JEPPIAAR INSTITUTE OF TECHNOLOGY

“Self-Belief | Self Discipline | Self Respect”

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

CS8392 – Object Oriented Programming

(Regulation 2017)

Year/Semester: II/03 CSE

2021 – 2022

Prepared by

Ms. R. Revathi

Assistant Professor/CSE

A

46×4

UNIT - I

Introdution to OOP and Java Fundamentals

Total HOURS: 09

Object Oriented Programming :-

Object Oriented Programming (OOP) is a Programming language model Organized around Objects rather than actions and data.

162

Basic Concept of OOPS (or) elements of OOPS :-

- ✓ Class
- ✓ Object
- ✓ Inheritance
- ✓ Polymorphism
- ✓ Data Abstraction
- ✓ Encapsulation.

① class :-

{

Definition :- A class is a Collection of data members (or) Variables and Methods. (or) Collection of objects is called class

}

The Primary purpose of class is to hold information.

✓ **Variables** :- It is used to store a single Value.

Syntax:-

```
datatype  Variablename = Value;
```

eg:-

```
int a = 5;
```

✓ **Methods**: - It is used to Perform a specific task.

Syntax:-

```
returntype  Method Name (arguments)
{
    // body of the Method.

}
```

eg:-

```
Void  getdata()
{

}
```

✓ **Structure of a Class**

```
Accessspecifier class  classname
{
        Variables

        Methods
}
```

eg:- - Class:

```
        access specifier   classname
Public class Student
{
     int regno;        // Variables
     char name;
   Public void getdata()   // Method
    {
      regno = 123;
      name = "Mohan";
    }
}
```

② . Object :-

Defimtion :- Instance of a class is called an object. Object means a real world entity such as Pen, Chair, table etc. Any entity that has state and behaviour is known as object.

✓ ~~An~~ Object has three charaterist

① State - represents (Variables) of an object.

② behaviour - represents the behaviour (functionality) of Methods of an object

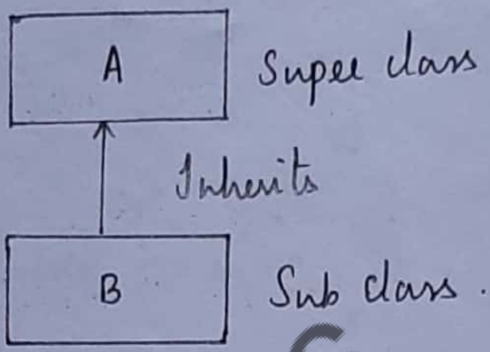
③ Identity - It is implemented (Via) Unique ID.

Syntax. Classname Objectname = new classname()

eg:- Student $\boxed{ob}$ = new Student();
objectname

③ <u>Inheritance</u>:-

Defintion:- It is a Process of deriving a Sub class from a Super class is called Single Inheritance. (or) Inheritance.



✓ When a Sub class inherits the Super class, the Sub class will have it Own behaviour/Properties and also the behaviour/Properties of the Super class.

<u>Advantage</u>:-

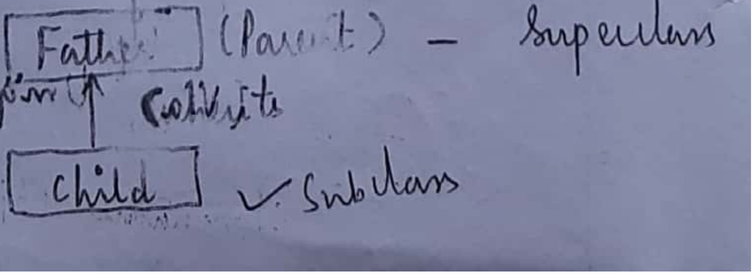
✓ Code - reusability.

✓ establishes the relationships between different classes.

<u>Types of Inheritance</u>:-

① <u>Single Inheritance</u>:-

Def:- It is a Process of deriving a Sub class from a Super class.



