

Introduction to Object Oriented Programming and Methodology

By

SMITA SANKHE

Assistant Professor

Department of Computer Engineering

K. J. Somaiya College of Engineering

Somaiya Vidyavihar University

smitasankhe@somaiya.edu

Lecture -01



SOMAIYA
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

smitasankhe@somaiya.edu



Contents

- What is Programming?
- Different approaches for Programming
- Different Programming Languages
- Difference between C,C++,Java
- What is Java?
- Features of Java
- Sample Program

What is Programming?

- Programming is somewhat like working with building blocks.
- The five basic elements in programming are:
 - **input**: getting data and commands into the computer
 - **output**: getting your results out of the computer
 - **arithmetic**: performing mathematical calculations on your data
 - **conditional**: testing to see if a condition is true or false
 - **looping**: cycling through a set of instructions until some condition is met
- Example: ATM Machine

Computer Programming

- Way of giving computers instructions about what they should do next.

These *instructions are known as code*, and computer programmers write code to solve problems or perform a task.

- End goal is to create something:

Anything from a web page or a piece of software or even just a pretty picture.

- A computer program is a list of instructions that enable a computer to perform a specific task.
- Computer programs can be written in high and low level languages, depending on the task and the hardware being used.

Low level

```
1010010110111010
1001110110000111
0001110010110001
1011010110111010
0000111001010111
1001110010011101
```

High level

```
sale_price = 1.66
if (sale_price > 2) {
    discount = 0.1
}
else {
    discount = 0.05
}
```

Processing time

Slow Fast

Processing time

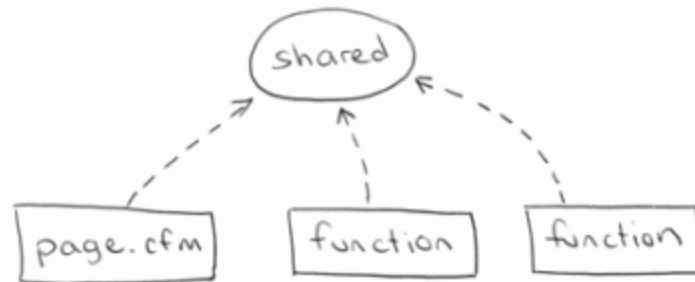
Slow Fast

Approaches of Programming

- Procedural Programming
- Structured Approach
- Object Oriented Approach

Procedural programming

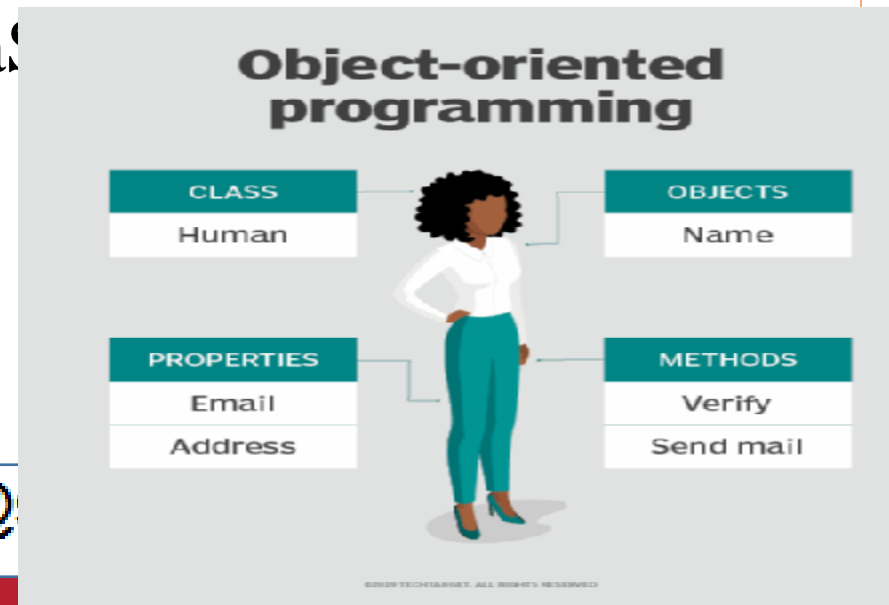
- Programming paradigm, **derived from structured programming**, based on the concept of the *procedure call*.
- Procedures, also known as routines, subroutines, or functions, simply contain a series of computational steps to be carried out.
- Example: Fortran, ALGOL, COBOL, BASIC, Pascal and C, C++



smitasankhe@somaiya.edu

Object-oriented programming

- Object-oriented programming (OOP) is a programming language model in which programs are organized around data, or objects, rather than functions and logic.
- An object can be defined as a data field that has unique attributes and behaviour.
- Example: Java, Python, JavaScript



