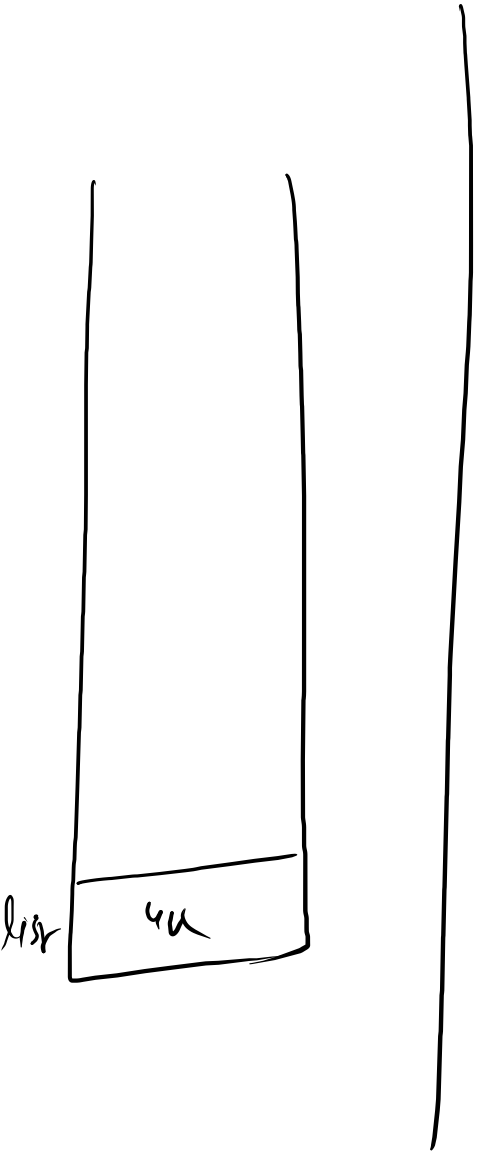
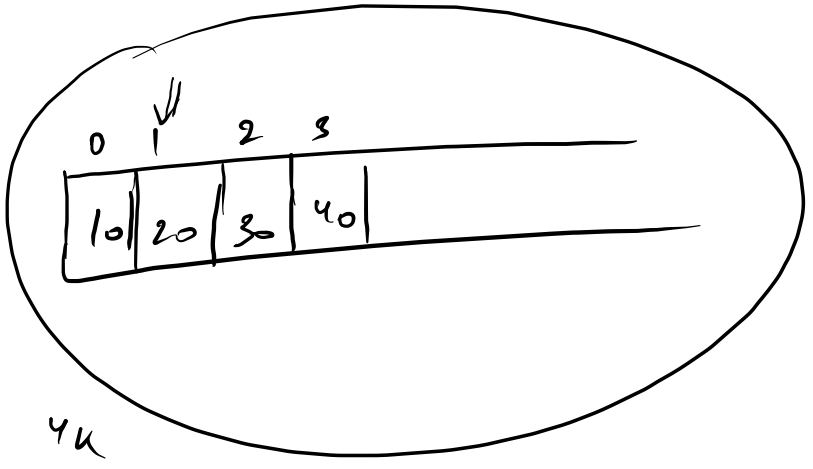
→ ArrayList ?

→ StringBuilder

→ StringBuilder vs String


ArrayList
Array + Dynamic
Usage
Declaration: ArrayList< Type> list = new ArrayList <> ();
Generics
int → Integer
float → Float
double → Double
boolean → Boolean
ArrayList<Integer> list = new ArrayList <> ();





[ ]

[10 20 30 40]

set(_, _)

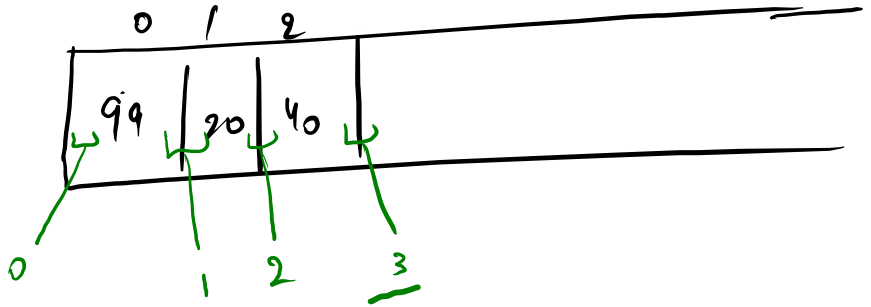
```
public static void main(String[] args) {
    ArrayList<Integer> list = new ArrayList<>();

    System.out.println(list);
    list.add(10);
    list.add(20);
    list.add(30);
    list.add(40);
    System.out.println(list);
}
```

add(val) → adds element at the end

remove(idx) → remove element according to idx

get(idx) → gets value @ idx

set(idx, val) → updates val @ idx if it is



add(2, 4)

```
public static void main(String[] args) {
    ArrayList<Integer> list = new ArrayList<>();

    System.out.println(list);
    list.add(10);
    list.add(20);
    list.add(30);
    list.add(40);
    System.out.println(list);

    System.out.println("val at 3 : " + list.get(3));

    list.remove(2);
    System.out.println(list);

    list.set(0,99);
    System.out.println(list);
}
```

