*Introduction to OOP:

→ OOP is ~~i~~

→ Object oriented approach is invented and the major motivating factor is to remove some flaws in the procedural approach.

→ OOP treats data as a ~~ele~~ critical element in pgm development & doesn't allow it to flow freely around the system.

→ OOP allows decomposition of prblm into no. of entities called objects & then builds data & function around these objects.

→ data of object can be accessed ~~by~~ by funcn associated with that obj, the function of obj can be accessed by func of other object.

*Principles:

1) objects
2) classes
3) Data abstraction & encapsulation.
4) Inheritance
5) polymorphism
6) Dynamic binding
7) Message passing.

Objects:

- These are the basic runtime entities of Object oriented system.
- They may represent a person, place, thing---
- that the program has to handle and also represent user-defined data.

- Prgm objects should be chosen that they should match closely to the real-world objs.
- Obj takes up space in memory & have an associated address like a record in pascal.

Classes

~~→ As mentioned above obj conts~~

→ The entire set of data & code of an obj can be made as a user-defined datatype with the help of class.

→ Objs are variables of the type class.

→ Once class is defined, we can create any no. of objects belongs to that class.

→ A class is collection of objects similar types.

Data Abstraction & encapsulation.

→ Hiding the unneccesary details to the user is abstraction.

→ classes use concept & are defined as a list of abstract attributes such as size, cost ---

→ Encapsulation is binding of methods & variables under a common class name.

polymorphism

→ It means many forms.

→ It is quality that allows one interface to access a general class of Actions.

eg: wheel steering

Inheritance

→ It is the process by which 1 obj can acquire the properties of another obj.

→ It supports the concept of hierarchical classification

→ It has 2 classes parent class & child class.

→ parent class can be accessed by child class by using "extends".

Dynamic Binding

→ Binding refers to the linking of procedural call to code to be executed.

→ It is associated with polymorphism & inheritance

Message passing

→ OOP consists a set of objs that communicate with each other by sending & receiving information

→ A message for an object is req for execution of a procedure.

*Procedural language Vs Object oriented.

POP	OOP
1) Prgm is divided into small parts called funcns.	Pgrm is divided into small parts called objs.
2) It follows topdown approach.	It follows bottom down approach.
3) There is no access specifier.	public, private, protected & default are access specifiers.
4) Overloading is not possible	Overloading is possible.
5) It is based on unreal world.	It is based on real world.
6) Eg: C, pascal .... etc	Eg: Java, python — etc.

---

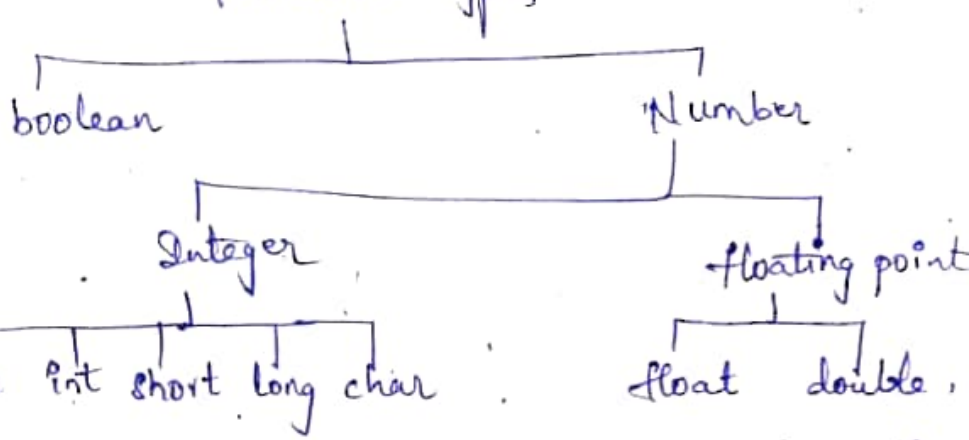
*Operators:

+ These are used to perform operations. Eg: +, -, /, *.

Operator type	Category	precedence
Unary	postfix	exp++, exp--
	prefix	++exp, --exp
Arithmetic	multiplicative	*, /, %
	Additive	+ -
shift	shift	<< >> >>>
Relational	comparison	<= >= <>
	equality	== !=
logical	AND	&&
	OR	||
Ternary	ternary	?
Assignment	Assignment	= += -= *=

Bitwise
- AND &
- inclusive OR |
- exclusive OR ^

* Primitive Types :



→ Integers & floating point no's are totally diff
representation, storage & operations are " "

Prgrm:
```
import java.util.Scanner;
public class DataTypes
{
    public static void main(String[] args)
    {
        Scanner sc= new Scanner(System.in)
        System.out.println("Int value : ");
        int x= sc.nextInt();
        System.out.println("byte value : ");
        byte b = sc.nextByte();
        System.out.println("byte default value : "
                                    +b);
        System.out.println("Int default value : "+i);
    }
}
```

* History & evolution of java: (James Gosling).