Difference Between C,C++,Java

| Feature | c | C++ | Java |
|---------------------------------|-------------------------------|---|--|
| Paradigms | Procedural | Procedural , OOP | Purely Object oriented |
| Pointers | Yes, Very commonly used | Yes, very commonly used, but some form of references available too. | No Pointers; references are used instead |
| Pre-processor directives | Supported (#include, #define) | Supported (#include, #define) | Not Supported |
| Header files | Supported | Supported | Use Packages (import) |

| | | | |
|----------------------------------|--|--|---|
| Storage Allocation | Uses malloc, calloc | Uses new, delete | uses garbage collector |
| Database Connectivity | Not Supported | Not Supported | Supported |
| Code Translation | Compiled | Compiled | Interpreted |
| Complex Data Types | Structures, Unions | Structures, Unions, Classes | Classes |
| String Type | Character ,Arrays | Character ,Arrays, Objects | Objects |
| Use of Exception handling | C does uses exception handling for exception generated while execution of program. | C++ makes use of exception handling for handling various types of exceptions generated while execution of program. | Java makes use of exception handling in a very effective way to handle exceptions generated while execution of program. |

Operator overloading

There is no such concept of operator overloading in C

C++ uses concept of operator overloading

There is no such concept of operator overloading in Java

Multithreading and Interfaces

Not Supported

Not Supported

Supported

Inheritance

No Inheritance

Supported

Multiple Inheritance not Supported

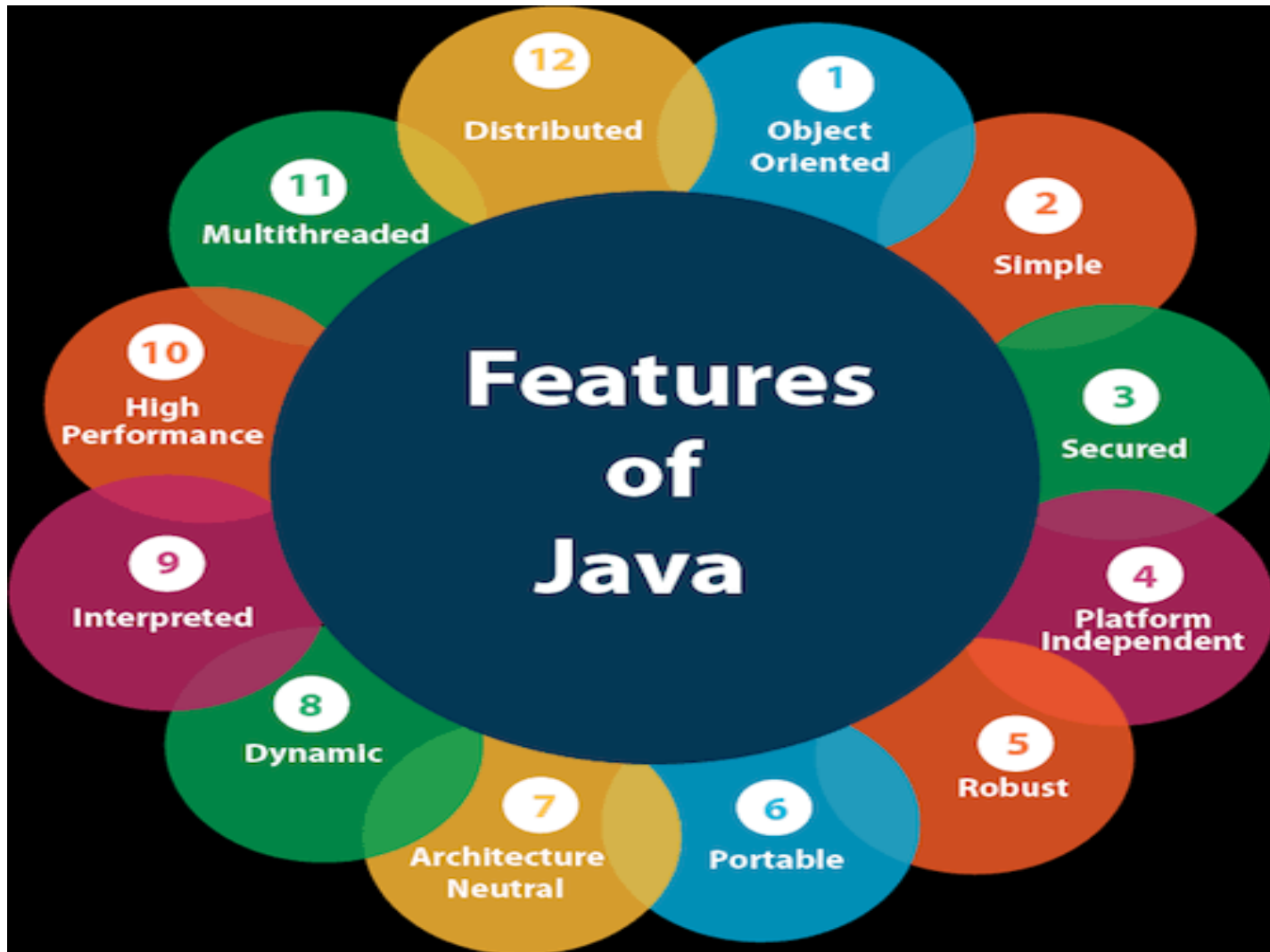
What is java ?