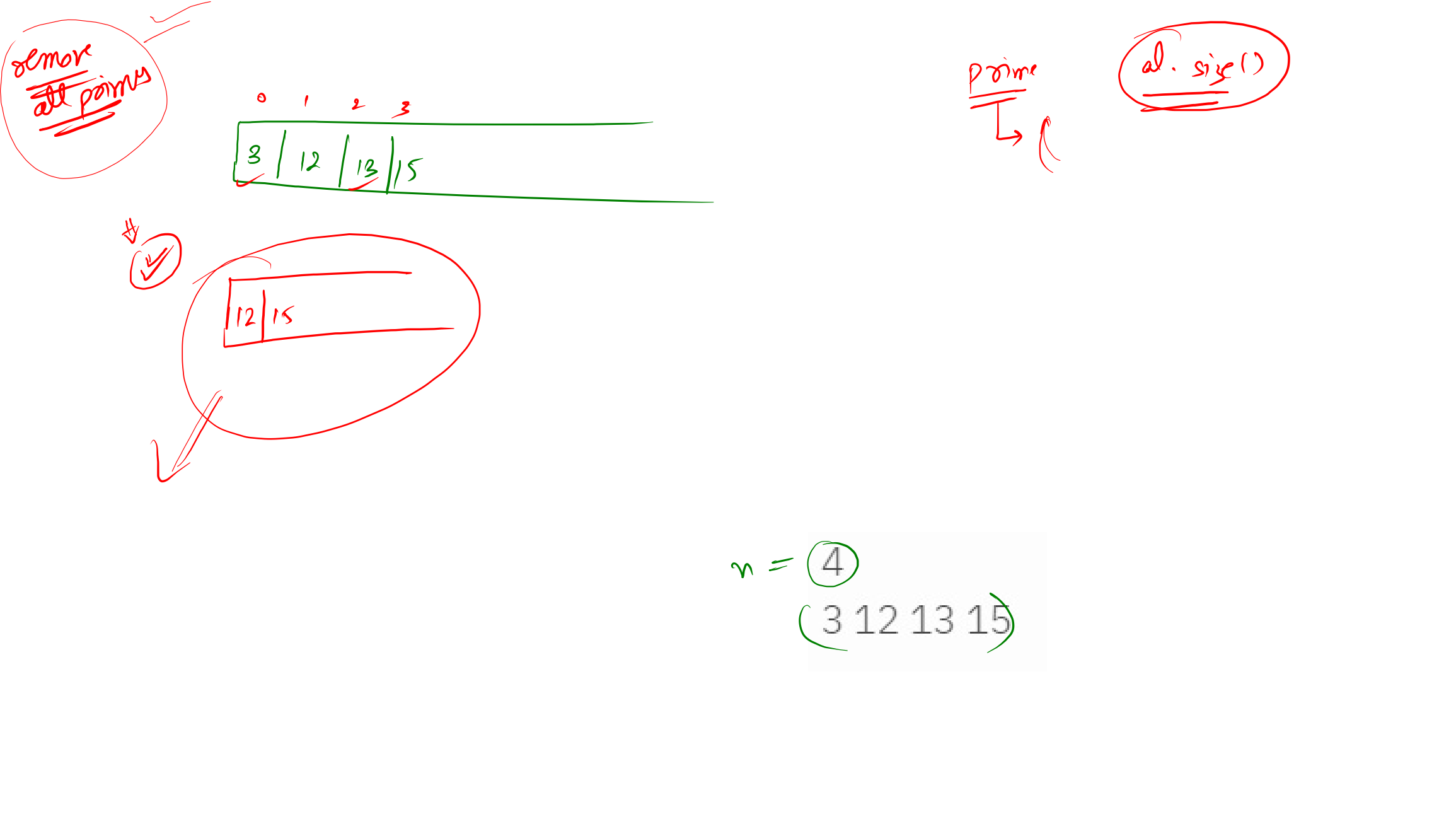
[ ]

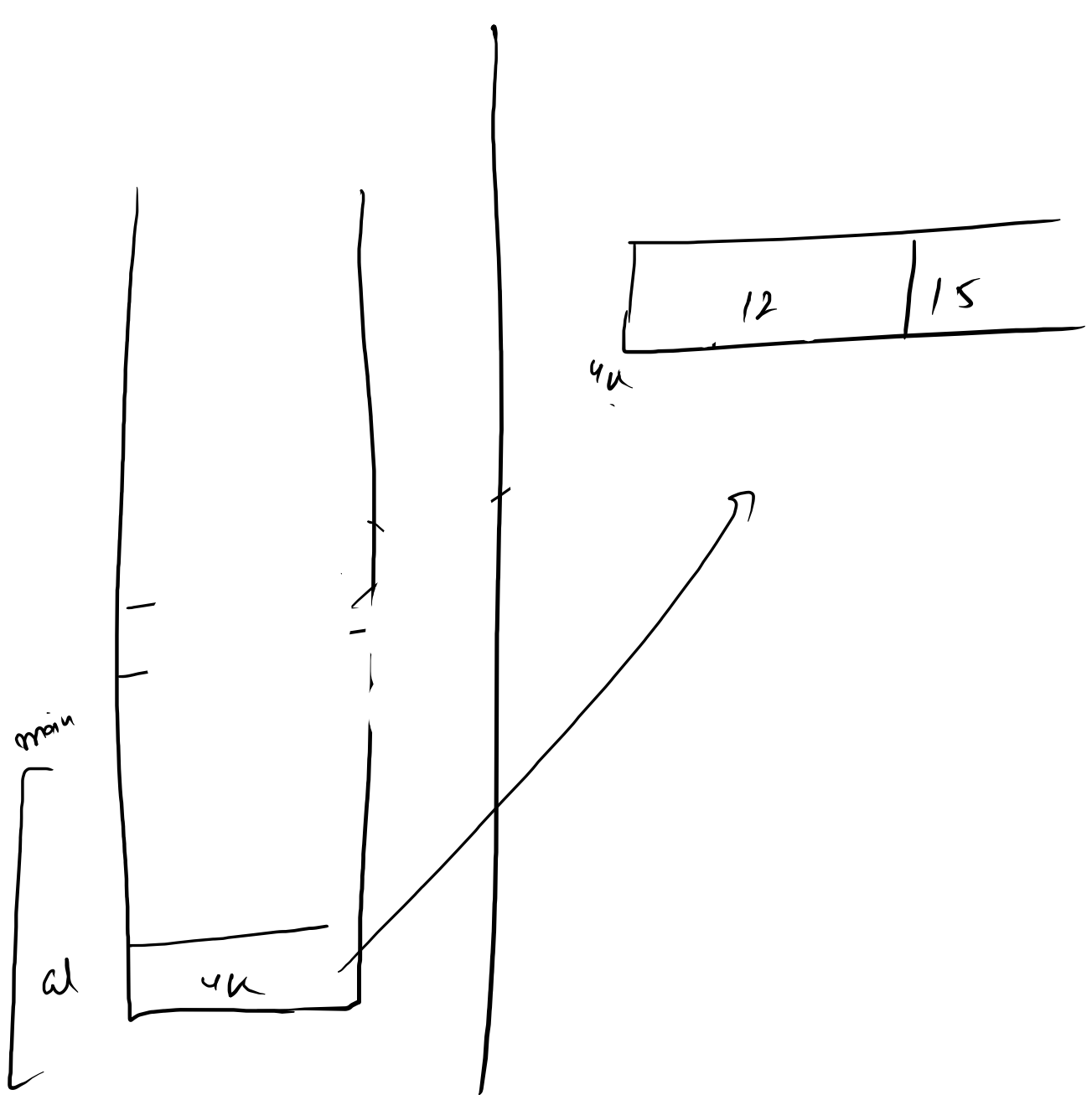
[10 20 30 40]

val at 3 40

[10 20 40]

