JAVA

JAVA (1995).

- Less Memory
- OOPS
- Portable
- Freely Downloadable Soft.
- Closed Source.

.java → HLL → java compiler → .class → ILL bytecode → JVM (Windows) → MLL → Windows OS ✓

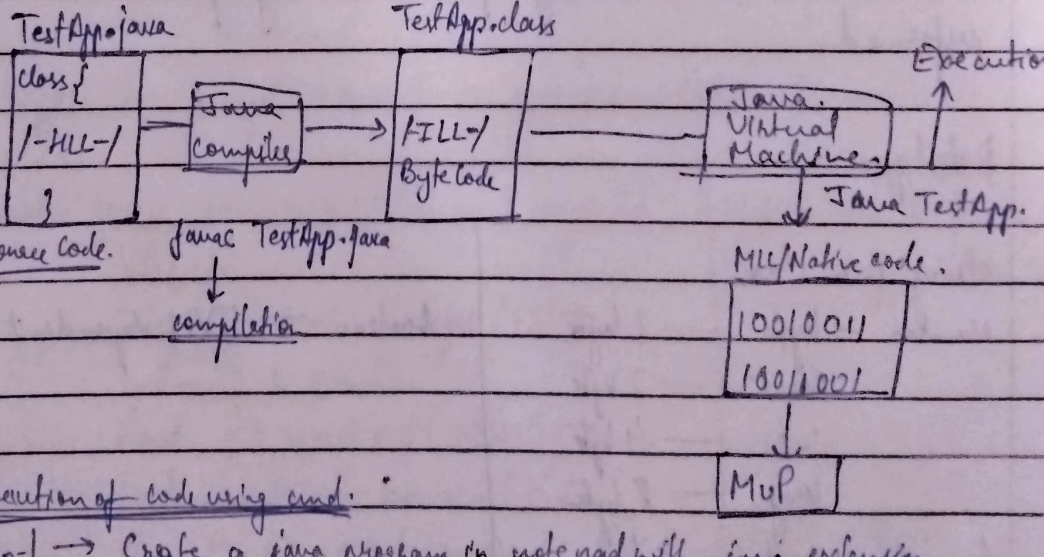
(ILL-Intermediate Level Language)

WORA
Write once RUN Anywhere.
Neither in HLL
Nor in MLL

JAVA → Banking Applications. { Security, Specific }
Enterprise Level Applications.

JDK → IDE → Eclipse IDE → STS ✓
IDE → Netbeans IDE
IDE → Intellije.

Execution Architecture.



Execution of code using cmd.

* Step-1 → Create a java program in notepad with .java extension.
Step-2 → Open terminal or CMD at the file location.
Step-3 → Run "javac filename.java".
Step-4 → java filename.

PAGE No
DATE

For example.

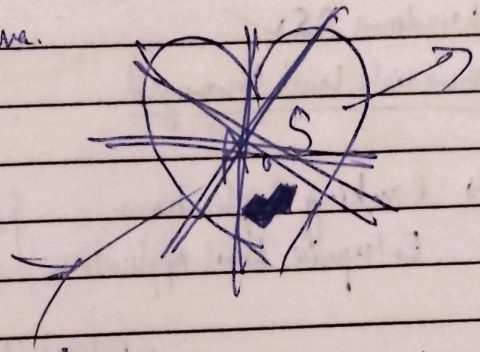
```
class TestApp{
    public static void main (String java[]) {
    System.out.println("Welcome to Full Stack Training"); }
}
```

(java[] → arg[] (arg))

Save above as TestApp.java.

→ javac TestApp.java.

→ java TestApp



* Entry point → main.
Without creating an object by using the class name than that method should be defined as static.

* String is a java class.
String is immutable because the contents cannot be changed, instead it's it create a new string object when any operation on string is performed.

Datatypes.

string is a class.
Numbers — byte — 1 byte
short — 2 byte
int — 4 byte
long — 8 byte
decimal — float — 4 byte.
double — 8 byte.

boolean — JVM dependent.

char — 2 bytes.

PAGE No
DATE

Typecasting - converting from 1 type to another type.

implicit & explicit typecasting.