- Java is a **programming language** and a **platform**.
- Java is a high level, robust, object-oriented and secure programming language.
- **Platform:** Any hardware or software environment in which a program runs, is known as a platform.

Different Programming Paradigms

- Functional/procedural programming:
 - program is a list of instructions to the computer
- Object-oriented programming
 - program is composed of a collection *objects that communicate with each other*

Features of Java



smitasankhe@somaiya.edu

A list of most important features of Java language is given below

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic

Simple

- Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because
- Java syntax is based on C++ (so easier for programmers to learn it after C++).

Object-oriented

- Java is an **object-oriented** programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.
- Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
- Basic concepts of OOPs are:
 - Object
 - Class
 - Inheritance
 - Polymorphism
 - Abstraction
 - Encapsulation

Objects and Classes in Java

- An object in Java is the physical as well as a logical entity, whereas, a class in Java is a logical entity only.
- An entity that has state and behaviour is known as an object e.g., chair, bike, marker, pen, table, car, etc. An object has three characteristics:
- **State:** represents the data (value) of an object.
- **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

- **An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.
- **Object Definitions:**
 - An object is *a real-world entity*.
 - An object is *a runtime entity*.
 - The object is *an entity which has state and behavior*.
 - The object is *an instance of a class*.

Class

- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
- A class in Java can contain:
- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**

Inheritance

- **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviours of a parent object. It is an important part of OOPs (Object Oriented programming system).
- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

Polymorphism

- **Polymorphism in Java** is a concept by which we can perform a *single action in different ways*. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.
- There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism.
- We can perform polymorphism in java by method overloading and method overriding.

Abstraction

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- Another way, it shows only essential things to the user and hides the internal details, for **Example**: sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Encapsulation

- **Encapsulation in Java** is a *process of wrapping code and data together into a single unit*.
- Example, a capsule which is mixed of several medicines.

Platform Independent

- Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines
- while Java is **a write once, run anywhere language**.
A platform is the hardware or software environment in which a program runs.

Class File

Mac/OS
JVM **windows**
JVM **Linux**
JVM



smitasankhe@somaiya.edu

Secured

- Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:
- **No explicit pointer**
- **Java Programs run inside a virtual machine**

Robust

- Robust simply means strong. Java is robust because:
- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Architecture-neutral

- Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

Portable

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

High-performance

- Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).
- Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Distributed

- Java is distributed because it facilitates users to create distributed applications in Java.

Multi-threaded

- A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.
- The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Dynamic

- Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.
- Java supports dynamic compilation and automatic memory management (garbage collection)

