Compiled Languages : C, C++
* Stored binary
+ performance,faster exec
- platform dependent

procedural
object-oriented
functional

WORA

Interpreted Langauges : Python
* No stored binary
+ portable
- slower performance

javac HelloWorld.java # Bytecode : HelloWorld.class
java HellWorld

JVM (Java Virtual Machine) part of JRE

https://github.com/caia-techblr/java-examples

Account
FoodDelivery
Ticket
ShoppingCart
Movie
Customer
Patient
Feedback
Billing

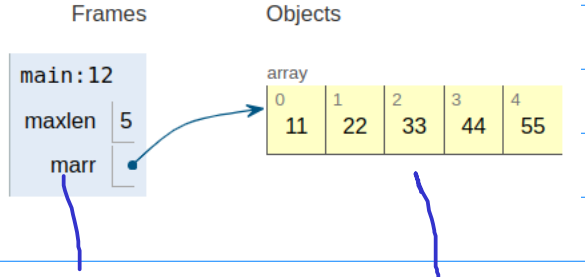
Student
Employee
Box
Point
Color
IPAddress

Unit-3:-
1D Arrays
2D Arrays
Irregular Arrays
Array Utils
ArrayList

Array creation:-

int m1, m2, m3, m4, ,m5;

int marr[]; // ref only
//int[] marr;
marr = new int[5]; //on heap



stack frame

heap

Memory sections:-
* Code
* Stack : local vars, params
* Heap : dyn mem

```
int sum(int x,int y)
{
        int res;
}
```

Box b1;
b1 = new Box(10,12,5);

b1 is a reference, stored on stack frame
Object (12 bytes) stored on Heap : dyn memory

Operations on Array elements:-

* Sum & average
* Min, Max
* Reverse
* Sort
* Search
* Compare
* Fill
* Populate
* Copy

---

* Rotate
* Shift

```
class Sample                                    instance variables
{
        int x;
        int y;
        static int k;
};

Sample s1, s2, s3;  //references
s1 = new Sample();
s2 = new Sample();
s3 = new Sample();

s1.x=10;        s1.y=20;
s2.x=11;        s2.y=21;
s3.x=12;        s3.y=22;

s1.k=100;
s2.k=200;
```

```
int[] numbers = { 11, 12, 13, 14, 44, 55, 66, 77 };
int len = numbers.length;

int sum=0;
for(int i=0;i<numbers.length;i++)
        sum += arr[i];
doubel avg = sum/len;

//enhanced for loop
for(int val : numbers)
        sum += val;
-----------------------------------          { 22, 11, 44, 55, 33, 77, 88, 66 }
int maxval = arr[0];                         0 and 7
int maxpos=i;                                1 and 6
for(int i=1;i<arr.length;i++)                2, 5
        if(arr[i] > maxval) {                3,4
                maxval = arr[i]
                maxpos = i;
        }
```

Reverse:-

```
for(int i=0;i<len/2;i++)
        //swap arr[i], arr[len-i-1];
```

Linear Search:-

Binary Search
Bubble Sort

```
import java.util.Arrays
```

Ref:- https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html
- binarySearch
- sort
- fill
- copyOf
- equals

visualgo.net

0	1	2	3	4	5	6	7	8
10	20	30	40	50	60	70	80	90

```
int arr[] = {10,20,30,40,50,60,70,80,90};

int index = java.util.Arrays.binarySearch(arr,key);

int nums = { 22, 11, 44, 55, 33, 77, 88, 66 };
java.util.Arrays.sort(nums);

int arr[] = new int[maxlen];
java.util.Arrays.fill(arr, 8);
```

https://github.com/caia-techblr/java-examples/

Types (classes,interfaces) -- Camel Casing, starts with uppercase
Fields (member variables, methods) - Camel Case, first word all in lowercase
final constants -- all upper case, separate multiple words by _
package names - single word, lower case

BoxDemo
findVolume
pricerPerUnit
MAX_VALUE NO_OF_SUBJECTS

StudentScoringAnalysis.java
==> class Student
==> public class StudentScoringAnalysis, will have main function

```
int scores[] = new int[MAX_NO_OF_SCORES];

String skills[] = new String[MAX_NO_OF_SKILLS];
```

Limitations of normal array:-
* Fixed memory
* Add/Delete operations - we should impl

In consecutive array:-
* Adding elements at end of array : O(1)
* Remove elements at end : O(1)
* Insert/delete at middle/start : O(n)

What is the complexity
- BinarySearch
- BubbleSort, SelectionSort,InsertionSort
- QuickSort, MergeSort

O(1)
O(logn)
O(n)
O(n logn)
O(n 2)

```
c = a + b;
```
==> O(1), constant

```
for(i=0;i<n;i++)
    sum += arr[i];
```
==> O(n), linear

```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
      //some
```
==> O(n 2), square

java.util package ==> ArrayList

```
//ArrayList is a generic class
ArrayList<Integer> alist;    //reference
alist = new ArrayList<>();    //object creation
//alist = new ArrayList<Integer>

alist.add(11);        //index :0, append
alist.add(22);
alist.add(33);
alist.add(44);
alist.add(55);        //index: 4

int sum=0;
for(int val : alist)
        System.out.println(val);
        //sum += val
alist.insert(3, 40);
```

Wrapper class

Integer
Float
Double
Byte
Long
Short
Char

```
alist.remove(2);
//alist.remove(33);
alist.remove(Integer.valueOf(33));
-----------------------------------------------

ArrayList<String> skills;
skills = new ArrayList<>;
//ArrayList<String> skills = new ArrayList<>;

skills.add("C Prog");                    skills.insert(2,"Python");
skills.add("TEP");                       skills.remove("UHV");
skills.add("UHV");                       skills.remove(4);
skills.add("Java");
skills.add("MySQL");

System.out.println(skills);       //skills.toString()

skills.get(1);   //TEP
skills.get(4);   //MySQL
skills.get(9);  //Error/Exception
```