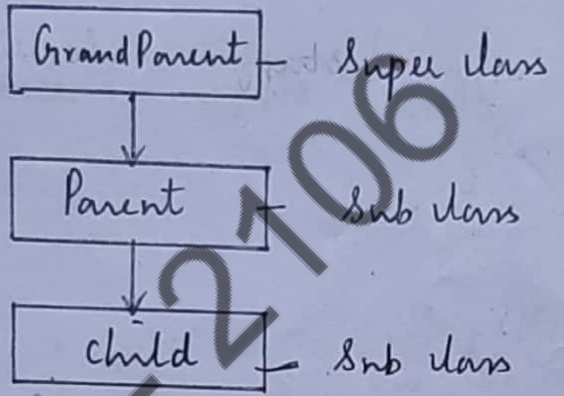
② Multiple Inheritance :-

Def:- It is a Process of deriving a Sub class from more than One Super class.

✓ In Java, Multiple Inheritance is not Supported directly. but indirectly Supported by the Concept Called "Interfaces".

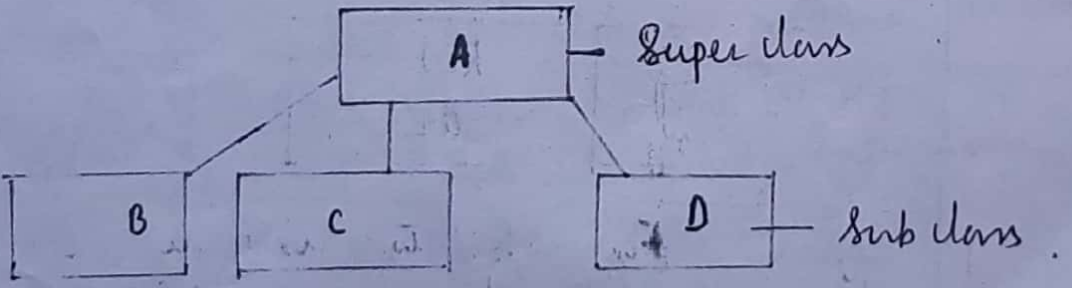
③ Multilevel Inheritance :-

Def:- It is a Process of derving a Sub class from Another Sub class.



④ Hirerchical Inheritance :

Def:- It is a Process of derving a more than One Sub class from a single super class.



Syntax: of Inheritance :-

```
Class Superclassname
{

}
```

```
Class Subclassname Entends Superclassname
{


}
```

Eg:- Inheritance :-

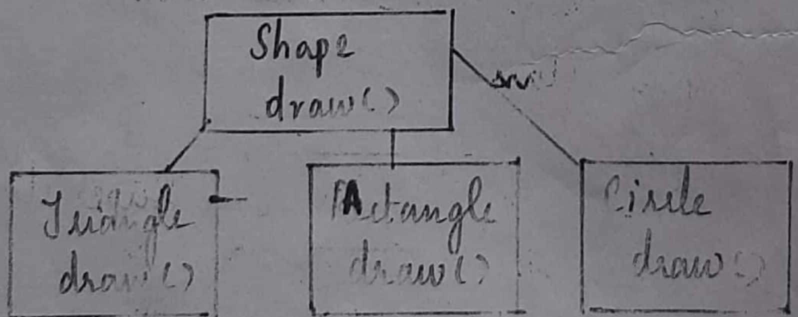
```
class student
{


}
class staff extends student
{


}
```

④ Polymorphism:-

Def:- Ability to do more than One form (or) task is called Polymerphism.



For eg:- to "Convince" the Customer differently, to draw something (eg) Shape or rectangle.

Polymorphism is classified into two types.

✓ Compile time Polymorphism:-

✓ Run-time Polymorphism

- Compile time Polymorphism (or) Method Overloading (or) Early binding.

  - Method Overloading is defined as " The method with the Same name, but differs with number of arguments or it differ with data types. Then that type of method Can be Overloaded.

  - It will happen only at Compile time.

- Run-time Polymorphism (or) Method Overridi (or) Late binding.

