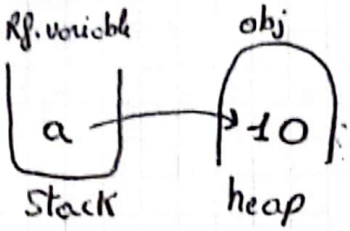
• ILYASSE YOUNES

01/08/2024

String & String Builder

• (1) Stack memory :
When we declare a variable eg : int a = 10 so, the reference variable is stored in stack memory.

• (2) Heap memory :
Reference var is stored in stack memory is pointing to the obj of that variable are stored in heap memory.



→ Stack :
- it performs type checking at Compiling time
- errors will show at compiling time
- Declare data type before you use it.
- More Control & Runtime errors are reduced.

→ Heap :
- performs type checking at run time.
- Here, errors might not shown at when the program runs.
- No need to declare a data type of a variable.
- it save time in coding but it might give errors at run time.

• garbage collection :

- More than one reference variable point to the same obj.
- If any exact change made of in the obj on reference var that will be referenced to all others pointing to the same obj.
- If there an obj without reference var then the obj will be disabled by garbage collection
  ↳ that how garbage collection work.

• What's String :

- "string will be in double quot".
- string is a sequence of caracters
- it a data type not a primitives.
- Syntax :

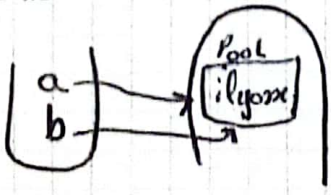
```
String nom = "ilyasse younes";
```

Note : everything start with a capitale letter is a Class.

- Every string that you creat it's actualy an obj of type string.

• **string Pooling** : is a seperad memory structure inside the heap.
Why Speacial pool ? Why not just putting it normaly in the heype like other obj ?
↳ All similar value of string are not recreted in the pool.

```
String a = "ilyasse";
String b = "ilyasse";
```



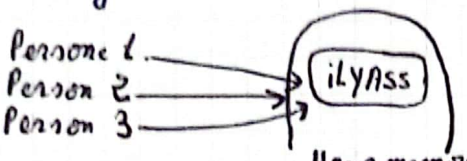
Use Case : it make the programme Optimised.

• Note :
If you try to change this obj via this reference var will it not change.

Why ?

• String immutability :

- String are immutable in Java we can't change or modify obj.
- If you want to renam "ilyass" you've to creat a new obj for that.
- String are immutible for Security purpose.



↳ if we allow P1 to change to yassine then all P2, P3, will change in data base.

• Comparaison of String :

== methode

a → "younes" } a==b will give false
b → "younes"

a → "younes" } a==b will give True.
b →

+ Comparater actully check for value and Reference var if it pointing to the same obj.

How to creat diff obj of same value ?

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

+ When you need to check obj for value Use .equal method or fun.

```
sout(a.equal(b)); → True.
```

• Note
We can not do :
sout(name1[0]);
We have a method called charAt();

```
System.out.println(name.charAt(1));
```

↳ var of type printstream → class it has method in it called println.

• Function Overloading :

• Fun overload happen When Tow Fun have Same name.

eg: fun(){...}
fun(){...}

• Pretty Printing in Java :

```
float a = 4.33f;
System.out.Print("number is %.2f", a);
```

• % is a place holder.

→ Some common format :

%c	character
%d	Decimal number (base 10)
%e	Expontial floating-point num
%f	floating-point num
%i	Integer (base 10)
%o	octal num (base 8)
%s	string
%u	unsigned decimal (integer)
%x	Hexadeimal (base 16)
%t	Date/time
%n	New line

• Operator Overloading :

→ Operator "+" overloaded for string type.

① Sout('a' + 'b') → 195
↳ Here the operator is converting this char into integer value and then adding it Unicode or Ascii value.

② Sout("a" + "b") → ab

③ Sout ("a" + 3) → a3

④ Sout ((char)('a'+3)); → d
↳ converting 100 into character

⑤ Sout ("a" + 1); → a1
Note: When an integer is concatenated
- With a string it is converted to wrapper class integer same as "a" + "1".
- Integ will converted to Integ that will called Tostring();

⑥ Sout ("ilyas" + new Arraylist<>(); → ilyas[]

⑦ Sout (new Integ(56) + new Arraylist<>());
↳ Error
the only condition is at least one of these obj should be of type string.
↳ Sout (new Integ(56) + " " + new arraylist<>());

- In Java Operator overloading is not supported for some software engineering consideration but in C++ is supported.
- So, you can modify what '+' is doing in C++ and python.
- you can also make it act like 'x' or '-' or you can add complex datatype as well.
- But, this results in poor code, that is why in java it not supported.
- it's operator that is intentionally overloaded in java to support string concatination or string joining.

• String Builder Class :

Code :
```
String series = "";
for (int i=0; i<26; i++){
  char ch = (char)('a'+i);
  series += ch;
```

① series = ""
② "" + "a"
③ "" "a" + "b" = "ab"
④ "" + "ab" + "c" = "abc"
...

* Here, the new obj is being created everytime, it not changing the origine obj because String are **immutable**.

* So much Wast of memory because of this and it won't Having any reference var, So time Complexity = $O(n^2)$ Worst.

* The solution to this is string Builder, and in it one obj # is made and the changes are done in that obj only and the reference will not change.

New Code :
```
StringBuilder builder = new StringBuilder();
for (int i=0; i<26; i++){
  char ch = (char)('a'+i);
  Builder.append(ch);
```