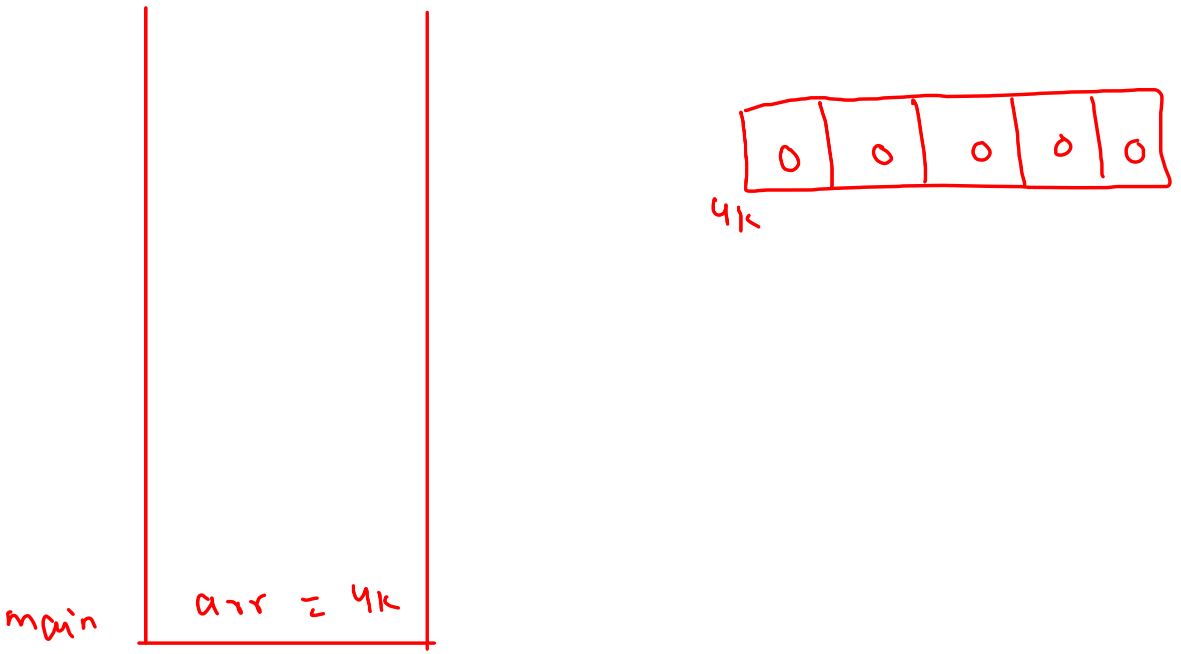
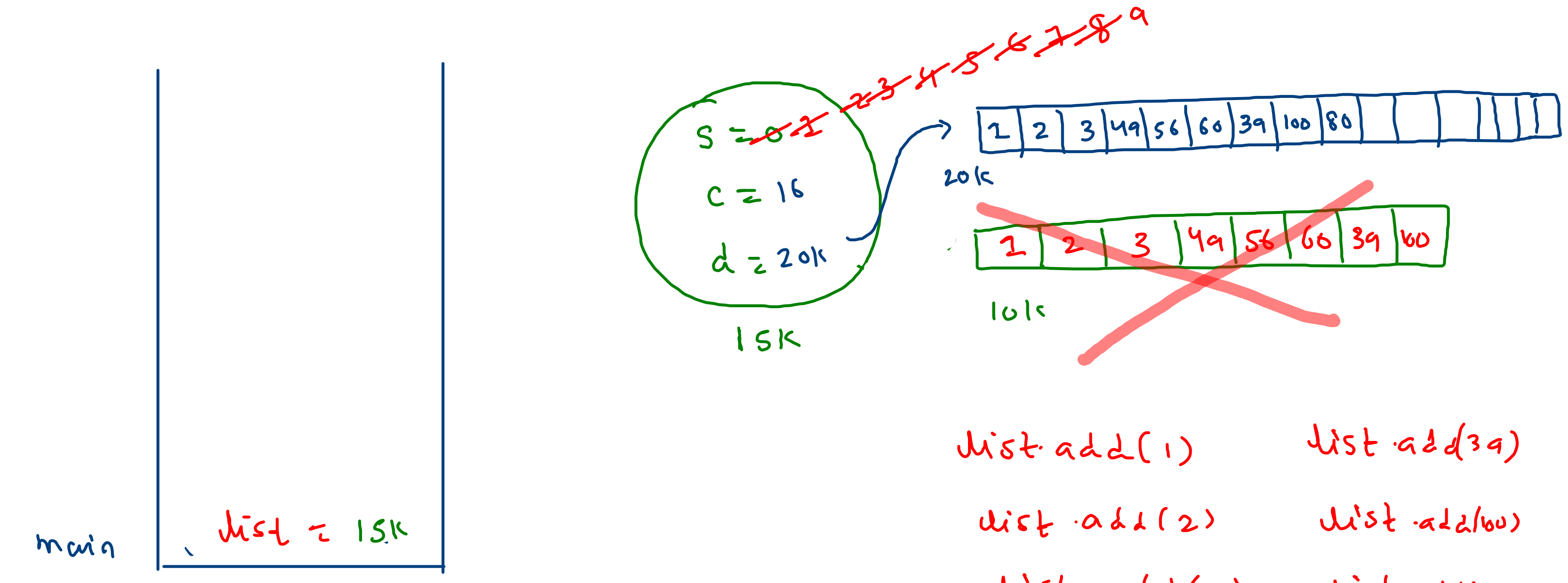
int [ ] arr = new int [5];