→ Java is designed for interactive television as it was too advanced technology at that time.

→ History of java starts with Green team.
Greenteam - Java team members.

→ Java is used in Internet programming, mobile devices, games ... etc.

1) James Gosling, Mike Sheridan & patrick Naughton initiated Java language project in June 1991.

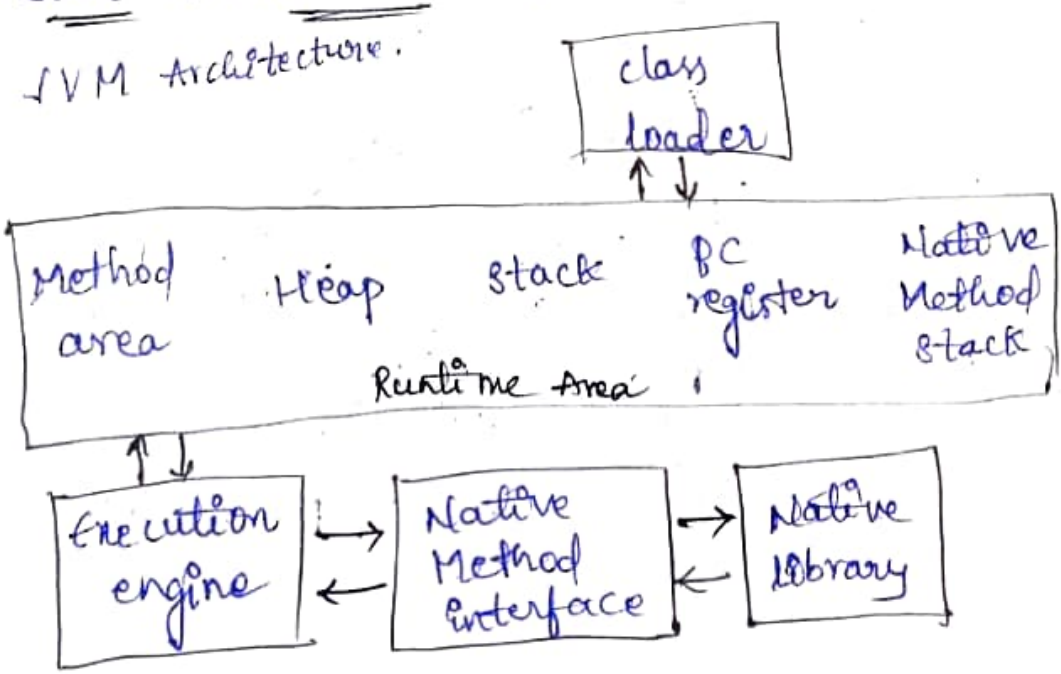
2) Initially designed for small, later used in electronic appliances like set-top boxes.

3) Firstly, it was called as "Greentalk" and later "Oak" was developed as part of Green project.

Many java versions have been released till now, The current stable release of Java is Java SE 10.

* Java Virtual Machine:

JVM Architecture.



class loader : It loads the class for execution.

method area : stores pre-class structure as constant pool.

Heap : It is used for allocation of objects

Stack : It is used to store the variables.

PC register : It holds the address of JVM instruction currently being executed.

Native method stack : It is used to store machine code.

Executive engine : It controls the execute of instructions in methods of classes.

Native method interface : It gives interface b/w native code & java code during execution.

Native library : It contains files required for execution of native code.

→ JVM provides runtime environment to execute java byte code.

→ we compile *.java files to obtain *.class files that contains bytecodes.

→ JVM control execution of every javapgm.

→ JVM enables automated exception handling.

* Program Structure :
1) Documentation section
2) Package statement
3) import statements
4) Interface statement
5) class Definition
6) main method class.
7) Main method definition

1) Documentation section :
We can write comments in this section. These are useful to prgrmmers to understand the code. These are optional but we use them for understanding purpose.

2) Package Statement :
A package can be created with any name. A package is a group of classes that are defined by a name. In package we can declare many classes with one element.
→ Package is a keyword that tells that it has been created. where it is also optional.

Declaration : package package_name ;

3) Import statements : If you want to use the classes of another package, we can simply import it to program the particular package within the class By using import keyword.

4) Interface statements : These are like a class that contains group of methods declarations.
→ It is also an optional section.
→ It can also used to implement multiple inheritance

within a program.

5) Class Definition: A java prgm may contain several class definitions. classes are main and essential elements of any java pgrm.

6) Main method class: Every java stand-along prgm requires main method as the starting point of pgrm
→ This is an essential part of a java prgm.
→ There are many classes in a prgm but only one class defines the main method.