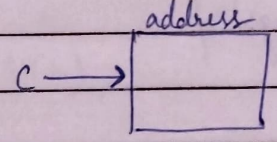
* implicit → widening conversion.
  int → long
  long → float
  float → double.

  ex- byte b=10;
  int i;
  i=b;

* explicit → narrowing conversion:
  double → float
  float → long
  long → int
  int → short
  short → byte.

  ex-
  int i=10;
  short s;
  s=(short)i;

cricketer c = new cricketer();



Wrapper Class - (get proposals for operations to be performed).

```
public static void main(String[] args){
    int i=10;
    Integer i1 = new Integer(10);
    System.out.println(i1);
    Float f = new Float(23.45);
    Boolean b = new Boolean(true);
    Character c = new Character('a');

}
```

Note: since from 1.9V wrapper class is depricated

PAGE No
DATE

```
return type methodname(input parameters){
    //method Body//
}.
```

Q- WAP to add two numbers?

```
class Add {
    int a;
    int b;
    public int add(int a, int b) {
        this.a = a;
        this.b = b;
        return a+b;
    }
}

public static void main(String[] args) {
    Add a = new Add();
    int s = a.add(5,10);
    System.out.println("Sum = " +s);
}.
```

Q- WAP to convert float into number.

```
public class typec{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        float f1 = sc.nextFloat();
        float f2 = 12.48f;
        System.out.println((int)f1+" "+(int)f2);}
    }.
```

PAGE No
DATE

* // string + any value = string.

Q. WAP to convert string type into number using wrapper class.

```
public static void main (String [] args){
    String s = "1235";
    int i = 10;
    int n = Integer.parseInt(s);
    S.O.P.(s+i);
    S.O.P.(n);
    S.O.P.(n+i);
}
```

* Constructor will return the address.

Arrays =

a -> | 10 | 20 | 30 | 40 | 50 |
0 1 2 3 4

S.O.P(a);

* Array size cannot be changed.
* Array is homogenous.

```
datatype [] arrayname;
datatype arrayname[];
int [] m;
int m[];
String [] names;
```

Array Object - arrayname = new datatype[size];
as numbers = new int[5];
names = new String[3];

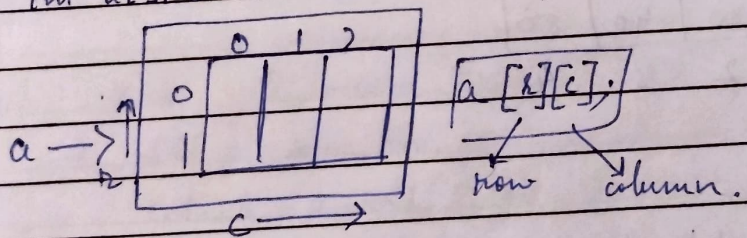
arrayname [index] = value;
numbers [0] = 10;
names [0] = "Alice";

datatype [] arrayname = {value1, value2, value3, ...};
int [] n = {10, 20, 30, 40, 50};
String names = {"Alice", "Bob", "Charlie"};

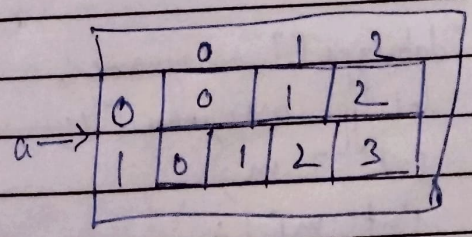
int firstnum = n[0];
String secondname = names[1];

int size = arrayname.length;

2D array.
int a[][] = new int [2][3];



Jacked Array :-
classroom = 2
classroom 1 with 3 students.
classroom 2 with 4 students.



Enter the classroom size = n;
int a[][] = new int[n][];
for(int i=0; i<n; i++) {
Enter the student size of classroom (i+1)
studentsize = m;
a[i] = new int[m];
}

int a[][] = new int[2][];
a[0] = new int[3];
a[1] = new int[4];

Q. WAP for Jagged Array.

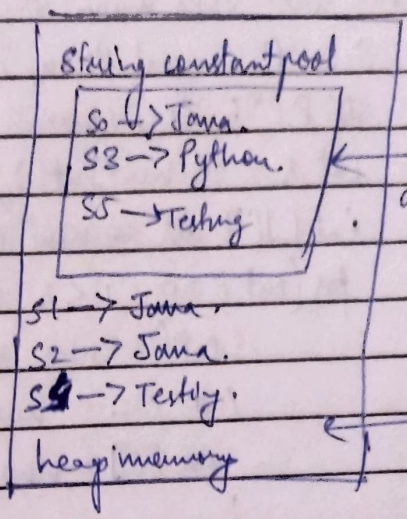
```
public class jaggedarray{
    public static void main(String[] args){
        S.O.P("Jagged Array");
        SO.P("Enter rows:");
        int r = sc.nextInt();
        int[][] arr = new int[r][];
        for(int i=0; i<r; i++){
            S.O.P("Enter column for "+(i+1)+" row:");
            int c = sc.nextInt();
            arr[i] = new int[c];
        }

        S.O.P("Enter elements in array:");
        for(int i=0; i<r; i++){
            for(int j=0; j<arr[i].length; j++){
                S.O.P("Enter row"+(i+1)+" column "+(j+1)+" element:");
                arr[i][j] = sc.nextInt();
            }
        }
        S.O.P("Jagged Array:");
        for(int i=0; i<r; i++){
            for(int j=0; j<arr[i].length; j++){
                S.O.P(arr[i][j]+" ");
            }
            S.O.println();
        }
    }
}
```