Arraylist < Integer > list = new Arraylist<>();



list.add(1)

list.add(2)

list.add(3)

list.add(49)

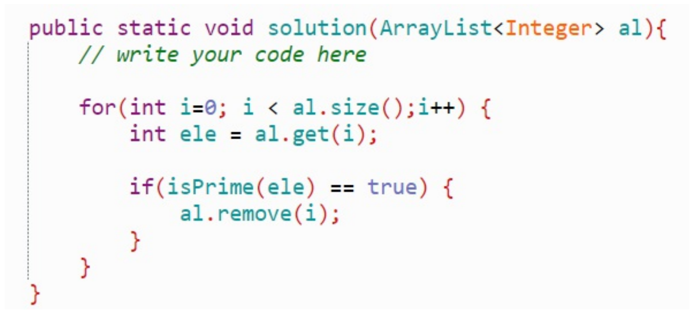
list.add(56)

list.add(60)

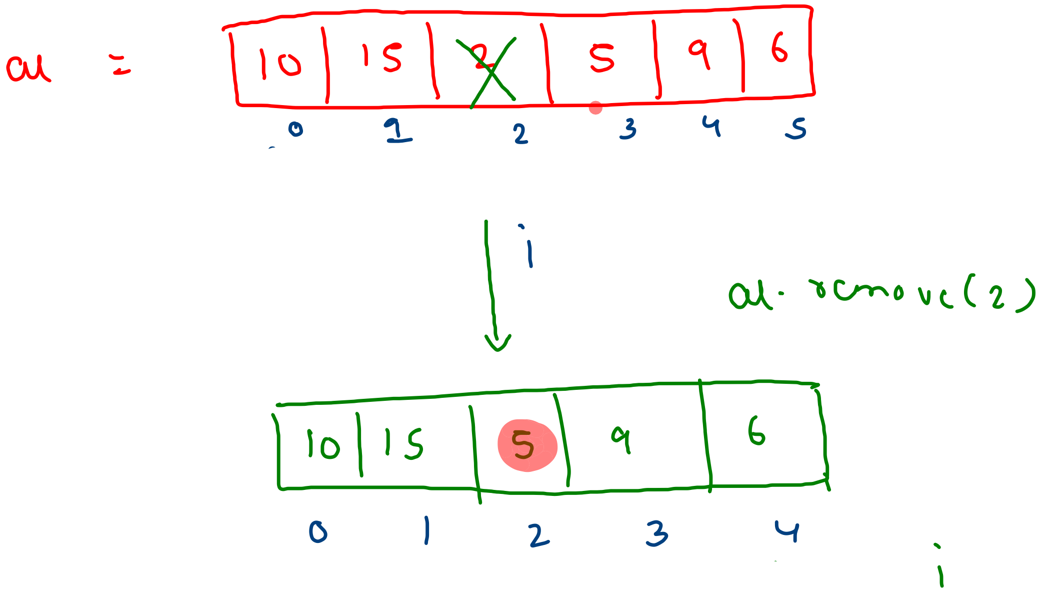
list.add(39)

