Strings

Datatype

reference Variable

String name = "Hello"

↳ object

[Anything that starts with a Capital letter is a class]

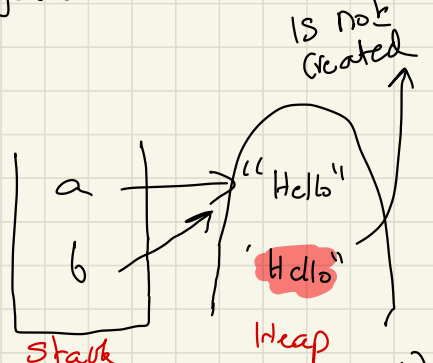
String ⇒ String.java

Ex :-

String a = "Hello"

String b = "Hello"

⇒



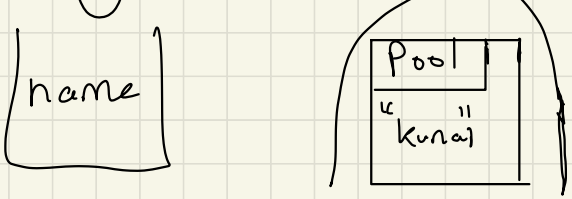
(* Hint :- String pool)

Two important Concepts

① String pool ② Immutability

String Pool :- It is a separate memory structure inside the heap

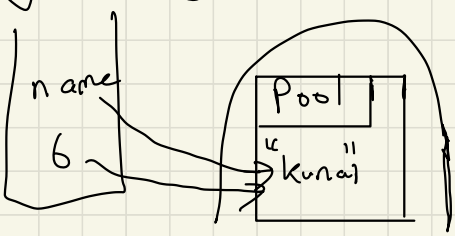
String name = "Kunal"



Note:- All similar values of String are not Recreated

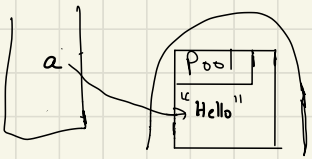
String name = "kunal"

String b = "kunal"



Immutability

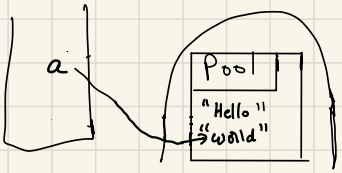
String a = "Hello"



Sout(a) ⇒ prints "Hello"

String a = "world" → we are creating a new object → we are not changing the value of the original object

Sout(a) ⇒ prints "world"



[* The old object "Hello" is now available for garbage collection]

Why Immutability

→ Say there are 10 people and all of them have the same name. So at this point all the 10 people object are pointing to the same name.

"what if one of them had decided to

Change his/her name"

X

Comparison of Strings

① "==" method → Checks both the values and the reference variable

Comparator

a ⟶ "Kunal"

b ⟶ "Kunal"

a==b will be "FALSE"

a ⟶ "Kunal"

b ⟶ "Kunal"

a==b will be "TRUE"

✱ "==" ⇒ Checks if ref variables are pointing to the same object

---

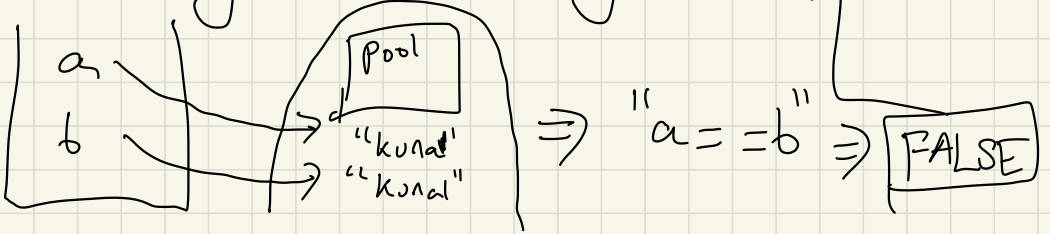
How to create different objects of same values?

⇒ Hint:- use "new keyword"

String a = new String("Kunal");

String b = new String("Kunal");

This creates two string values outside the "String Pool"



String Comparison (Cont...)

(2) When we only want to check and Compare the values alone, we use a ".equals()" method

```
String a = new String("Kunal");
String b = new String("Kunal");
```

Comparison: "a.equals(b)"

---

* Behind the scenes, String is using byte[] (byte array).

To print a character at a particular Index, use

```
String a = "Hello";
Sout(a.charAt(0));
```

↳ prints "H"