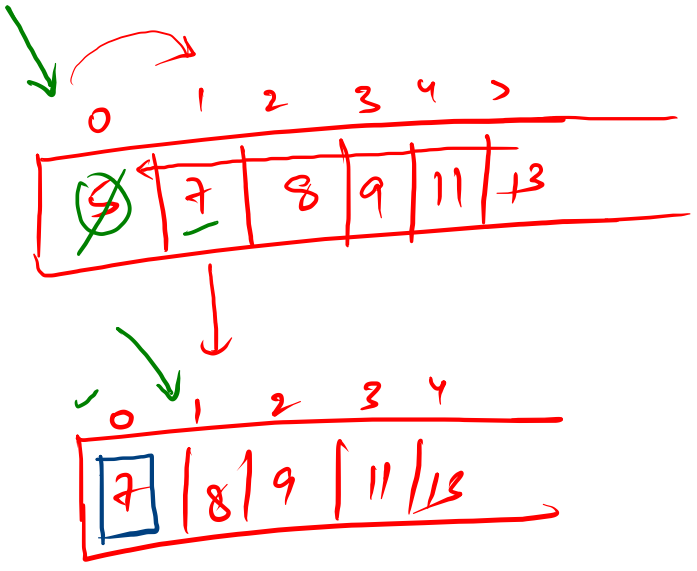
[99 20 40]

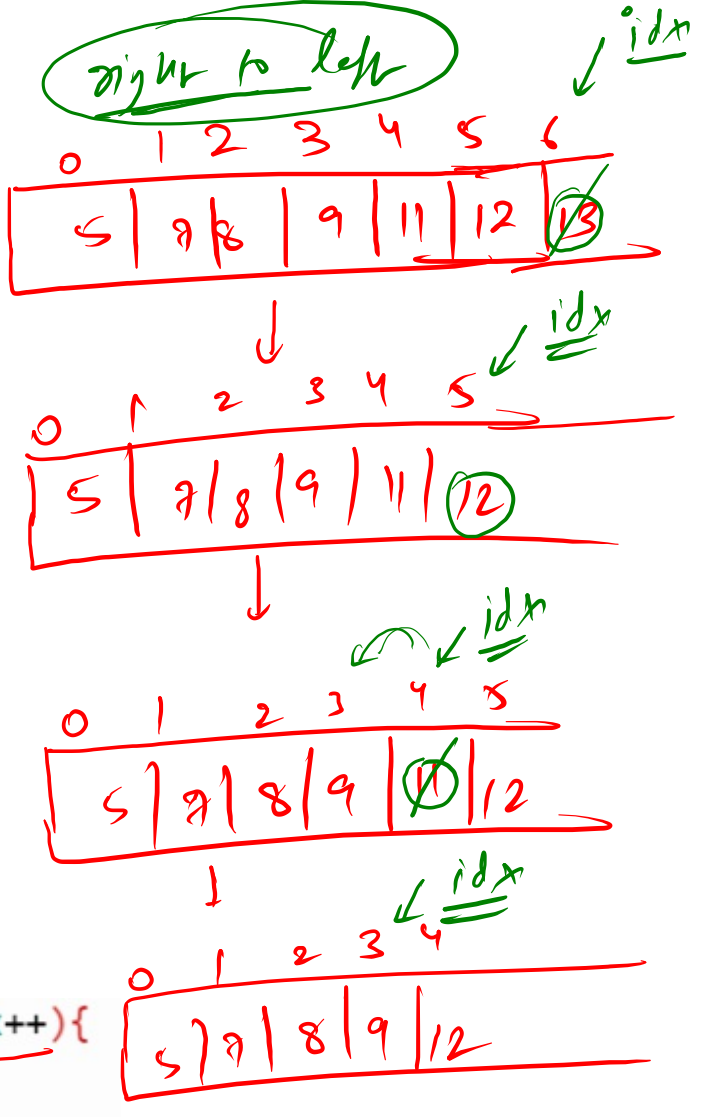

remove all primes
0 1 2 3
3 12 13 15
12 15
prime
al.size()
n = 4
3 12 13 15




```
4
3 12 13 15
```

```java
public static boolean isPrime(int val){
    for(int div = 2 ; div * div < val ; div++){
        if(val % div == 0){
            return false;
        }
    }
    return true;
}
public static void solution(ArrayList<Integer> al){
    for(int idx = 0 ; idx < al.size() ; idx++){
        int val = al.get(idx);
        if(isPrime(val)){
            al.remove(idx);
        }
    }
}
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    ArrayList<Integer> al = new ArrayList<>();
    for(int i = 0 ; i < n; i++){
        al.add(scn.nextInt());
    }
    solution(al);
    System.out.println(al);
}
```





arrayList → left to right removed

↳ element skip

```
for(int idx = 0 ; idx < al.size() ; idx++){
    int val = al.get(idx);
    if(isPrime(val)){
        al.remove(idx);
    }
}
```