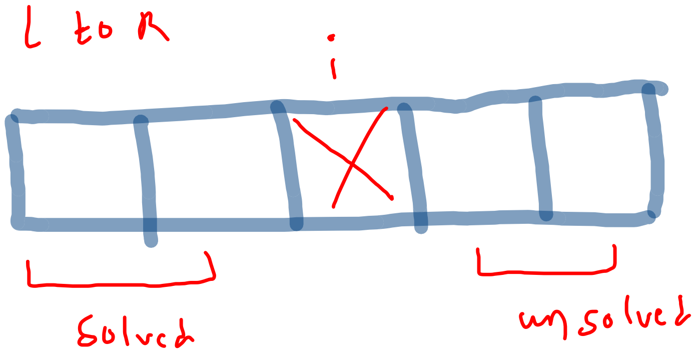
list.add(100)

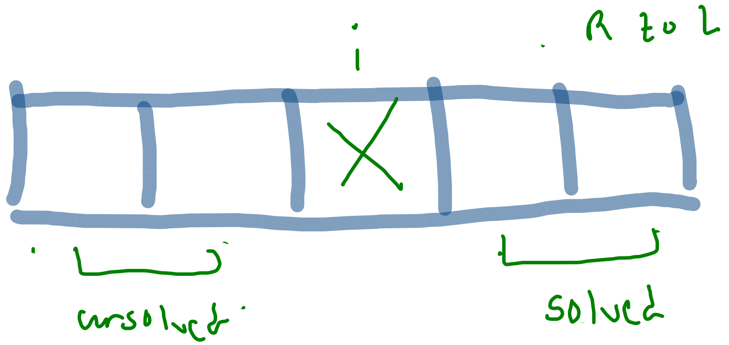
list.add(80)

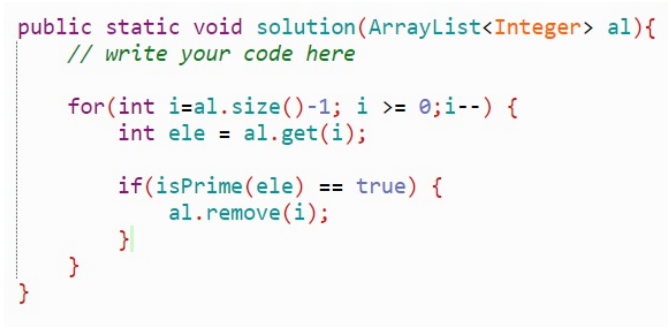


```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    for(int i=0; i < al.size();i++) {
        int ele = al.get(i);

        if(isPrime(ele) == true) {
            al.remove(i);
        }
    }
}
```

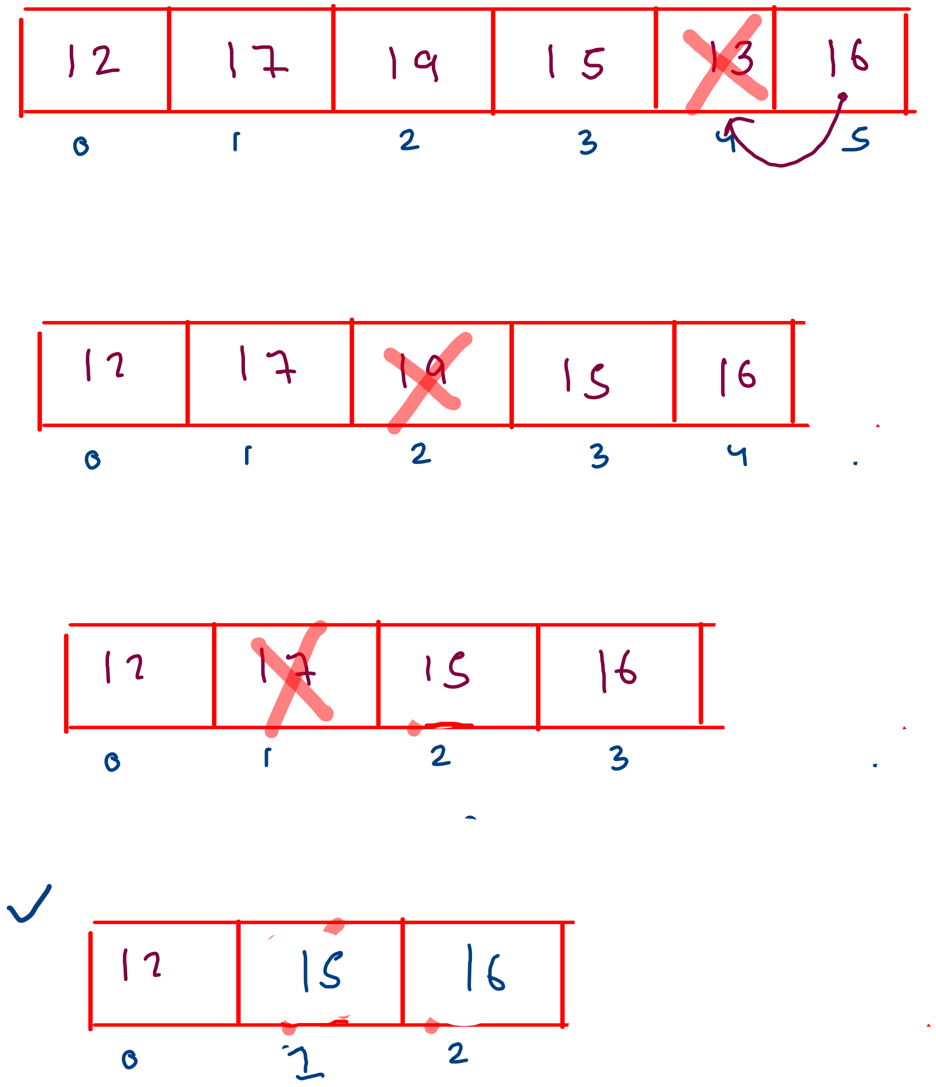








```java
public static void solution(ArrayList<Integer> al){
    // write your code here

    for(int i=al.size()-1; i >= 0;i--) {
        int ele = al.get(i);

        if(isPrime(ele) == true) {
            al.remove(i);
        }
    }
}
```