PAGE No
DATE

Strings: String is a series of characters in object form.
String are immutable.

creation.

```
String s0 = "Java";
String s1 = new String("Java");
String s2 = new String("Java");
String s3 = "Python";
String s4 = new String("Testing");
String s5 = "Testing";
```



Q. WAP to perform CRUD on strings.

```
public static void main(String[] args){
    String[] names = {"John", "Doe", "Alice", "Bob", "Jane"};
C,R {  String newname = "Charlie";
       String[] nnames = new String[names.length + 1];
       nnames[names.length] = newname;
       names = nnames;
       S.O.P("Names in array:");
       for(String name : names){
           S.O.P(name); }
U,R {  names[2] = "David";
       S.O.P("Names:");
       for(String name : names)
           S.O.P(name);
D,R {  int rem = 1;
       String[] Nname = new String[names.length-1];
       for(int i=0,j=0; i < names.length; i++){
           if(i != rem){
```

```
            uname[j++] = names[i];
        }
    }
    names = uname;
    S.O.P("Array Names:");
    for(String name : names){
        S.O.P(name);
    }
}
```

D.R.

PAGE No
DATE

OOPS -

Object Oriented Programming.

//Entity —> Domain Object.

POJO —> Plain Object Java Object.

instance variable is always 0.

Interface is a blueprint of a class.

class — a template or blueprint for creating objects.
object — a real instance of a class with unique data & behavior.
Abstraction — hide complex implementation & show essential features (abstract class)
encapsulation — wrapping of data members & methods into a single unit.
inheritance — inheriting properties of a class to subclass.
polymorphism — using methods for different ops.

1- WAP to implement OOPS?

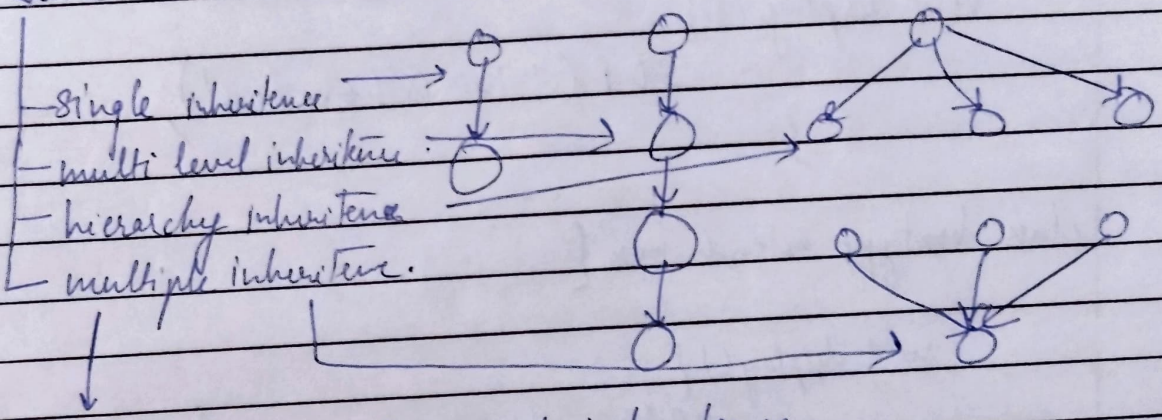
→
```
class Box{
    double l, b, h;

    Box()
    { this.l = l;
      this.b = b;
      this.h = h; }

    Box(double l, double b, double h){
        this.l = l;
        this.b = b;
        this.h = h;}
```

```
box (double s) {
    this.l = s;
    this.b = s;
    this.h = s;
}

double volume() {
        return l*b*h;
    }

void display() {
        S.O.P("This is box class");
    }

class boxtype extends box {

    void display() {
        SOP("This is boxtype class");
    }
}

public class OOPS {
    public static void main (String[] args) {
        box b1 = new box();
        box b2 = new box(10, 20, 30);
        box b3 = new (25);
        boxtype b4 = new boxtype();
        S.O.P("Volume of b1 = " + b1.volume());
        S.O.P("Volume of b2 = " + b2.volume());
        S.O.P(" b1.display(); // box class output.
            b4.display(); // boxtype class output
    }}
```

this keyword is to define the current class variable (instance).
Objects are independent.
static variables → separate memory is created.
common variable for multiple objects.

static — can be accessed without object.
static methods can access non-static through object.
non-static methods can access everything directly.