  - If a Sub class has the Sam method as declared in the Parent class, is Known as Method Overriding in Java.

  - More than One method which have a Same method name, Same returntype, Same no. of arguments in both the Sub class & Superclass. are Called as Method Overriding.

  - It will happen only at run time.

⑤. Data Abstraction:-

Abstraction is a Process of hiding the implementation details and showing only the functionality to the user.

eg:- Phone-call, We don't know the internal Processing of how a Phone-call Works, but Pressing of a button in a Phon will show the functionality.

✓ In Java, we use Abstract class and interfa to achieve Data Abstraction.

⑥. Data Encapsulation:-

Encapsulation is a Process of Wrapping (or) binding of Variables and methods within a class.

Difference between Procedure - Oriented and Object Oriented Programming.

Procedure - Oriented	Object Oriented
(i) In PoP, Program is divided into small [illegible] (or) Parts called functions.	(i) In OOP, Program is divided into Parts called objects.

(ii) In Pop, importance is given to functions, rather than data	(ii) In OOP, importance given to data rather than function
(iii) It follows Top-down Approach	(iii) It follows Bottom up approach
(iv) Pop does not have any access-specifier	(iv) OOP has access specifiers named Private, Public, Protected.
(v) Data can move freely from function to function.	(v) Objects can move and communicate with each other thro' functions.
(vi) less-secure, since no data hiding	(vi) More secure, because it provides data hiding
(vii) Overloading is not Possible	(vii) Overloading is Possible.
(viii) C, VB, FORTRAN.	(viii) C++, Java.

Features of Java (or) characteristics of Java:-

Simple :-

— Java is very easy to learn and it's syntax is simple, clean and easy to understand.

- Java is a simple Programming language, because

✓ Java has removed many confusing and rarely-used features. eg:- Pointers, operator-Overloading.

② Object-Oriented :-

- Java is Object-Oriented Programming language. Everything in Java is an object.

- Concepts of OOPS are

✓ Class

✓ object

✓ Inheritance

✓ Polymorphism

✓ Abstraction

✓ Encapsulation

③ Platform-Independent language :-

- Java is a Platform independent language, because it is different from other languages like C, C++, whic are compiled into Platform specific machines, while Java is a Write-Once, run anywhere language. (WORA).

- Java Code can be run on multiple Platforms (eg) Windows, Linux, Mac/OS. Java Code is Compiled by the Compiler and Converted into bytecode. The bytecode is a Platform-independen code, because it can be run on multiple platforms (ie) [WORA].

④. Secured language:-

- Java is best known for security. With Java, we can develop virus-free systems. It is secured because

✓ no explicit pointers

⑤. Robust language:-

- Robust simply means strong. Java is Robust because.

✓ uses strong memory management.

✓ lack of pointers avoids security problems.

✓ Automatic garbage collection in Java makes automatic deletion of unused objects.

✓ It has exception handling mechanism.

⑥ Architecture - Neutral:-

- Java is architecture neutral because there is no implementation dependent features. (eg) size of primitive types is fixed.

⑦. Portable:-

✓ Java is portable because it facilitates you to carry the Java byte codes to any platform. It doesn't require any type of implementation.

⑧ **Multi-threaded:-**

✓ Java is a Multi threaded Program because, It executes more than one thread Simultaneously.

- **Java Environment:-**

✓ **JRE:-** - Java Run time Environment. It is a set of software tools which are used for developing Java applications.

- It is the implementation of JVM.

✓ **JDK:-** - Java Development Kit. It is a software development environment which is used to develop Java applications and applet. It contains JRE + development tools.

- JDK Contains a Private JVM (Java Virtual Machine) and few other resources such as interpreter (java), a Compiler (javac), an archiver (jar)

✓ **JVM:-** - Java Virtual Machine is an abstract machine. It Provides an environment in which Java Byte codes Can be executed.

✓ Jvm Performs.