String

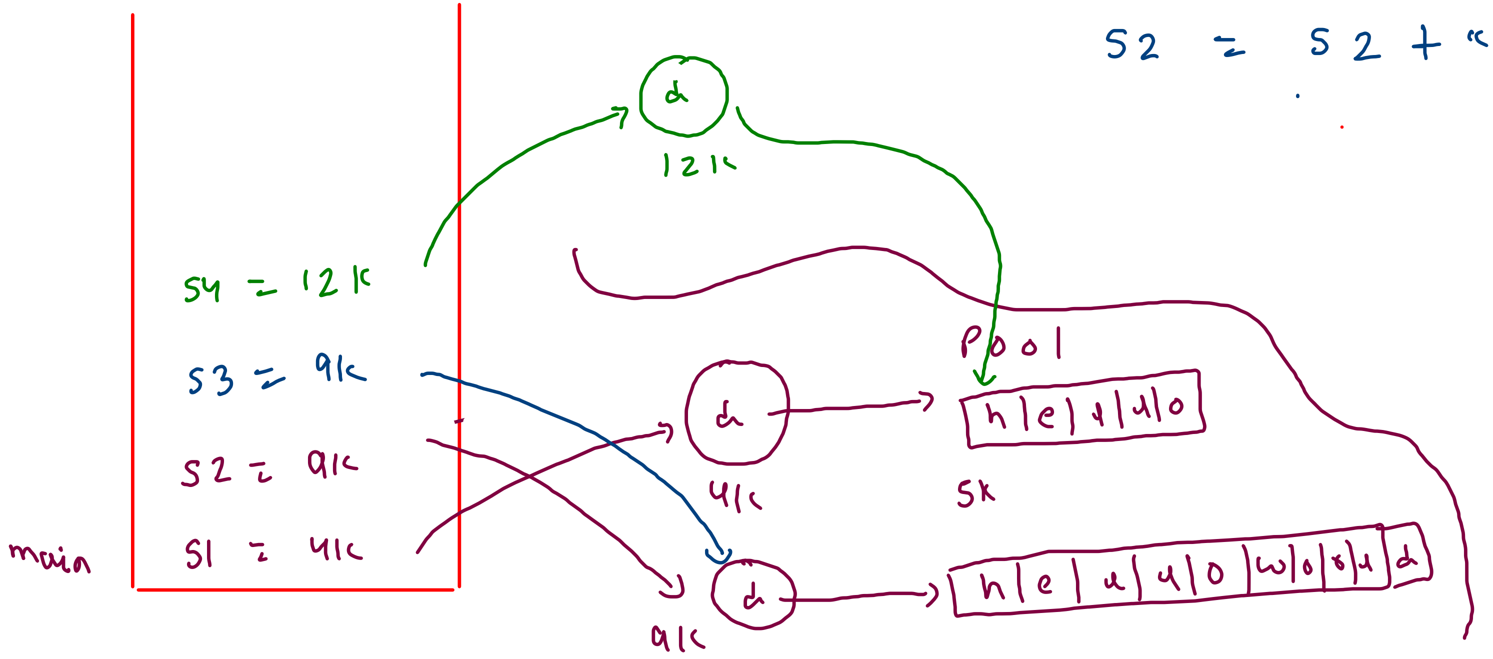
String s1 = "hello";