Java



class doesn't support multiple inheritance.
- extend supports only 1 class.
- ambiguity issues.

Interface is used to achieve multiple inheritance.

Method Overriding	Method Overloading
* Possible only in multiple classes.	* Possible in single & multiple classes.
* Signature of method is same but body changed.	* Signature of method is changed.
* method names are same.	* method names are same.
* belongs to inheritance.	* belongs to polymorphism.

PAGE No
DATE

final — variables, methods, class.

Variables — cannot change the value of variable.

methods — cannot override in child classes.

class — we cannot extends the class.

super -

will be used to invoke the immediate parent class variable.

will be used to invoke the immediate parent class method.

Data Abstraction -

Abstraction is a process of hiding the implementation details and showing only functionality to the users.

- Abstract Class.
- Interface

Interface.

* An interface is a blue print of a class.
* Interface contains final & static variables.
* Interface contains abstract methods (also allowed default methods & static methods from java 8 onwards).
* An abstract method is a method contains signature but not body (un-implemented method).
* Methods in interface are public
* Interface supports the functionality of multiple inheritance.
* We can define interface with interface keyword.
* A class extends another class, an interface extends another interface but a class implements an interface.
* We can create object reference for interface but we cannot instantiate interface.

PAGE No
DATE

Wrapper Class.

```
ArrayList arr = new ArrayList();
ArrayList<Integer> arr = new ArrayList<Integer>();
```

Packages & Access Modifiers.

built in packages.
user defined packages.

access modifiers. — defines scope of variables & methods.

public — directly access all variables & methods everywhere.
protected — accessible outside of package through inheritance.
default — only within the package.
private — access only within the class.

Exception Handling:-

Exception is an event which will cause program termination.

Types
- checked exception
- un-checked exception

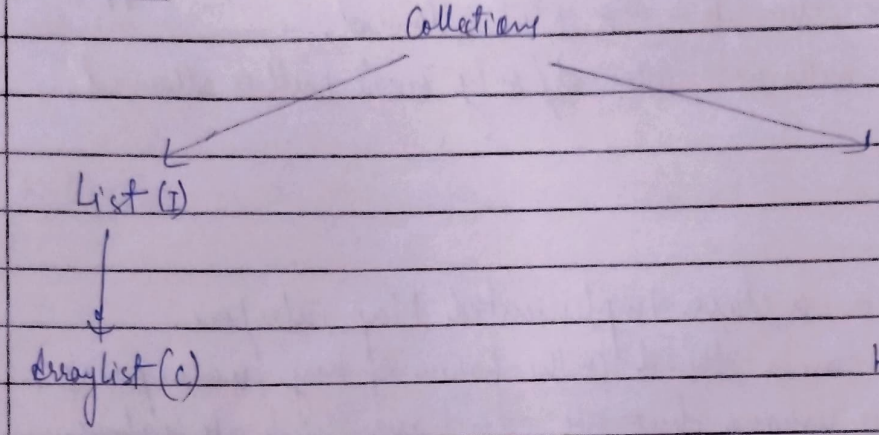
ex.
- Interrupted exception.
- File Not found exception.
- IOException

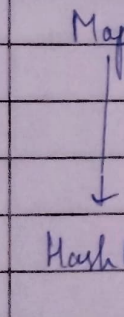
Ex.
- Arithmetic Exception
- Null Pointer exception
- Array Out Of Bound Exception

```
try { statement }
catch (ArithmeticException e) { statement }
finally { statement }.
```

Case1: Exception occured, catch block handled ---> finally block will execute.
Case2: " " , " " NOT handled ---> " " " "
Case3: Exception Does Not occured, catch block ignored ---> " " " "

Collections -





Collection - To represent group of elements/objects/data into single entity.
Collection is an interface available in java.util.

ArrayList - is class which is implemented List interface.
- Heterogenious data - allowed
- Insertion order - preserved (Index).
- Duplicate elements - allowed.
- multiple nulls - allowed.

Iterator it = mylist.iterator();

PAGE No
DATE

HashSet -
- a class implemented Set interface.

- Heterogenious data --> allowed.
- Insertion order --> Not preserved (Index not supported)
- Duplicate elements --> Not allowed.
- Multiple nulls not allowed / only single null is allowed.

HashMap
- a class implemented Map interface.
- Data can be stored in the form of key, value pairs.
- Key is unique, but we can have duplicate values.
- Insertion order not preserved (Index not followed).