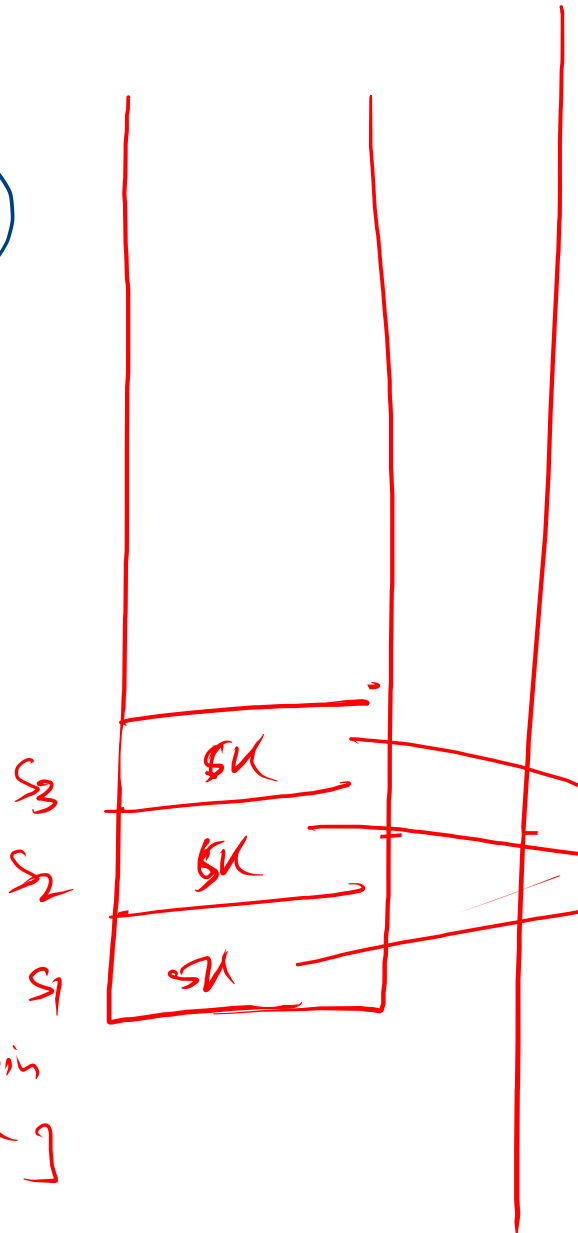
String

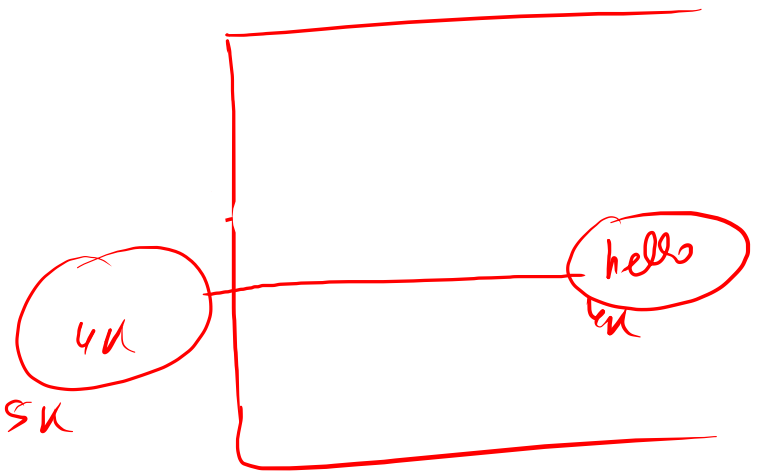
- `String s1 = "hello";`
- `String s2 = "hello";`
- `String s3 = s2;`

String → Intern pool → Memory String

Interning

[the, good morning, a, is, am, ok]





Interning

- Java immutable
- Compare (==) ✓

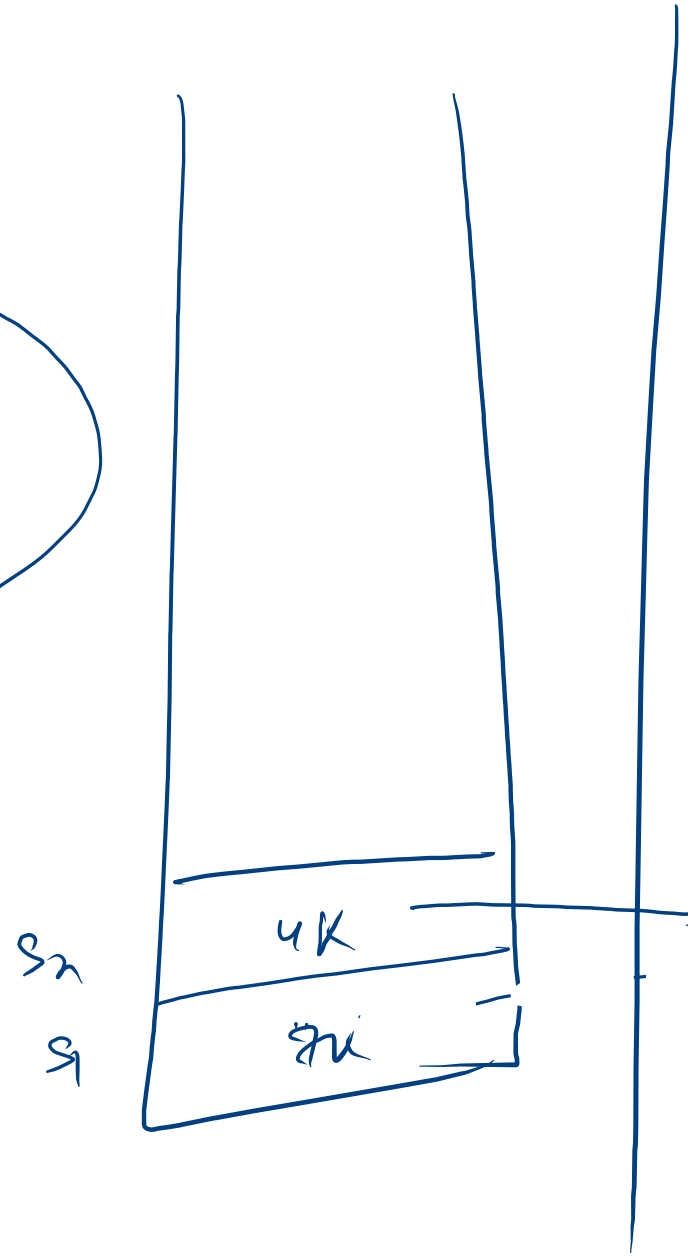
→ String s1 = "hello"

→ String s2 = "hello";