String s2 = s1;

String s3 = "hello world";

String s4 = new String("hello");

s2 = s2 + "world";

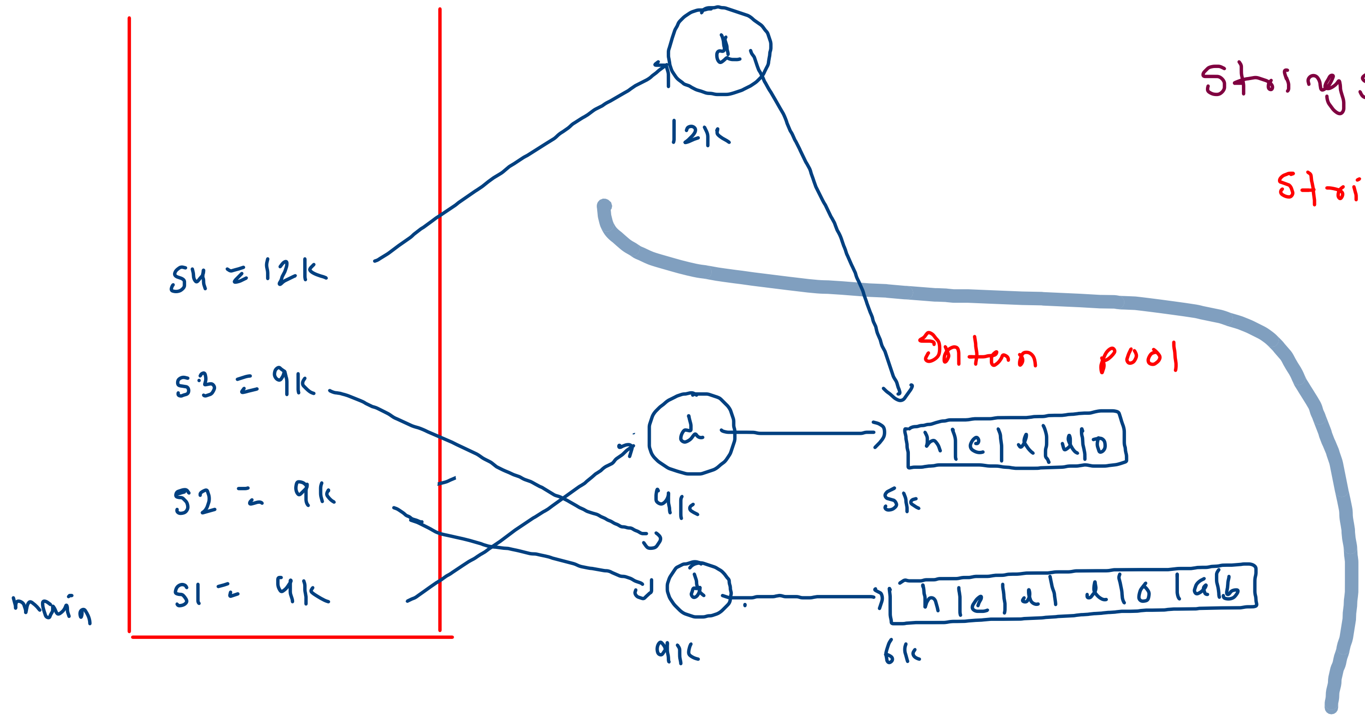


string s1 = "hello";

string s2 = s1; s2 = s2 + "ab";

String s3 = "hello ab";

string s4 = new string ("hello");



Strings -> immutable

String concatenation

O(n)