✓ loads code

✓ Verifies Code

✓ executes Code

Structure of a Java Program:-

```
Class Sample
{
   Public static Void main(String args[])
   {
      System.out.Println("Java World");
   }
}
```

- The above code have to be typed in Notepad and to be saved as filename.java.
- The filename of the Java Program need to be same as the classname. [Sample.java]

Compile:-

```
javac sample.java ↵
```

- It generates a .class file, which Contains byte codes.

Execute:-

```
java Sample ↵
```

- It executes .class file, which Contains byte codes.

Explanation:-

✓ Class:- It is a Keyword used to declare class in Java.

✓ Public: Access modifier, which means it is Visible to all.

✓ Static:- It is a Keyword. if we declare any method as static, it is Known as Static method. The advantage of static method is that there is no need to Create object invoke the Static method. The main metho is executed by JVM. So it does n't requi to Create object to invoke the main metho so it saves memory.

✓ Void:- It is the return type of the mett It means it does not return any Value

✓ String[] args :- – used for Command line arguments

✓ System. out. Println() – Output statemen
Class — System; object — out; Method — Println()

Java Source Program ⟶ Java Compiler (Javac) ⟶ Java Byte Code ⟶ . class file

⟶ Java Interpreter (java) ⟶ O/P

Defining Classes in Java:-

The class contains Variables and methods, and an object is an instanc of a class. A class is declared by use of "class" Keyword.

Syntax:- Keyword

```
Class classname          // class definition
  {
     datatype Variablename1;     // Variables
     datatype Variablename2;
           :
     datatype Variablename N;
return type  methodname (arguments) // Methods.
   {
     // body of the method

   }

  }
```

✓ Method:- It is used to Perform a Specific task.

✓ Variables:- Contains a Value, and if a Variable is Present inside the class is called as Instance Variables.

Example of a class:-

```
Class Box - classname
{
   double width; — Variables
   double height;
   double depth, Result;

   Void getdata() // Method,
   {
     Width = 20.0;
     height = 15.0;
     depth = 27.0;
   }
Void Volume()  // Method 2
 {
   Result = depth * height * width,
  }
 Void display()   // Method 3
 {
   System. ont. Println(" The Volume of
             the box is"+ result),
  }
}
```

Declaring objects:-

✓ First, declare a Variable of class type The Variable does not define an object. Instead, it is simply a Variable that refer to an object.

✓ Second, you must acquire an, Physical copy of the object and assign it to that Variable. This is done using the `new' operator.

✓ The new operator dynamically allocates memory for an object.

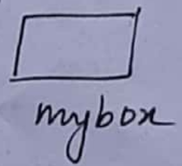
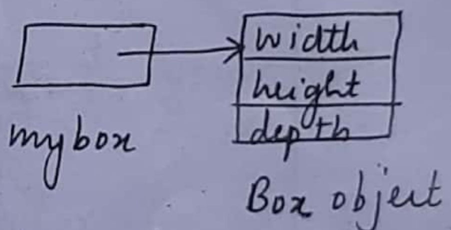
<u>Syntax</u>:-- Classname obname = new Class name()

<u>eg</u>:- Box mybox = new Box();

(or)

eg 2: Box mybox; // declare object
mybox = new Box(); // allocate a Box Object.

✓ The first ~~Example~~ line of (eg2) declares mybox as a reference to an Object of type Box. At this Point, mybox does not yet refer to an actual object. The next line allocates an object and assigns a reference to it to mybox.

Statement	Effect
Box mybox;	
mybox = new Box();	

Example - Program (Illustrates class and Object). [Democlass. Java]

```
Class Box
{
  double width;
  double height;
  double depth, result;
  Void getdata()
  {   width = 26.0;
height = 30.6;
depth = 28.7;
}
   Void  Volume()
    {
     result =  depth * height * width;
    }
   void  show()
    {
     System.out.println("The volume of the
                          box is" + result);
    }
}
Class  Democlass
 {
   Public  static  void  main(String args[])
    {
     Box  ob = new  Box();
      ob.getdata();
      ob.volume();
       ob.show();
    }
 }
```

Output:-

```
The Volume of the box is 1972.62.
```

V.A.
13m

Constructors :-

Defimtion :- It is a Method. Whenever an object is Created, Constructor will be automatically called.

