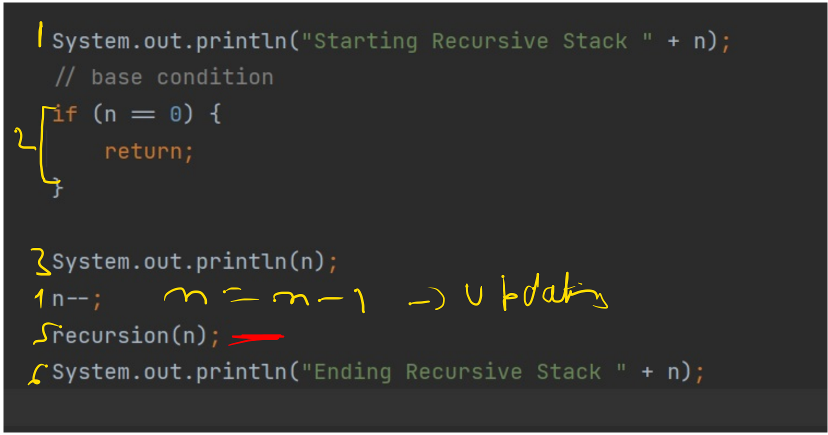
r(2) → 1, 2, 3, 4, 5, 6 | n = ~~2~~ 1

r(3) → 1, 2, 3, 4, 5, 6 | n = ~~3~~ 2

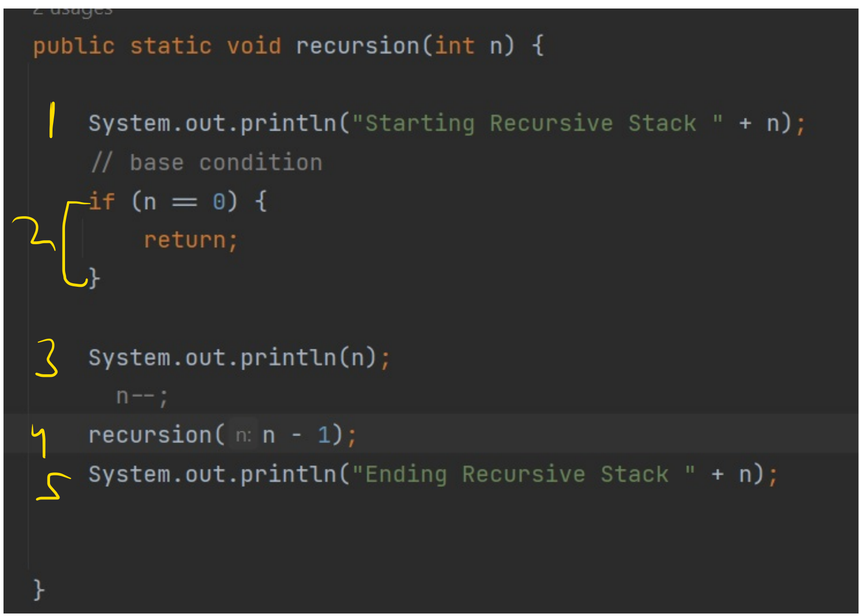
r(4) → 1, 2, 3, 4, 5, 6 | n = ~~4~~ 3

r(5) → 1, 2, 3, 4, 5, 6 | n = ~~5~~ 4

main



```
1 System.out.println("Starting Recursive Stack " + n);
  // base condition
2 if (n == 0) {
      return;
  }

3 System.out.println(n);
4 n--;          n = n-1 → updating
5 recursion(n);
6 System.out.println("Ending Recursive Stack " + n);
```



```
2 usages
public static void recursion(int n) {

    System.out.println("Starting Recursive Stack " + n);
    // base condition
    if (n == 0) {
        return;
    }

    System.out.println(n);
      n--;
    recursion( n: n - 1);
    System.out.println("Ending Recursive Stack " + n);


}
```

r(1-1)   n = 0

↑

r(2-1)   n = 1

↑

r(3-1)   n = 2

↑

r(4-1)   n = 3

↑

r(5-1)   n = 4

↑

r(5) → 1, 2, 3, 4 | n = 5