Arrays	Strings
subset	subsequence
1, 2, 3	str: abc
_ _ _	_ _ _
_ _ 3	_ _ c
_ 2 _	_ b _
_ 2 3	_ b c
1 _ _	a _ _
1 _ 3	a _ c
1 2 _	a b _
1 2 3	abc
subarray (continous)	substring (continous)

print all palindromic substring

str : a b c c b d   n = 6

a
ab
abc
abcc
abccb
abccbd

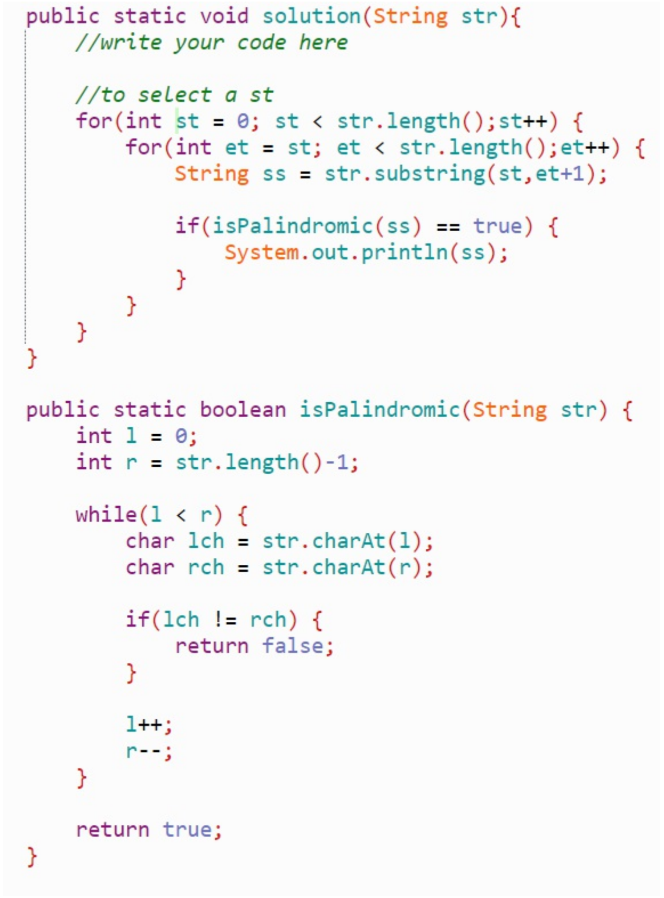
b
bc
bcc
bccb
bccbd

c
cc
ccb
ccbd

c
cb
cbd

b
bd

d



```java
public static void solution(String str){
    //write your code here

    //to select a st
    for(int st = 0; st < str.length();st++) {
        for(int et = st; et < str.length();et++) {
            String ss = str.substring(st,et+1);

            if(isPalindromic(ss) == true) {
                System.out.println(ss);
            }
        }
    }
}

public static boolean isPalindromic(String str) {
    int l = 0;
    int r = str.length()-1;

    while(l < r) {
        char lch = str.charAt(l);
        char rch = str.charAt(r);

        if(lch != rch) {
            return false;
        }

        l++;
        r--;
    }

    return true;
}
```

✓ a
✓ aba
✓ b
✓ a
c

str : a b a c
0 1 2 3

st = 0 → et = 0 1 2 3

(a) (ab) (aba) (abac)

st = 1 → et = 1 2 3

(b) (ba) (bac)