Rules for Constructors :-

✓ Construtor should have the Same name as the class name.

✓ It should not have return type.

✓ It is used to initialize the objects.

✓ Constructors in Java Cannot be abstract, static (or) final.

✓ It is used to allocate memory for objects with the help of 'new' operator.

Types of Constructors :-

There are three types of Constructors They are.

① Default Constructor

② Parameterized Constructor

③ Overloaded Construtor.

Default Constructor :-

Defimtion :- A Construter having 'no' Parameter is called as default construtor.

Syntax :-

```
Class Classname
{
  Classname ( )   no arguments
```

```
 {
  // body of the Constructor.
 }

}
```

Example Program:-

```
        Class Box
        {
          double width;
          double height;
          double depth;
           Box ()    // Default Constructor
           {
              Width = 10.0;
               height = 10.0;
               depth = 10.0;

           }
          double Volume()
           {
             return width * height * depth;

           }
         }
         Class DemoDefault
          {
           Public static Void main(String
                                   args[])
            {
             Box mybox1 = new Box();
             Box mybox2 = new Box();
             double vol;
```

```
Vol = mybox1. Volume();
 System. ont. Println ("Volume is" + vol);
vol = mybox2. Volume();
 System. ont. Println (" Volume is" + vol);
 }
}
```

Output:

```
      Volume is 1000.0
      Volume is 1000.0
```

Parameterized Constructer:-

Defimtion: A Constuctor which has a more than One arguments is called Parameterized Constructor.

Syntax:-

```
        Class classname
        {
          Classname(datatype Variable1, datatype
                                  Variable2....)
           {
              // body of the Constructer.
           }
        }
```

Example Progeam:-

```
        Class Box
        {   double width;
            double height;
```

```
height = -1;
depth = -1;
}
Box(double w, double h, double d)
{
    Width = w;
    height = h;
    depth = d;
}
double Volume()
{
    return width * height * depth;

}
}
Class Demo_Overloaded
{
    Public static Void main(String args[]
    {
        Box mybox1 = new Box();
        Bon mybon2 = new Bon(10, 20, 15)
        double Vol;
        Vol = mybon1. Volume();
        System. out. Println(" Volume is" + V
        Vol = mybox2. Volume();
        System. out. Println(" Volume is" + V

    }.
}
```

Output :-

```
Volume is -1.0
Volume is 3000.0
```

8m ⊗⊗

Methods: in Java:

Defintion: Method is used to Perform a speific task.

Syntax :-

```
returntype Methodname(arguments)
{
  // body of the method.
}
```

eg:-

```
void add()
{

}
```

Example Program:- [Method returning a Value]

```
Class Box
{
   double width;
   double height;
   double depth;
  double Volume()
  {
   return Width * height * depth;

  }

}
```

```
Class DemoMethod
{
 Public static Void main (String args[])
  {
    Box mybox1 = new Box();
    Box mybox2 = new Box();
   double vol;
    mybox1. width = 10;
    mybox1. height = 20;
    mybox1. depth = 15;
     mybox2. width = 3;
     mybox2. height = 6;
     mybox2. depth = 9;
 Vol = mybox1. Volume ();
  System. out. Println (" Volume is" + vol);
   Vol = mybox 2. Volume ();
  System. Out . Println ( " Volume is" + Vol);
  }
}
```

Output:

```
 Volume is 3000.0
 Volume is 162.0
```

Example Program: [Adding a Method that takes Parameters]

```
      Class Box
      {
```

```
double width;
double height;
double depth;
  Void setDim (double w, double h, double d)
   {
     width = w;
     height = h;
     depth = d;
    }
double Volume ()
   {
     return width * height * depth;
   }
 }
  Class Add Parameters
   {
     Public static Void main (String args[])
      {
        Box mybox1 = new Box ();
        double Vol;
        mybox1. setDim ( 10, 20, 15);
        Vol = mybox1. Volume ();
        System. out. Println (" Volume is" +
                                      Vol);
      }
   }
```

Output:

```
Volume is 3000.0
```

This Keyword:-

Defintion→ This keyword is used to refer to the Object that invoked it. this Can be used inside any method, to refer to the Current object.

