Var args [Variable no of arguments]

```
int add(int... args) {
    int sum =0;
    for(int i=0;i<args.length;i++) {
        sum+=args[i];
    }
    return sum;
}
```

args = [0,0]

add (10,20) ← args = [0,0,0]

add (10, 20, 30)

Notes:

→ In above program method add() can accept any no of arguments of same data type which we call it as "var args"

→ Var args are such methods which can accept any no of arguments of same data type

→ The argument which it collects & stores in the array format

→ The elements passed can be accessed through index values

Alternative ways of declaring var args

```
int add (int... args) { }

int add (int ... args) { }

int add (int ... args) { ... }
```

Rules & Regulations

(1) we can have only one var args in a method

```
int add ( int... ar, double ... br) { ... }
```

(2) In case if we are having var args along with named parameters, then the var args should be declared at the last

```
add ( int x, int y, int ...args) { ... }
```

Calc

(1) Calc.add(5) ✗

(2) Calc.add(5,3,2) ✓

(3) Calc.add(10,20,30,40,50,60) ✓
x, y — args

(4) Calc.add() ✗

(3) if we have 2 methods of same (method overloading) where one method accept normal parameter & other method accepts var args as element, then preference is given to the normal parameter

Compiler decision

```
int add(int x, int y){
```
x, y

① }

```
int add(int... args){
```

② }

Calc.add() ← Calls 2

Calc.add(10) ← Calls 2

Calc.add(10,20) ← Calls 1

Calc.add(10,20,30) ← Calls 2

Strings

Def: String is a collection of Characters enclosed with in a " " double quotes

' ' ← Char

" " ← String

(1) In case of Java String is a object

(2) We can create a string using new keyword as well as without new keyword

(3) Strings are immutable in java

(4) Immutable objects are such objects which doesn't change after the object creation

Different ways of Creating Strings

(1) String s1 = new String("ABC");

→ (2) String s2 = "ABC";

(3) Char a1[] = {'A', 'B', 'C'}

String s3 = new String(a1)

String Constant Pool: It is a region present inside the Heap Area

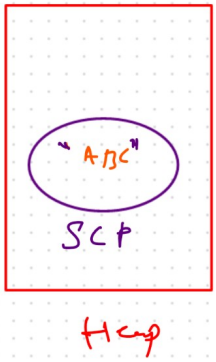
String s = "ABC"

String s1 = "ABC"

→ Only string objects gets created inside the region.

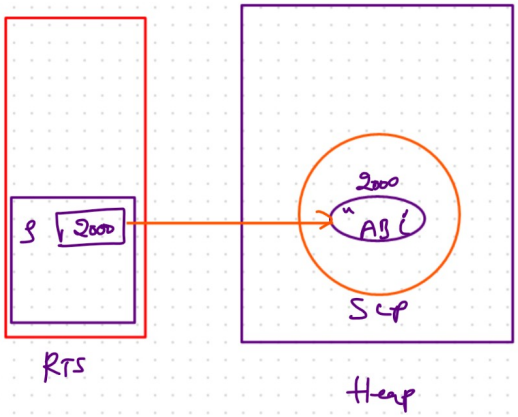
→ No duplicate copies are allowed inside the region

→ Even if no one is refering to the String Object in this region, that memory does not get deallocated



Case 1 :

String s = "ABC"

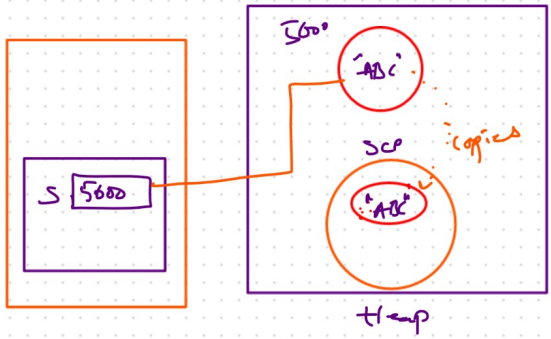


Case 2 :

String s = new String ("ABC");

Note:

(1) when ever there is "new keyword" associated while creating String object it Creates a string object first outside Non Constant pool but inside the heap Area & returns that address

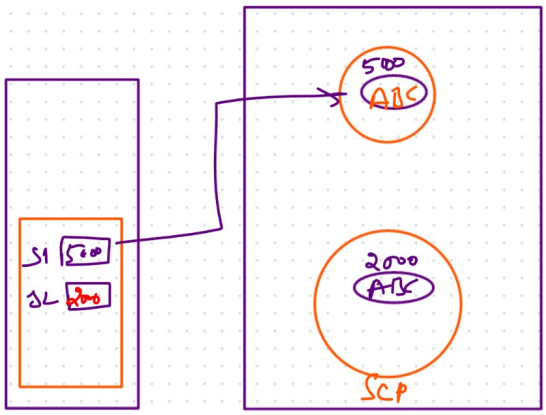


(2) Copy of the same object value will be created inside the Constant pool only if that value doesn't exist

Case 3 :

String s1 = new String("ABC");

String s2 = "ABC"



Notes- "abc" == "abc"

(Object == object)

s1 == s2

Address == address

5000 = 2000 ✗

int a = 10
int b = 10

(Value == value)

10 == 10 ✓

Case 4:

(1) String s1 = new String("ABC")

(2) String s2 = new String("ABC")

(3) String s3 = new String("ABC")