Getting Started with Java Programming

- A Simple Java Application
- Compiling Programs
- Executing Applications



A Simple Java Program

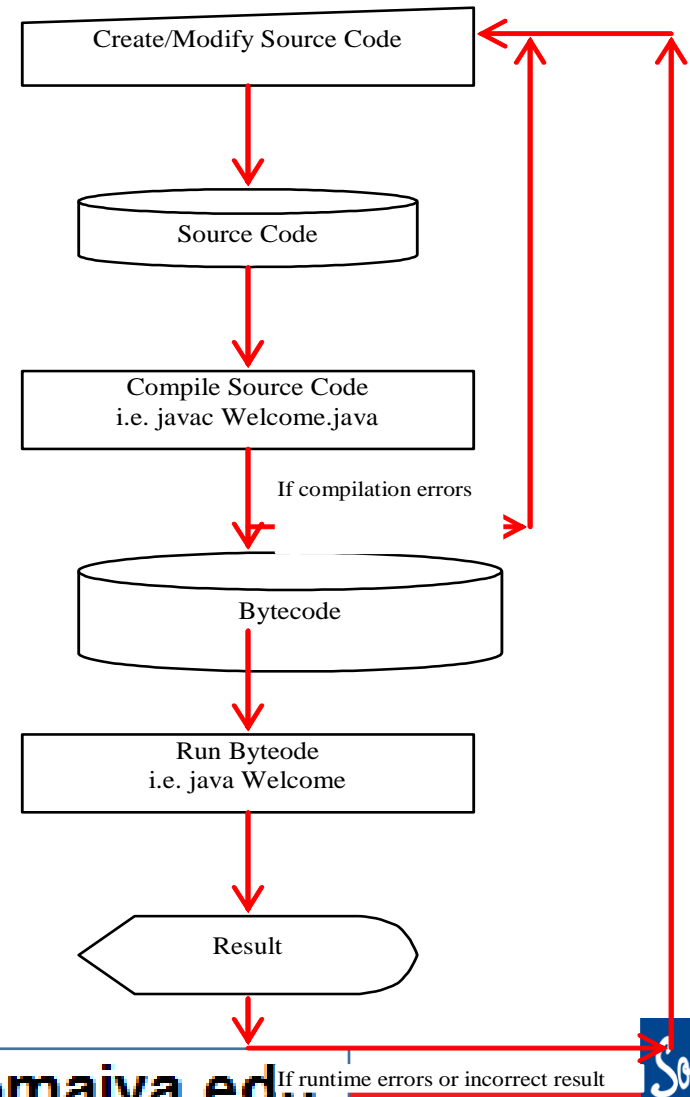
Example 1.1

//This application program prints
Welcome to Java!

```
public class Welcome
{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java!");
    }
}
```

Creating and Compiling Programs

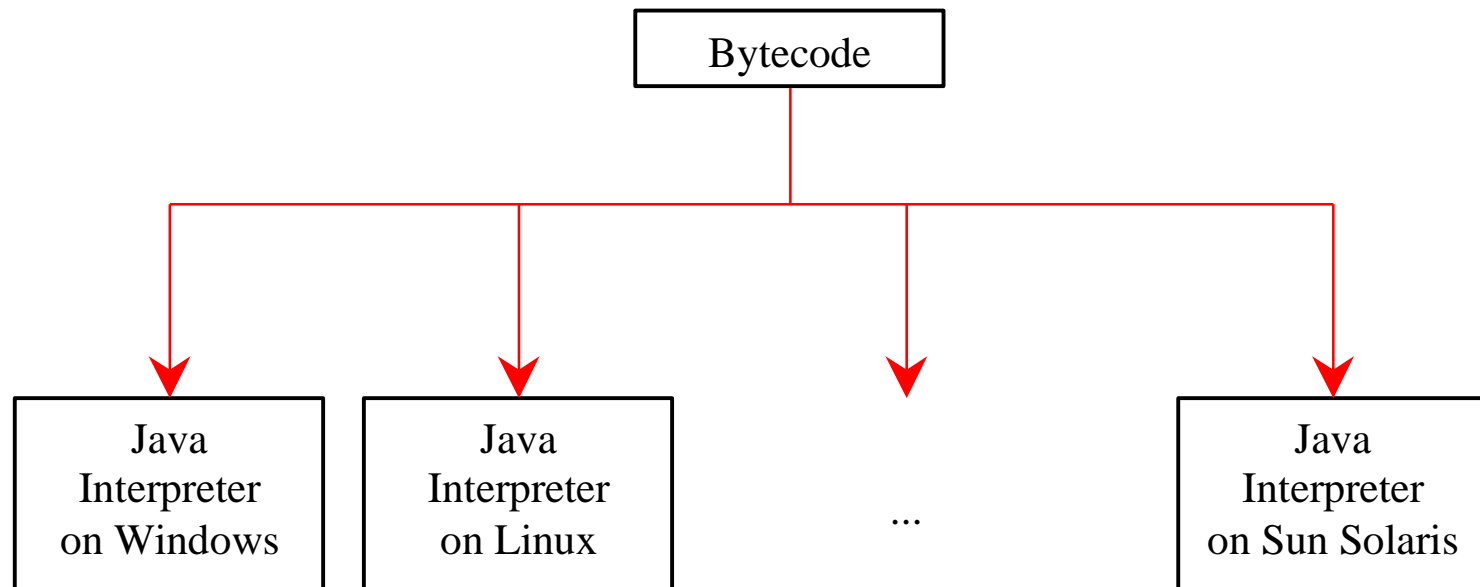
- On command line
 - `javac file.java`



smitasankhe@somaiya.edu

Executing Applications

- On command line
 - `java classname`



smitasankhe@somaiya.edu

```
javac Welcome.java
```

```
java Welcome
```

```
output:...
```