# # Strings #

Q. what are strings ?

→

- Strings are a sequence of character. e.g. "Arti"

- It is a non-primitive data-type.

- Syntax : ↗ Everything that starts with a capital letter is a class.

```
String name = "Arti" ;
System.out.println(name);
```

- It is declared in the double quotes.

- It is the most commonly used class in the JAVA's class library.

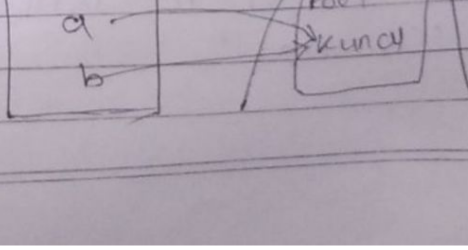
## * Concepts *

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

Q. why separate Pool? why not just putting it out in the heap normally likes every other object?

Ans - All the similar values of strings are not recreated in the pool.

E.g. :- String a = "kunal"
String b = "kunal"



Page No.
Date :

Use case :- It makes our program more optimized

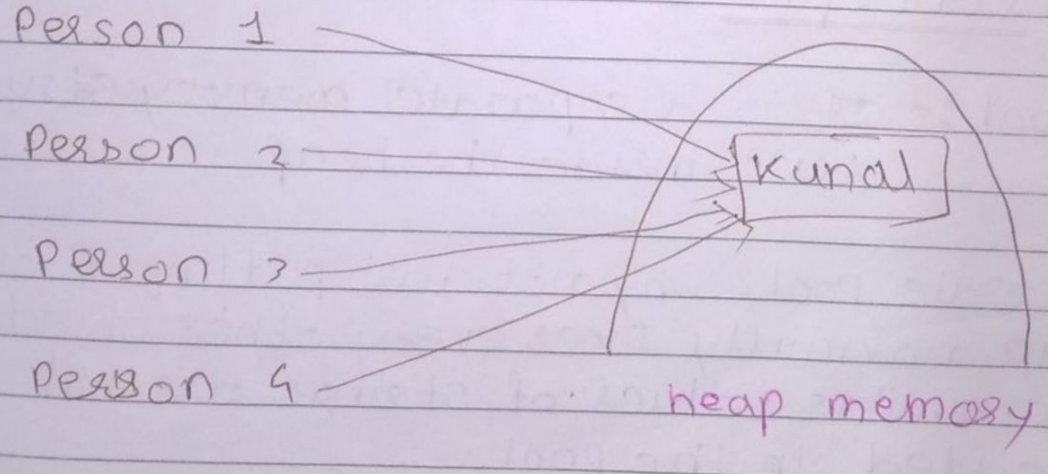
Note :- If you try to change this object via this reference variable it will not change for b

Q. Why ?
Ans :- Immutability.

- strings are immutable in java we can't change the object or modify object

- If you want to rename to "Arti" you have to create a new object for that.

- Strings are immutable for security purpose



Here all the person's name are kunal.
∴ All of them will be having just one object "kunal"

- Suppose one person decides to change their name if it was not immutable, if a person 1 decided to change their name to Karan. So that will change his ~~n~~ name to Karan.

- If it wass allowed to modify this ~~then~~ all person 1, 2, 3, 4 name will be Karan in the database.

- Therefore for the security reasons strings are immutable

## * Comparison of Strings *

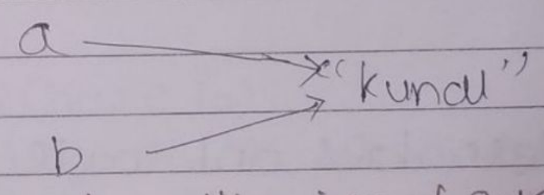
1) == method (Compatitor)

Eg.

a → "Kunal"

b → "kunal

a==b will give false

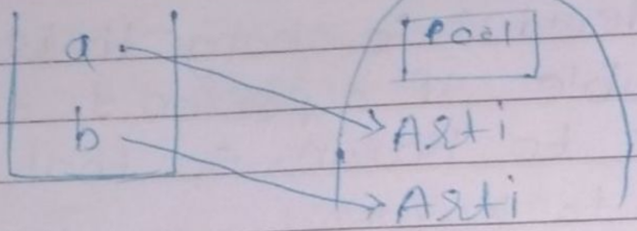


a=b will give true.

> ~~Cam~~ Compatitor actually checks for ~~the~~ bothe the values & the reference variable. If the reference variable is pointing to the same object

Q. How to create different objects of same values
→ String a = new String ("Arti");

String b = new String ("Arti");
// creating these values outside the pool but in heap. because it is object so it will be in heap only.



a = = b //false

Even though the values are same but ~~there~~ the two a & b are not pointing to the same object in that case it will give false.

• when you only need to check value use .equals method or function.

so,

```
String name1 = new String ("kunal");
String name2 = new String ("kunal");

System.out.println(name1.equals(name2));
                          output = ~~false~~true
```

Here does ont care whether the reference variable are pointing to same object or not it just cares about the value.

* class is a name group of properties and fuct$^n$

* We can't do :-

```
System.out.println(name[0]);
```

- In arrays we can do this
- Though string is sort of collection of characte but we cannot do ~~ab~~ above syntax.

* so we have to use method called charA().
System.Out.Println(name.CharAt(0));

# *Pretty Printing —

Placeholder

float a = 453.1234f;
System.Out.Printf("Formatted Number is %0.2f", a);

Formatted string

name declared

- well % means a placeholder & till how many decimal value do we want
- It rounds of as well.
- For π :- System.out.print~~ln~~(Math.PI);

- Some common formate specifier.

1) %c :- character
2) %d :- Decimal number (base 10)
3) %e :- Exponential floating-point number
4) %f :- floating-point number
5) %i :- Integer (base 10)
6) %o :- Octal number (base 8)
7) %s :- String
8) %u :- unsigned decimal Integer number
9) %x :- Hexadecimal number (base 16)
10) %t :- Date/time
11) %n :- Newline.