Advanced Programming

Tutorial-1

1. Off-blw languages
   Website of Indian Railway

Lecture-1

Introduction

# JAVA

* Programming lang.
  * 1999 (Oak)
  * 1995 (JAVA) James Gosling
  * Platform Independent (Byte Codes)
  * OOPS (Concept Used)
  * Don't have pointers in JAVA because complexity less,

* Security
  * Secured Ring
  * It is simple
  * Concepts based on real life problems

* Types of JAVA application:
  * We can design basically 4 applications in JAVA

  * Stand Alone Applications (Desktop) e.g. Media Player
  * Web Applications e.g. Indian Railway
  * Enterprise Application e.g. MQM
  * Mobile Applications e.g. Android
Standalone
There are also known as desktop applications or window based application i.e. an application we need to install on every machine such as antivirus, media players etc. User-end Swing are used in Java for creating standalone applications.

Web
An application that runs on the server site & creates dynamic web pages is called as Web app. Servlets, JSP, struts technology are used in Java.

Enterprise
An application i.e. distributed in nature such as banking app etc. In Java EJB (Enterprise Java Bean) is used for creating enterprise application.

Mobile
An application i.e. created for mobile devices currently Android & JAVA J2E are used to creating mobile app.

What is JAVA?
JAVA is a general object oriented programming language & a computing platform developed by "James Gosling" of Sun micro system in 1995.
Why Java?
Java is secure
It is platform independent
Java is portable

Java platform

JRE  JDK
Java Run Time  Java Development
environment  Kit (JRE + Development tools)

JVM: It is an abstract machine, it is specification that provides run time environment in which JAVA byte code can be executed.

JVM's are available almost for many hardware & SW platform i.e. Java is platform independent.
Main features of Java:

1. Simple: Java is simple because most of the concepts have been taken from C++, it is very easy to learn because it does not use any header file. It eliminated the use of pointers, operator overloading & virtual base classes eliminated.

2. Object Oriented: Java is a pure object-oriented programming language. Everything in Java is an object, all program data resides in objects & classes.

3. Distributed: Java has network facilities, it enables multiple programmers at remote locations to work together on a single project.

4. Robust: Java virtually eliminates the problem of memory deallocation by using garbage collection for unused object. Moreover, run-time errors are managed by exception.
Handling. Therefore, Java is robust for program failures i.e. memory mgmt. mistakes & mishandled exceptional conditions.

5. Platform Independent & Portable: Most significant contribution of Java over other lang. is its portability. Java program can be easily moved from one computer to another anywhere anytime.

This is the reason why Java has become a very popular lang. for programming on internet which interconnects all kinds of system worldwide.

6. Secure: Since Java is used on internet, security is an imp issue. Absence of pointers ensures that programs cannot gain access to memory locations.

7. Compile & Interpreted: Generally comp. lang. are either compiled or interpreted but Java combines both compile & interpreter.

8. Multithreading: Java was design to meet the real world environments of creating interactive network programs to accomplish this, Java supports multithreaded programming which allows u to write programs that do do many things simultaneously.
Reusability: is an aspect of OOP paradigm. Java supports this concept i.e. Java classes can be reused in several ways. It is always nice if we could use something that already exist rather than creating the same thing all over again.

The inheritance allows sub class to inherit all the variables & methods of their parent class. Inheritance may take different forms:
1) Single inheritance (only one super class)
2) Multilevel inheritance (derived from derived class)
3) Multiple inheritance (several super classes)
4) Hierarchical (one super class & many sub classes)

There is no multiple inheritance in the Java but we can implement multiple inheritance through interfaces.

Polymorphism: It is a Greek word poly + morphism i.e. same interface acting differently with different inputs.

Polymorphism Ex -

Dog - Nose

Input: food → Output: eat
cat → Barking

It is a mechanism by which same interface is used for general class of action but depending upon a different inputs different outputs
5. **Encapsulation:**
   
   It is a mechanism by which data members i.e., member function & variables are enclosed into a single entity called class to protect from outside interference. Would you any interference?

   *Example.* Mobile phone having all features combine in one, class having students combine in one become CSE.

6. **Data Abstraction & Data Hiding**

   1. **In Data Abstraction**
      
      It is all about hiding complexity, providing security to data.

   2. **In Data Hiding**
      
      It means no need to show how complicated steps you have i.e., just hiding the person to do a particular operation, not exposed.

   It's a philosophical concept i.e., almost everything a good
developer writes in abstraction
Ex: Just to hide the complexity as such & in

O*A
Ex: Working of an engine

Data hiding is one of the main goals of object-oriented programming, which is achieved by hiding private attributes and methods from the outside world. This helps in keeping the data safe as it may affect the other data.

Ex: O.H. passwords, college data etc. It is available to authorized members not to everyone.

Of blw C++ & JAVA

C++
1. C++ is basically C with extended Object-Oriented extension.

2. It implements the concepts of multiple inheritance.

3. In C++, we use pointers.

JAVA
1. Java is purely OOP Lang.

2. Java does not support multiple inheritance of classes.

3. There is no use of pointers.
4. In C++ we have destructor.
4. Java replaced destructor with the `finalize()` method.

5. In C++ we use header files.
5. There is no use of header files in Java.

6. There is operator overloading in C++.
6. There is no operator overloading in Java.

7. In C++ we use global variable.
7. In Java there is no use of global variables.

8. In C++ we have template.
8. Java does not have template classes as in C++.

---

In Java:

- Data Types
  - Primitive
    - Defined by language
  - Non-primitive
    - Defined by the user

- Non-numeric
  - boolean
  - integer
    - Byte (8B)
    - Short (16B)
    - Int (32B)
    - Long (64B)
  - Float (4B)
  - Double (8B)
  - Character (16B)
  - Non-numeric
    - String
    - Interface
Type conversion

* In some cases, it might want to assign a value of one data type to a variable of another type.
  * If both source and destination types are compatible, then Java performs a conversion.