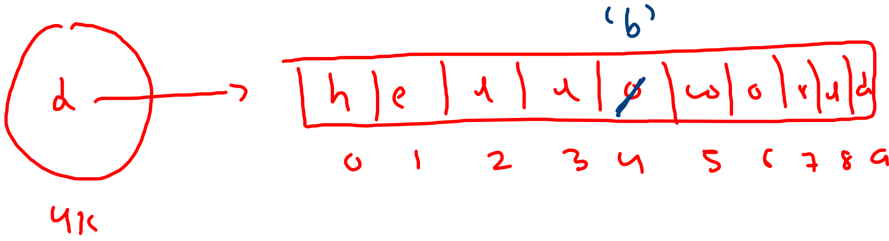
Stringbuilder

:: ArrayList <character>

mutable

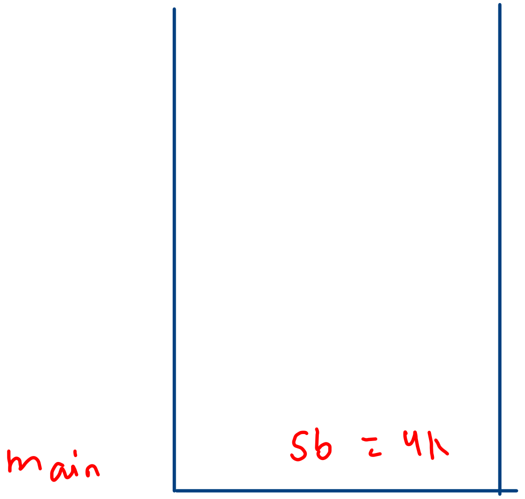
concatenation

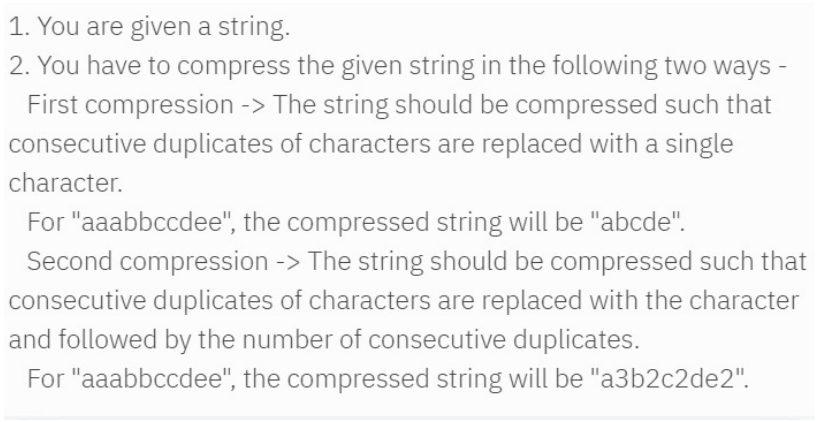
StringBuilder sb = new StringBuilder("hello");



sb.append("world");

sb.setCharAt(4,'b');





1. You are given a string.
2. You have to compress the given string in the following two ways -
   First compression -> The string should be compressed such that consecutive duplicates of characters are replaced with a single character.
   For "aaabbccdee", the compressed string will be "abcde".
   Second compression -> The string should be compressed such that consecutive duplicates of characters are replaced with the character and followed by the number of consecutive duplicates.
   For "aaabbccdee", the compressed string will be "a3b2c2de2".

Str : a a a b b c c d e e

c1 : a b c d e

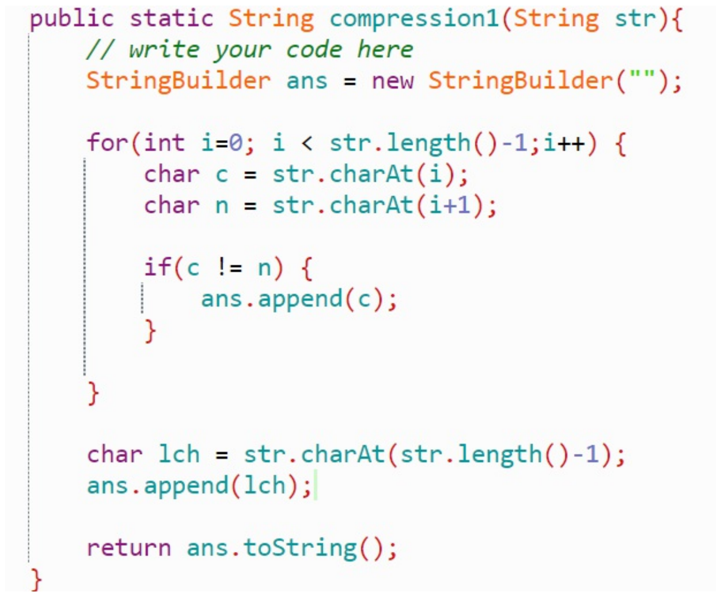
c2 : a3 b2 c2 d e2

Compression 1

str :	a	a	b	b	a	c	c	c	d	e	e	e
	0	1	2	3	4	5	6	7	8	9	10	11
												i

c1 : abacde

```
if (ch(i) != ch(i+1)) {
    ans.append(i);
}
else {

}
```



```java
public static String compression1(String str){
    // write your code here
    StringBuilder ans = new StringBuilder("");

    for(int i=0; i < str.length()-1;i++) {
        char c = str.charAt(i);
        char n = str.charAt(i+1);

        if(c != n) {
            ans.append(c);
        }

    }

    char lch = str.charAt(str.length()-1);
    ans.append(lch);

    return ans.toString();
}
```

a a b b b c c d d d

c n

ans = abcd