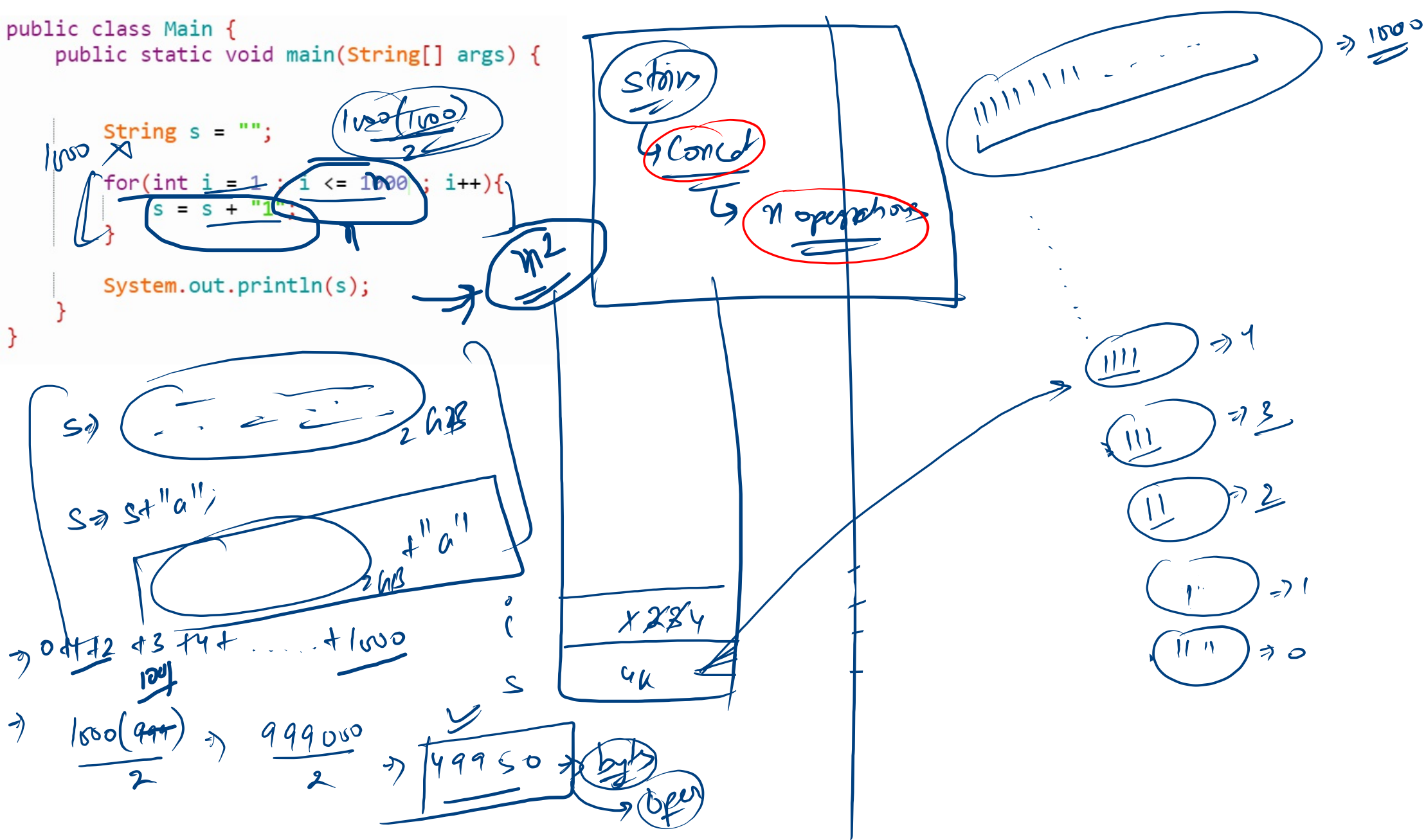
s1 = s1 + " world";

