28th oct/2021

Q → Difference between length() Vs length

length

→ It is a final Variable applicable only for arrays.

→ Represents the size of array

```
int[] a01 = new int[10];
S.o.P(a01.length);
```

o/p = 10.

```
S.o.P(a01.length()) //C.E
```

length()

→ It is a final method applicable only for String objects.

→ Represents the no. of Characters in String

```
String s1 = "Hello";
S.o.P(s1.length()); //5
S.o.P(s1.length); // C.E
```

Q

```
Public Class LenArr {
    Public static void main (String[] args) {
        int a88[] = {15, 87, 34, 18, 10, 15, 67};
        for (int i=0; i<a88.length-1; i++) {
            S.o.Pln(a88[i]); // 15 87 34 18 10 15 67
            S.o.Pln(a88.length()); // CE
            S.o.Pln(a88.length); // 7777777
        }
    }
}
```

*) Why length & length() on int datatype

A) => length()

length method cannot invoke methods on primitive type since they are not objects.

=> array.length:- length is a final variable applicable for arrays. With the help of the length variable, we can obtain the size of the array.

=> String.length():- length() method is a final variable which is applicable for String objects. the length() method returns the number of characters present in the String.

*) length Vs length()

=> The length variable is applicable to an array but not for String objects whereas the length() method is applicable for String for String objects but not for arrays.

=> To directly access a field member of a array we can use .length; whereas .length() invokes a method to access a field member.

Ex:-

```
Public Class Test {
    Public Static Void main (String[] args) {
    int [] arr = new int [4];
    System.out.Println ("The size of the array is" +
                                    arr.length);
    String str = "Geeks-For Geeks";
    System.out.Println("The size of the String is" +
                                    str.length());
    }
}
```

O/P => The size of the array is 4
The size of the String is 13.

*)
```
Public Class Test {
    Public Static Void main (String[] args)
    {
    String [] str = {"Hello", "Hi"};
    System.out.Println (str.length); // 3
    System.out.Println (str.length()); // Error
    }
```

O/P = 2

*)
```
Public Class Test {
    Public Static Void main (String [] args)
    {
    String [].
```

*) Sorting Arrays.

```
Public class SortArr {
    Public static void main(String[] args){
    int arr[] = new int[] {20, 15, 55, 42, 32};
      int temp = 0;
      for(int i=0; i<=arr.length; i++){
        for(int j=i+1; j<arr.length; j++){
          if (arr[i] > arr[j]){
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
          }
        }
      }
    for(int i=0; i<arr.length; i++){
      System.out.println(arr[i]+" ");
    }
  }
}
```

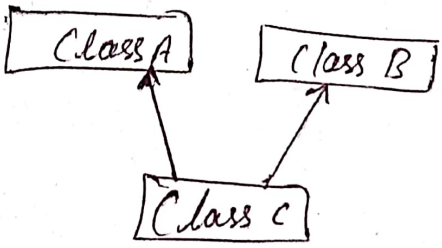
O/P = 15, 20, 32, 42, 55.

*) For each loop :- The for each loop is used to traverse array or Collection in Java. It is easy to use than simple for loop because we don't need to increment value and use suf script notation.

Ex:-
```
for (datatype variable : array-name){
    // Code to be Executed.
}
```

```
Public class ForEach{
    Public Static Void main(String[] args){
        int arr[]={12, 55, 36, 72};
        for(int i : arr){
            System.out.Println(i);
        }
    }
}
```

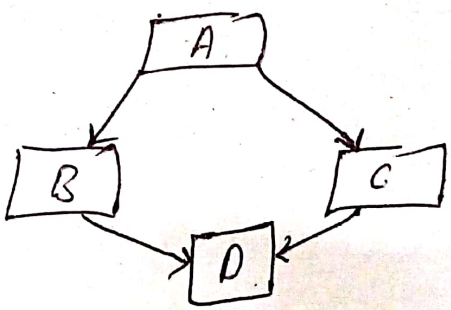
Multiple level Inheritance:-




```
Class Parent1{
    Void fun(){
        S.o.Pln("Parent-1");
    }
Class Parent2{
    Void fun(){
        S.o.Pln("Parent-2");
    }
Class Test extends Parent1, Parent2{
Public Static Void main(String[] args){
    Test t=new Test();
    t.fun();
}}
```

⇒ method Ambiguity will occured to access ~~fun~~ method from Parents Classes.

Hybrid Inheritance:-



It is a mix of two or more of the above types of Inheritance. Since Java doesn't support multiple inheritance with classes. We can not achieve hybrid inheritance by class but we can achive by Interface.