* Java automatic conversion

Java automatically converts one type to another only when one of the following 2 conditions is satisfied:
1. Both types are compatible with each other.
2. Size of the destination type is more than the source type.

When one of the above two conditions is satisfied, then Java performs "implicit conversion." It is also known as "widening conversion."

* Type casting "Narrowing" (Big -> Short)

If we want to convert two types which are incompatible, size of destination type is less than the size of the source type, then a conversion is done "explicitly." This process is known as type casting. Ex: If we want to convert integer value through byte value, Java cannot do this automatically as a size of int is.

```
Double -> float -> int -> long -> Byte -> Int i;

Byte = (destination type) / float i;
```
\[ a = 15 \quad b = 5 \quad c = 10 \quad \text{? is equivalent to if else statement} \]

\[ \text{if } (a > b) \]

\[ A \]

\[ \text{else} \]

\[ B \]

\[ (a > b) \rightarrow a : b \]

\[ (a > b ? (a > c ? a : c) : (b > c ? b : c)) \]

\[ \text{int num} = 5 \]

\[ \text{Integer num} = \text{new Integer}(5); \]

\[ \text{Float instance of class} \]

\[ \text{Double} \]

\* Object

It is a thing through which we can interact. We can send messages to objects. It is a physical entity. Every object has its own state, behavior, and identity.

\* State

\* Behaviour

\* Value

\* Functionality

\* Identity

\* Reference

\* What object has

\* What object can perform

\* It is defined by the value that variable contains

\* It is defined by the face of class

\* We can identify an object by its name
* class
  It is a user defined data type which is a collection of objects.
  It contains member variables & member func.
  Values are assign to objects & to variables. It acts as a template for objects.

Types of Variables in JAVA

3 types of Variables in JAVA
1. local
2. Instance
3. Static

* Variables that will be declare inside any func. that will be known as local variables.

* Variables declare outside any func. that will be known as Instance variables.

* Variables declare outside any func. with a keyword static is known as static variables.

```java
class Cse
{
    public static void main(String arg[])
    {
        int num1 = 5, num2 = 10, sum = 0
        sum = num1 + num2
        System.out.println("Sum is" + sum);
    }
}
```
Example

```java
public class Rectangle {
    private int length, breadth;

    public Rectangle() {
        length = 10;
        breadth = 20;
    }

    public void area() {
        int area = length * breadth;
        System.out.println("Area: "+area);
    }
}
```

class RectangleMain {
    public static void main(String[] args) {
        Rectangle rect = new Rectangle();
        rect.area();
    }
}
Rectangle obj = new Rectangle();
obj Area();

How to create a Simple class:

class Area
{
int length, breadth; int area;
void area();

length = 10;
breadth = 20;
area = length * breadth;
}

Class Area Main
{
psvm (string arg[1])
Area obj = new Area(); // object created.
obj. length = 10;
obj. breadth = 20;
int area = obj. length * obj. length * breadth;
obj. area;
so. pln ("Area is " + obj. area);

How to create Constructor

class Area
{
int length, breadth, int area;
Area()

length = 10;
}
breadth = 203

class Area Main
{
    psym();
    Area obj = new Area();
    How to pass parameters in the func.
    class Area:
        int length, breadth, int area;
        void area (int l, int b);
        
        length = l;
        breadth = b;
        
    class Area Main
    {
        psym();
        Area obj = new Area();
        obj area (10, 20);
        
        This keyword is used when any ambiguity
        is exist btw the local & instance
        variable.

        class Area:
            int length, breadth, int area;
            void area (int length, int breadth);
            
            this. length = length;
class Area Main

```java
public class Area {
    private int length, breadth;

    public void area(int l, int b) {
        length = l;
        breadth = b;
        System.out.println("Length: "+length + " Breadth:"+breadth);
    }
}
```

```java
Area obj = new Area();
obj.area(10, 20);
obj.area(10, 20);
```

**This Keyword**

It is a special keyword in JAVA which is used to refer to the current instance variable of any particular class.

If there is any ambiguity between the instance variable & the parameters pass, this keyword is used to resolve the ambiguity.

**Method Overloading**

Same func. name but different parameters.

```java
int length, breadth, int area;

void area(int l, int b) {
    length = l;
    breadth = b;
    System.out.println("Length: "+length + " Breadth:"+breadth);
}
```
area = length * breadth
S.O. pln ("Rectangle") area

```java
3. int5
void area (x) 11 square

area = length * length;
```

```java
Ex  class Area Main
{
    public /
    Area obj = new Area(12, 13, 14);
    obj.area (x) 11 Rectangle
    obj.area (x) 11 Square.
}
```

```java
Ex  class Employee
{
    int id;
    String name, address;
    double salary;

    Employee (int i, String n, String a, double s)
    {
        id = i;
        name = n;
        address = a;
        salary = s;
    }
}```
```java
void display()

class Employee

class Employee Main

sum

Employee obj1 = new Employee(101, ("Rayal", "+123", 50000));
Employee obj2 = new Employee(110, ("Ria", "+33", 25000));

obj1.display();
```

9. Write a program to calculate factorial of the no. using recursion.

10. Fibonacci series.

* Inheritance

ex class parent

```java
int num1 = 10;
```

ex class child extends parent

```java
int num2
num2 = num1 + 10;
```
Single

Parent 1  Class 1  Parent 2

Child 1  Class 2  Child 1

Class 3  Child 2

Multiple

Parent 1  Parent 2

Child 1  Child 2