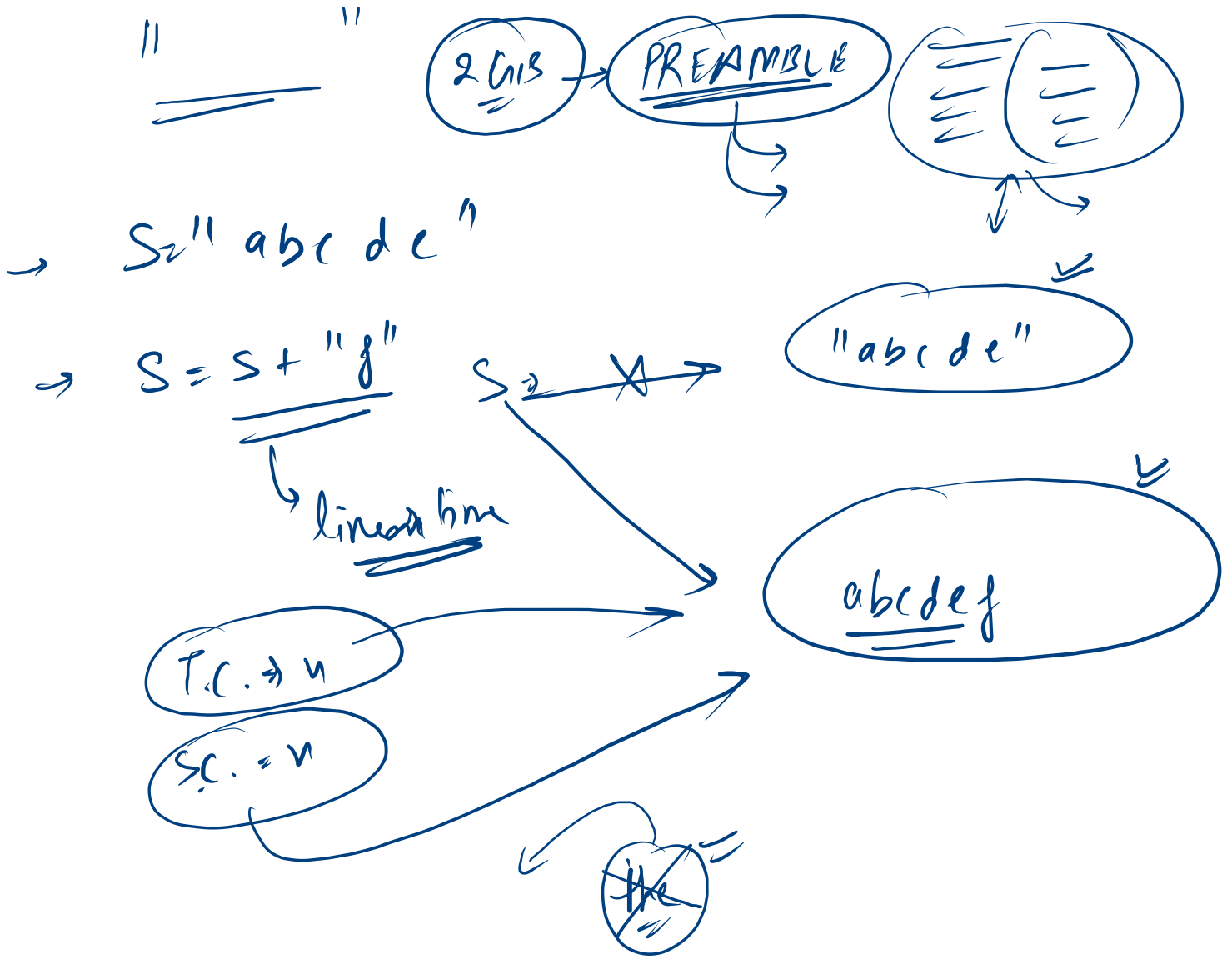
any change in string

↳ will lead to new string



```java
public class Main {
    public static void main(String[] args) {

        String s = "";
        for(int i = 1 ; i <= 1000 ; i++){
            s = s + "1";
        }

        System.out.println(s);
    }
}
```




"
"
2 CHS
PREAMBLE
→ S2 "abcde"
→ S = S + "f"
"abcde"
linear time
abcdef
T.C. → n
S.C. = n
"f"

String s1 = "hello";

String s2 = new String("hello");

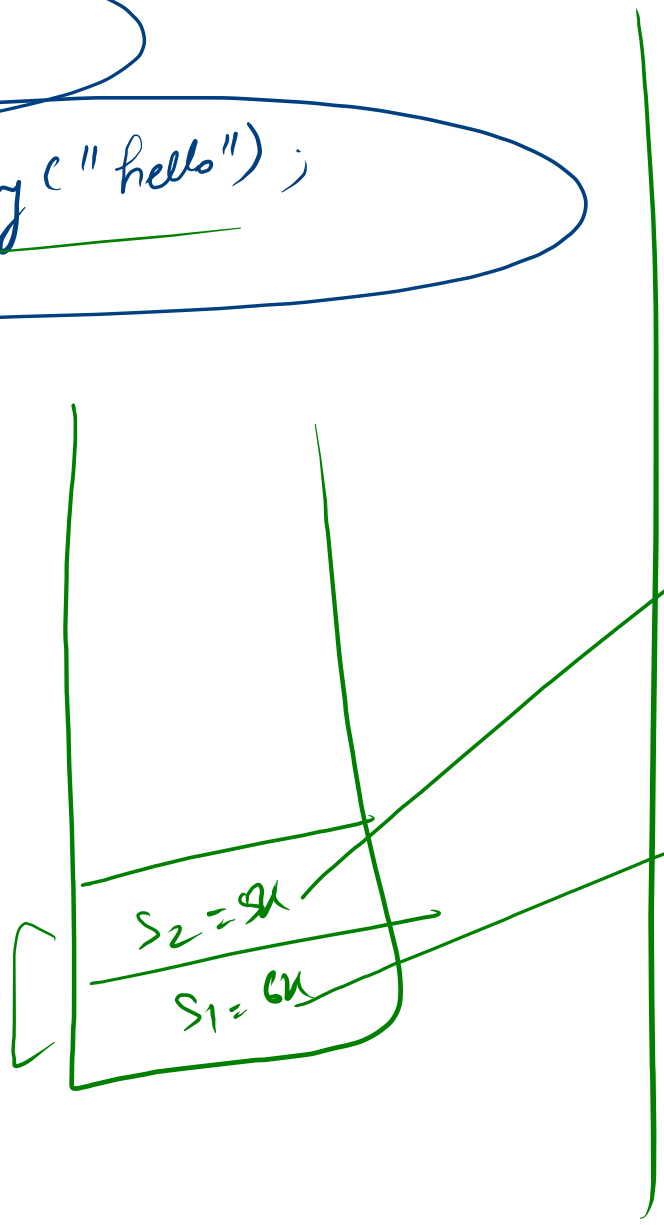
== → Program stuck

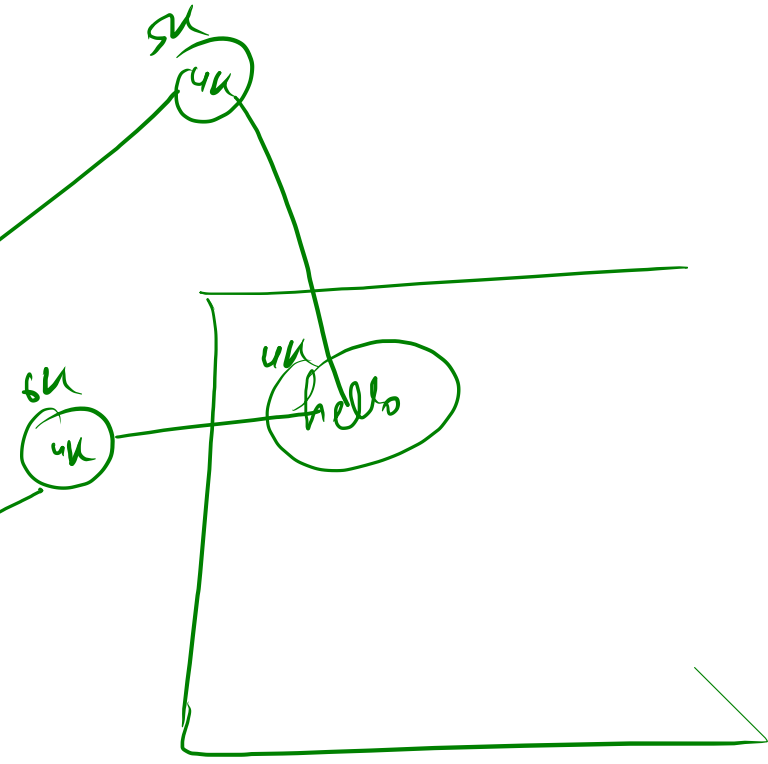
(SM) s1 == s2 (HM)