Syntax:(or) Example:-

```
Class Box
  {
     double width,
     double height,
     double depth;
     Box( double width, double height,
                        double depth)
       {
   this . Width = width,
   this. height = height,
   this. depth = depth,
       }
  }
```

Method Over-loading:-

Defintion:- When two or more method within the Same class have the Same name, but differs with no. of argumen Or it differs with datatypes, then that type of method Can be Overloa

Program:-

```
Class OverloadDemo
 {
    Void test ()
    {
      S.O.P("No Parameters");
    }
    Void test (int a)
    {
      S.O.P("a:" +a);
    }
    Void test (int a, int b)
    {
      S.O.P("a and b"+a+ " "+ b);
    }
 }
 Class Overload
  {
 Public static Void main(String args[])
  {
   OverloadDemo ob = new OverloadDemo()
   ob. test();
   ob. test (5);
   ob. test (5,6);
  }
 }
```

Output:-

```
No Parameters
a : 5
a and b : 5  6
```

Method - Overriding:-

Defimtion:- When a method in a Sub class has the Same name and Same no. of arguments, Same return type as a method in it's Super-class, then the method in the Sub class is Said to Override the method in the super class.

Example Progeam:-

```
class A
{  int i, j;
   A (int a, int b)
   {
     i = a;
     j = b;
   }
   Void show()
   {
    S.o.p ("i and j "+ i + " "+ j);
   }
}
 class B extends A
 {  int K;
    B (int a, int b, int c)
    {
      Super (a, b);  // this calls Super
      K = C;            class Constructor
    }
    Void show ()
    {
```

```
System.out. Println("K"+ K);
  }
}
Class override
 { Public static Void main(String args[])
   {
     B subob = new B(1,2,3);
     subob. show();   // this calls show()
                         in B.
   }
 }
Output:-
 K = 3
```

Access - Protection (or) Access - specifiers in Java:-

Defintion: The "access modifiers" in Java specifiers (scope) of a Variable, method, Constructor of a class.

✓ There are 4 types of access modifiers.

① Private
② default
③ Public
④ Protected.

① Private:- If any Variable (or) a method is declared as Private, then that

Variable and a method Can be accessible only within the class.

Program:-

```
Class A
 {
  Private int data = 40;
  Private void msg()
  {
   S.o.p(" hello java");
  }
 }
 Public Class Sample
 {
Public static void main (String args[])
  {
   A ob = new A();
   S.o.p(ob.data); // error
   ob.msg(); //error, since msg()
                method is Private, it
               Can be accessible
                 only within Class
  }
 }
```

② Default:-

If you don't use any access modifier, it is treated as default. The default modifier is accessible only within Package.

③ Public :- If any Variable or method of a class is declared as Public, it can be accessible from Outside the class.

Program :-

```
Class A
{
   Public int data = 40 ;
   Public Void msg ()
   {
     S.o.p (" hello java");
   }
}
Public class Sample
{
Public static Void main (String args [])
{
   A ob = new A();
   S.o.p (ob. data);
   ob. msg ();
 }
}
```

Output :-

```
40
hello java.
```

④ Protected :- - If any Variable or method is declared as Protected, only the inherited class can access it's Variable and methods. It is used only in Inheritanc

UQ
8m
⊗⊗

Static Members:-

Static is a non-access modifier in Java which is applicable for

- ✓ Variables
- ✓ Methods
- ✓ blocks

Static as Variables:- (characteristics)

✓ When a Variable is declared as static, then a single copy of Variable is Created and shared among all objects.

✓ We can Create static Variables at class-level only.

✓ Automatically Variable is initialized to Zero, One Object of that class is Created.