Program:

```
// Java Example Program. → Documentation section
class Demo                // → class definition
{
// main method
    public static void main(String[] args)
    {
        System.out.println("Jyothi");
    }
}
```

Explain java buzzwords with relevant programs

The features of java are also known as java buzzwords. They are:

10

1) Simple
2) Object - Oriented
3) Portable
4) platform independent
5) Secured
6) Robust
7) Architecture neutral
8) Interpreted and compiled
9) High Performance
10) Multithreaded
11) Distributed
12) Dynamic.

* Simple:

Java is very easy to learn, and its syntax is simple, clean and easy to understand.

Example:

```
public class Demo
{
    public static void main(String[] args)
    {
        System.out.println("Sasi");
    }
}
```

* Object - oriented:

Java is an object-oriented programming language. Everything in Java is an object. OOP is a methodology that simplifies software development & maintenance by providing some rules.

Basic concepts of OOPs are :

1) object 2) class 3) Inheritance
4) polymorphism 5) Abstraction 6) Encapsulation

Example :

```
class Parent
{
    void add (int x, int y)
    {
        System.out.println(x+y);
    }
}
class Child extends Parent
{
    void add (int x, int y)
    {
        System.out.println (x+y);
    }
}
public class DemoInheritance
{
    public static void main (String[ ] args)
    {
        Child c = new Child ();
        c.add (5,10); Parent p = new Parent ();
        p.add (10,20);
    }
}
```

* platform - Independent :

Java can be executed on multiple platforms. Java code is compiled by its converted into byte code.

Example:

```
public class File
{
    public static void main(String[] args)
    {
        System.out.println("can be executed");
    }
}
```

* **Secured:**

Java is best known for its security. It is because

- → No explicit pointer
- → Programs run inside a virtual machine sandbox.
- → Class loader.
- → Byte code verifier.
- → Security Manager.

Java provides these securities by default.

* **Robust:** Robust means strong.

Java is robust because

- → It uses strong memory management.
- → There is a lack of pointers that avoids security problems.
- → Java provides automatic garbage collection, which runs on JVM to get rid of objects.
- → There are exception handling & type checking mechanism.

Example:

```
class Handling
{
    public static void main(String[] args)
    {
        try
        {
            int divideByZero = 5/0;
            System.out.println("error occurred");
        }
        catch(ArithmeticException e)
        {
            System.out.println("exception: " +
                                e.getMessage());
        }
    }
}
```

* Architecture-neutral:

Java is architecture-neutral because there is no implementation dependent features.

Example: Here the size of primitive types is fixed. It occupies 4 bytes of memory for both 32 & 64-bit architectures in Java.

~~Portable~~ prgm: portable etc.

```
import java.util.Scanner;
class PrimitiveTypes
{
    public static void main(String[] a
    {
        Scanner sc = new Scanner(Syste
        System.out.println("byte value:
```

```
System.out.println("int value : ");
int i=sc.nextInt();
System.out.println("byte default value : "+b);
System.out.println("int default value : "+i);
}
}
```

* Portable :

Java is portable because it facilitates to carry the Java bytecode to any platform. It doesn't require any implementation.

* Interpreted and compiled :

The Java source code first compiled into a binary byte code using Java compiler, then this bytecode runs on JVM, which is a software based interpreter.

* High-Performance :

Java is faster than other traditional interpreted programming languages because its byte code is "close" to native code.

* Multi-threaded :

Java pgms that deal with many tasks at once can defined by using multiple threads. The main advantage is it doesn't occupy memory for each thread, it shares a common memory area.

Some methods are

→ start() - Initiates the execution of a thread.
→ run() - triggers an action for the thread
→ getName()
→ sleep() - used to suspend thread temporarily.
→ yield() - used to send current executing thread to standby mode.

Example:

```
class MultiThread extends Thread
{
    public void run()
    {
        try {
            System.out.println("Thread is
                running");
        }
        catch(Exception e) {
            System.out.println("Exception
                Caught");
        }
    }
}
class DemoThreading
{
    public static void main(String[] args)
    {
        int n=3;
        for(int i=0;i<n;i++) {
            MultiThread mt=new
                MultiThread();
            mt.start();
        }
    }
}
```

Arrays

→ It is a collection of homogeneous elements of same datatype in a contiguous memory.

Syntax : datatype ArrayName [size];

→ Instantation means creating a memory for array object.

Instantation : int a[ ] = new int [10];

→ Initialization : int a[ ] = {1, 2, 3, 4}; .

NOTE : New is also known as operator which is used to create a memory for objects.

* Difference b/w String & String Buffer :

String	String Buffer
It is immutable	It is mutable.
It is slow and consumes more memory when we concatenate too many strings because every time it creates new instance	It is fast and consumes less memory.
It uses string constant pool.	It uses heap memory
It overrides the equals ( ) method of object class.	It doesnot override the equals ( ) method