→ (false)

s1.equals(s2) → true

→ Program stuck = data

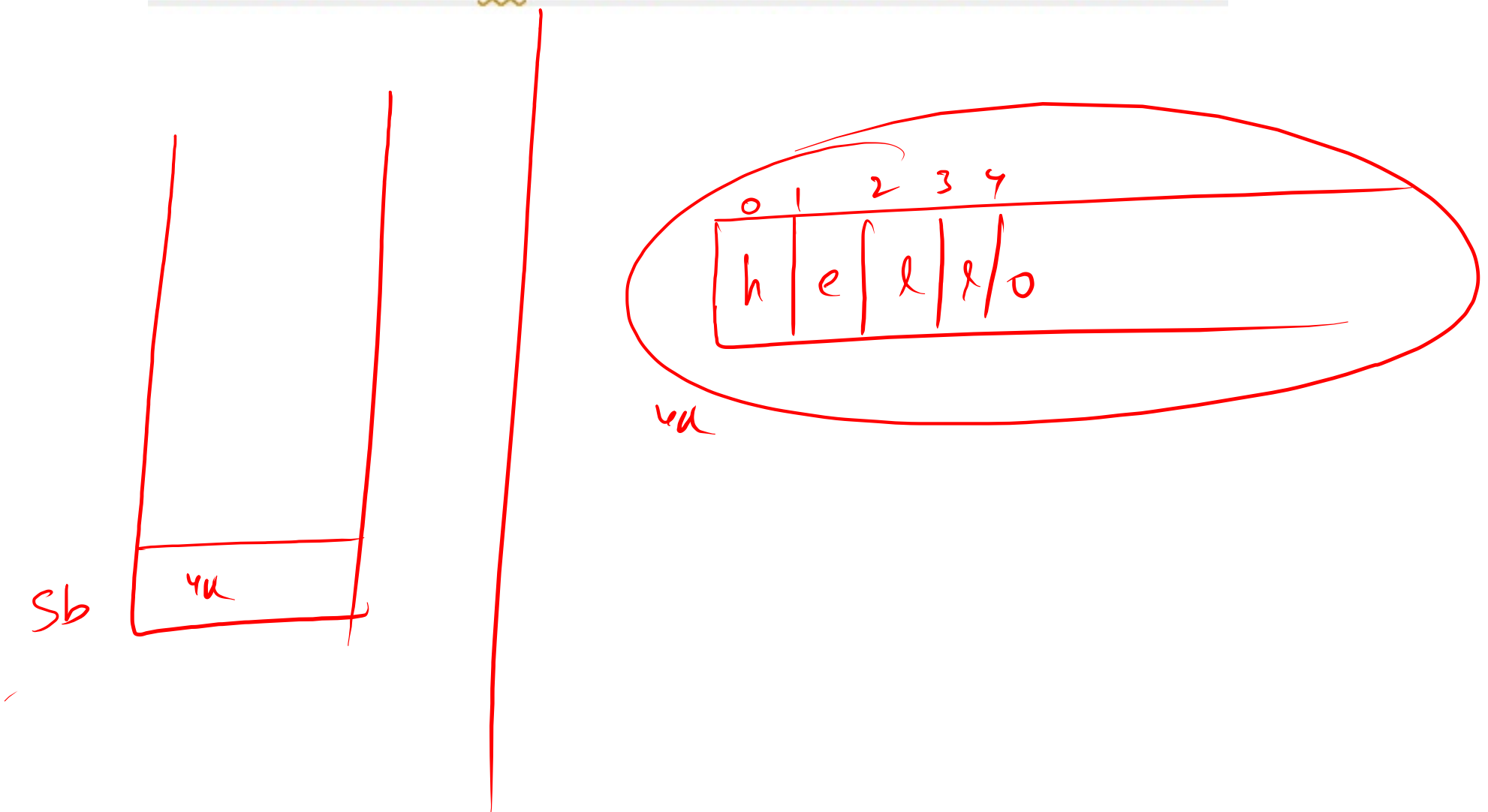


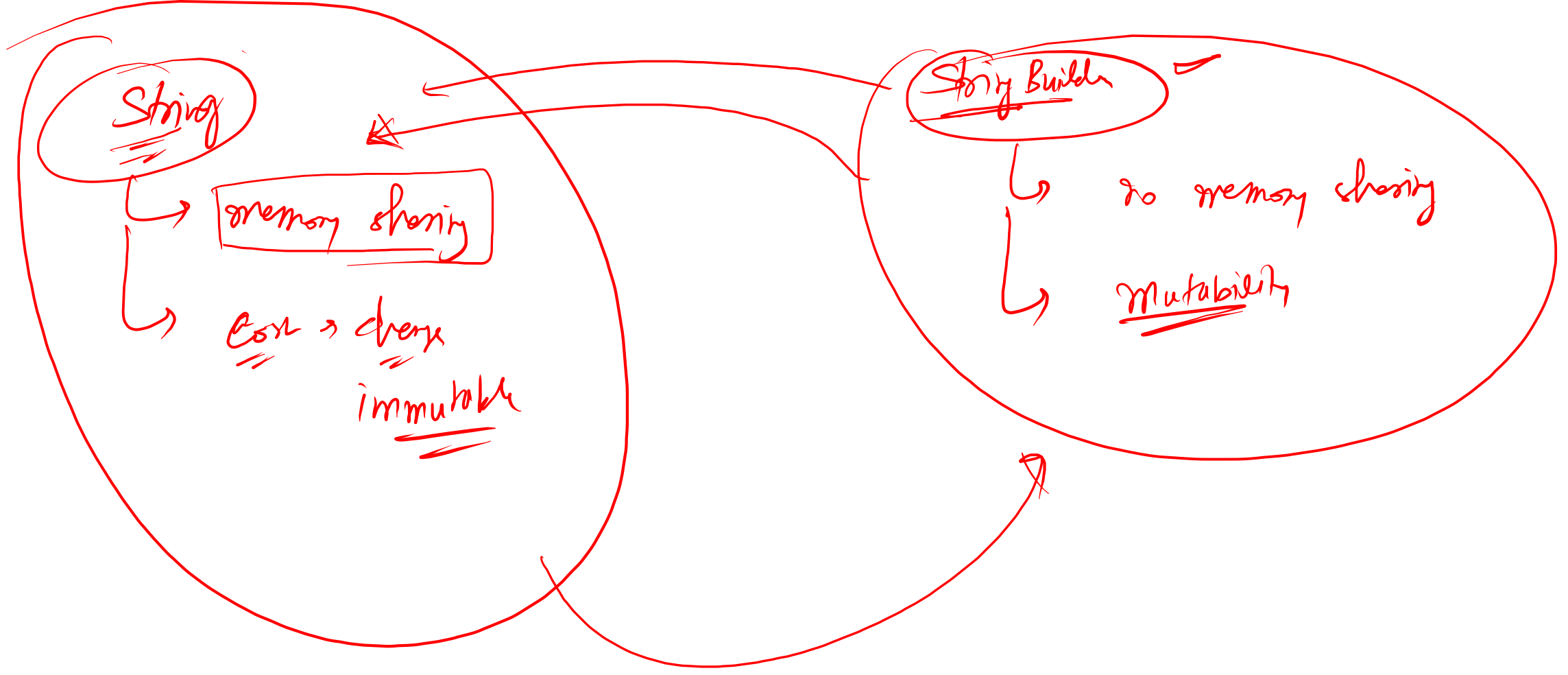


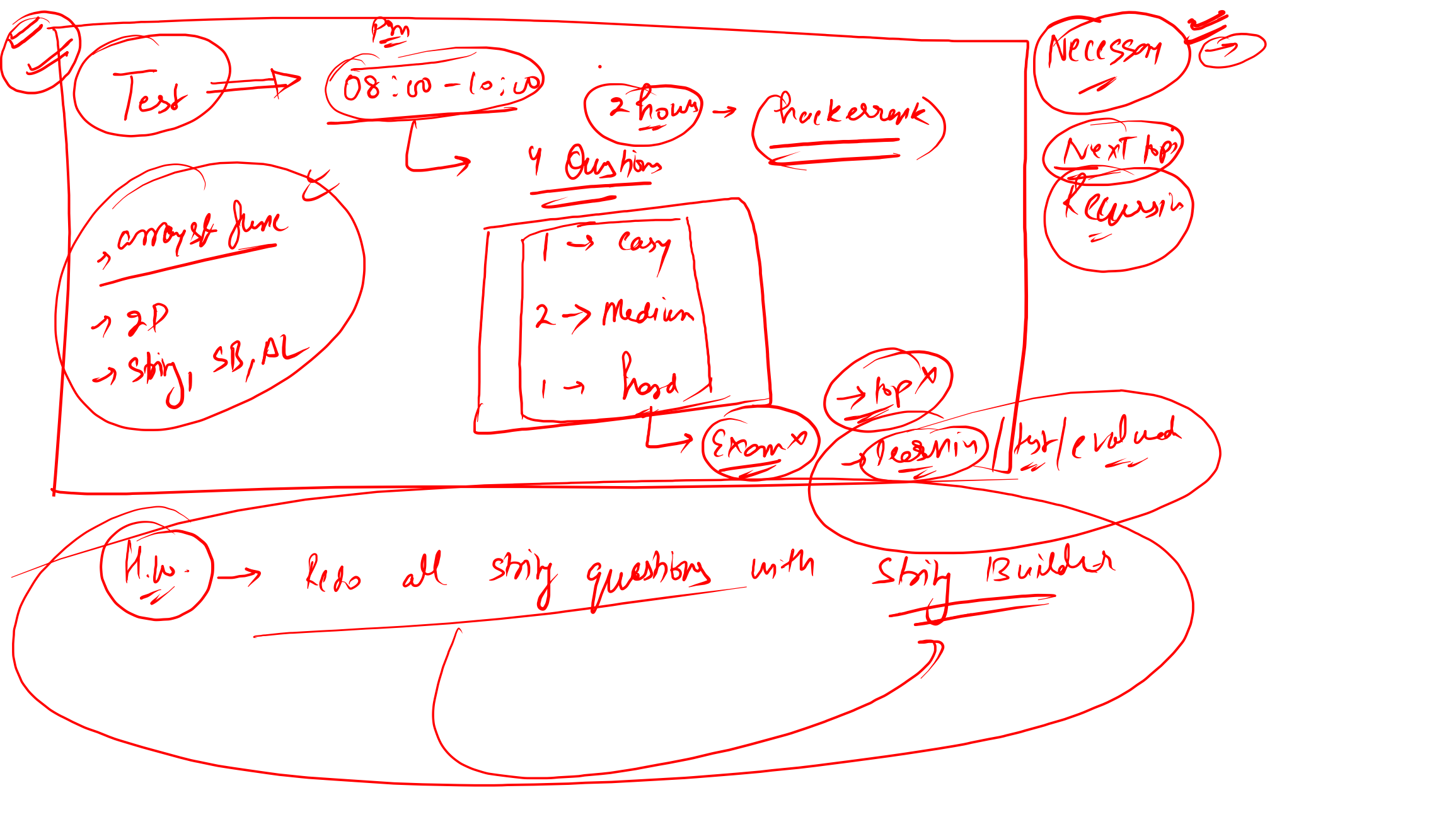
String → StringBuilder → ArrayList + Characters

StringBuilder sb = new StringBuilder( );

```
StringBuilder sb = new StringBuilder("hello");
```




String
memory sharing
Cost → cheap
immutable
String Builder
no memory sharing
Mutability


Test
08:00 - 10:00
Pm
2 hours
hackerrank
4 Questions
1 → easy
2 → Medium
1 → hard
Exam
→ top
→ Reasoning
Not evaluated
Necessary
Next top
Recursion
→ arrays & func
→ 2D
→ String, SB, AL
H.W. → Redo all string questions with String Builder