Syntax:-

static datatype Variablename;

Program:- [Static Variables, methods + block

```
Class UseStatic
{
    Static int a = 3; // Static Variab
    Static int b;
    Static Void math(int x) // stat
                               method
    {
```

```
S.o.p("x="+x);
S.o.p("a="+a);
S.o.p("b="+b);
}
Static        // Static block
 {
   S.o.p("Static block intialized");
   b=a*4;
 }
}
Class Sample
 {
   Public static void main(String args[])
     {
        math(42);
     }
 }
```

Output:-

```
Static block intialized
x = 42
a = 3
b = 12.
```

Static as Methods (characteristics)

✓ When a method is declared with static Keyword, it is Known as static method.

Rules (or) characteristics:-

① They can only directly call other static methods.

②. They can only directly access static Variables.

③ They cannot refer to this (or) Super Keyword.

Syntax:-

```
static returntype Methodname()
{

}
```

Static blocks:-

It gets executed exactly once, when the class is first loaded.

Syntax:-

```
Static
{

}
```

Data-types in Java:-

Definition:- Data types specify the different Sizes and Values that can be stored in the Variable.

Types of data types in Java:-

1. Primitive data type - includes integer

✓ Non-Primitive data type. – includes classes, Interfaces and Arrays.

Java defines 8 Primitive data types of data: byte, short, int, long, char, float, double and boolean. These can be Put in four groups.

① Integers:- includes byte, Short, int and long. Which are Whole-signed numbers.

②. Floating-Point numbers –: includes float and double.

③ Characters – includes char

④ boolean – includes boolean, representing true/false values.

Example Program:-

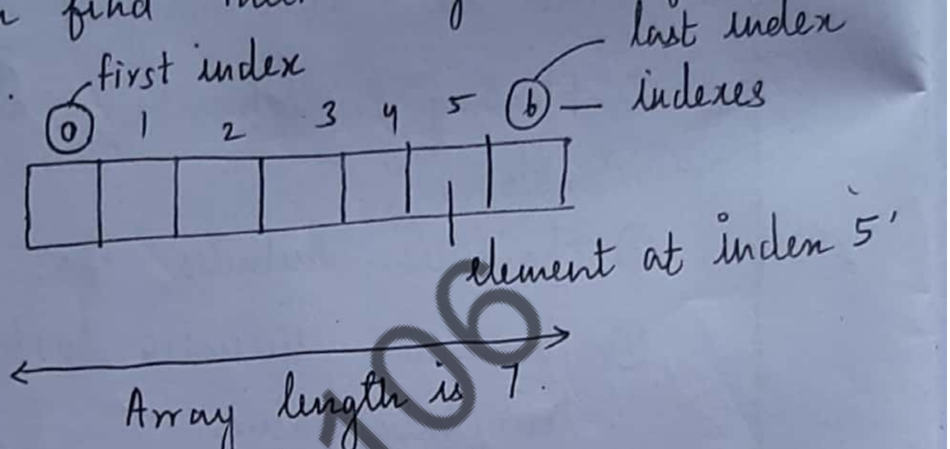
```
        class light
        {
          Public static Void main(String args[])
          {
data     ⎧ int lightspeed;
types    ⎨ long days;
         ⎩ long seconds;
           lightspeed = 186000;
             days = 1000;
           Seconds = days * 24 * 60 * 60;
           S.O.P("lightspeed is"+ lightspeed);
           S.O.P("days is"+ days);
          }
        }
```

UQ
13m.

Arrays:-

Defintion:- Array is a collection of similiar (or) Same datatype which is stored under a Common name.

- In Java, all arrays are dynamical allocated. Sime arrays are objects in Java, we can find their length using "length" Property.



Advantage of Array:-

✓ Code optimization:

✓ Random-access.

Disadvantage:-

✓ Size limit:- We can store only fixed size of elements in the array. It does n't grow it's Size at run time. To solve this Problem, collection framework is used.

Types of Array:-

✓ One-dimensional Array

✓ Two-dimensional Array.

One-dimensional Array:-

Syntax:-

```
datatype Variablename [] ;
Variablename = new datatype [Size] ;
```

eg:-

```
int arr[];
arr = new int [5] ;
```

Program:- [linear-search] :-

```
Class linear
 {
  Public static void main(String args[])
  {
     int arr[], Key = 7;
     arr = new int [5];
     arr[0] = 6;
     arr[1] = 7;
     arr[2] = 8 ;
     arr[3] = 9 ;
     arr[4] = 5 ;
     for(int i=0; i<=(arr.length-1); i++)
     {
         if (arr[i] = = Key)
         {
             return i;
         }
     }                     o/p:- 1
  }
}
```

Two-dimensional Array:-

Syntax:-

datatype Varname[][] = new datatype
[row size] [column siz

eg:-

int a[][]= new int [3][3];

Program:- [Matrix Multiplication]

```
Class matrixmul
{
 Public static void main(String args[]
 {
   int a[][]= new int [3][3];
   int b[][]= new int[3][3];
   int c[][] ,k;i,j;

 for(int i= 0; i<3; i++)
 {
  for(int j =0; j<3; j++)
   {
     a[i][j] = i;
   }
 }
 for(i=0; i<3; i++)
  {
   for(j=0; j<3; j++)
    {
      b[i][j]= j;
    }
  }
for( i=0; i<3; i++)
{
  for (j=0; j<3;  j++)
   {
     for(k=0; k<3, k++)
      {
       c[i][j] = c[i][j] + a[i][k] * b[k][j];
      }
   }
}
for (i=0; i<3; i++)
 {
  for(j=0; j<3; j++)
   {
     S.o.p(" The resultant matrix" + c[i][j]);
    }
  }
}
}
```

VQ

13m ⊗ ⊗ ⊗

Packages in Java :-

(2m) Defintion:- Packages is a collection of classes, Interface and sub Packages.

Types of Packages:-

- ✓ Pre-defined Package (or) built in Packages.
- ✓ User-defined Package.

(2m) Benefits (or) Advantages of Packages :-

✓ Java Packages removes name collisions

✓ Java Packages Provides access Protections

✓ Used to Categorize classes and interfaces so that they can be easily maintained.

<u>User-defined Package:-</u>

Syntax:-

```
    Package Packagename;
```

eg:-

```
    Package Pack;
```

Example Program:-

```
 Package Pack; ← userdefined Package
 class Balance
  {
     String name;
     double bal;
    Balame(String n, double b)
     {
      name = h;
      bal = b;
     }
     Void show()
      {
       if(bal < 0)
        { S.O.P("name="+ name);
        }
      }
  }
   Class AcountBalame
```

```
Public static Void main(string args[])
{
   Balame c[] = new Balame[3],
    c[0] = new Balame("Mohan",-123.23),
    c[1] = new Balame("Ram", 157.02),
    c[2] = new Balame("Kumar", +12.23),
    for(int i=0, i<3, i++)
     {
        c[i]. show(),              o/p:
                                     name = Mohan
     }
  }
}
```

Call this abone file(or) Program as AccountBalame.java and Put in a directory called "Pack". Next Compile the file. Make Sure that the resutting .class file is also in the "Pack" directory.

Then try java Pack.AccountBalame ✓

Remember you will need to be in the directory above "Pack", When you execute the abone Command.

Pre-defined Package:-

Syntax:-

```
import java. Packagename.*;
```

eg:

```
import java. util.*;
```

name of the Packag

all the classes + interfaces of that Package will be imported.

Program:- [Utitity Package]:-

```
[import java. util.*;] Predefined
                        Package
class PredefinedPack
{
  Public static void main('String args[]
  {
    int a, b, C;
    Scanner ob= new Scanner (System. in)
    a = ob. nextInt();
    b= ob. nextInt();
    C=a+b;
    S.o.P("Addition is"+C);
  }
}
```

O/p

```
Addition is 30.
```