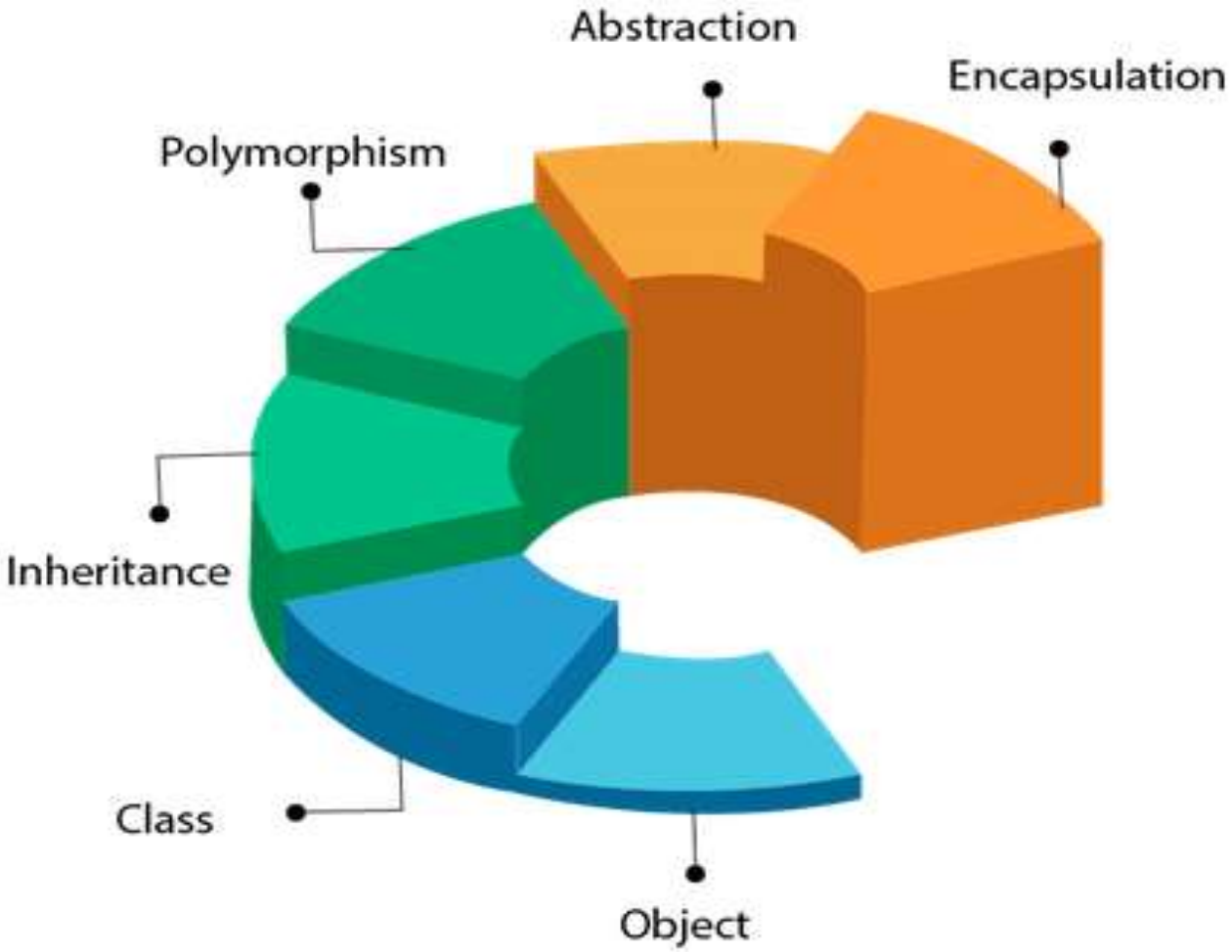
Semester III
BCA311 JAVA Programming
Introduction
(Unit-1)

Harshita Mathur
Department of Computer Science
Email : hmathur85@gmail.com

What is JAVA

- Java is a **programming language** and a **platform**.
- Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by *Sun Microsystems* in the year 1995. *James Gosling* is known as the father of Java.
- **Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

Object Oriented Concepts in JAVA



object

- Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

- An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

- Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

class

- *Collection of objects* is called class. It is a logical entity.

- A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

- *When one object acquires all the properties and behaviors of a parent object*, it is known as inheritance.

- It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

- If *one task is performed in different ways*, it is known as polymorphism.
- In Java, we use method overloading and method overriding to achieve polymorphism.

.

Abstraction

- Hiding internal details and showing functionality is known as abstraction.

- In Java, we use abstract class and interface to achieve abstraction.

Encapsulation

- Binding (or wrapping) code and data together into a single unit are known as encapsulation.

- A java class is the example of encapsulation.

Comparison of JAVA and C++

Comparison Index	C++	JAVA
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.

Comparison of JAVA and C++

Comparison Index	C++	JAVA
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.

Comparison of JAVA and C++

Comparison Index	C++	JAVA
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.

Comparison of JAVA and C++

Comparison Index	C++	JAVA
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Object-oriented	C++ is an object-oriented language.	Java is also an object-oriented language.

Features of JAVA

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

Features of java: **Simple**

- Java is very easy to learn, and its syntax is simple, clean and easy to understand.
- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Features of java: **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 behavior.

Features of java: **Portable**

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

Features of java: **Platform Independent**

- Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc.

- Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

Features of java: **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 sandbox

Features of java: **Robust**

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Features of java: Architecture Neutral

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

- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Features of java: Distributed

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

- RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Features of 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.

Features of java: 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++.

Semester III
BCA311 JAVA Programming
Introduction
(Unit-1)

Harshita Mathur

Department of Computer Science

Email : hmathur85@gmail.com

Java Source File Structure

- Java source file should start with a **package declaration**. The package declaration is optional for Java source file. If the package declaration is default than the package declaration is optional.

- After the package declaration there comes a section of **import declarations**. Import declarations can be zero or more. These import statements introduce type or static member names in the source code. Its mandatory to place all import declarations before actual class declarations in Java source code.

Java Source File Structure

- After import statements comes the **type declarations** statements. There can be any number of such type declarations, generally consisting of **class** , **interface** etc. The order of type declarations is not mandatory.

- The Java source file should have one and only one **public class**. The class name which is defined as public should be the name of Java source file along with .java extention.

A simple java program

```
//Name of this file will be "Hello.java"
public class Hello
 {
     public static void main(String[] args)
     {
        System.out.println("Hello Java");
    }
 }
```

Some Keywords

- When the main method is declared **public**, it means that it can also be used by code outside of its class, due to which the main method is declared public.

- The word **static** used when we want to access a method without creating its object, as we call the main method, before creating any class objects.

- The word **void** indicates that a method does not return a value. main() is declared as void because it does not return a value.

- **main** is a method; this is a starting point of a Java program.

- **String[] args:** It is an array where each element of it is a string, which has been named as "args".

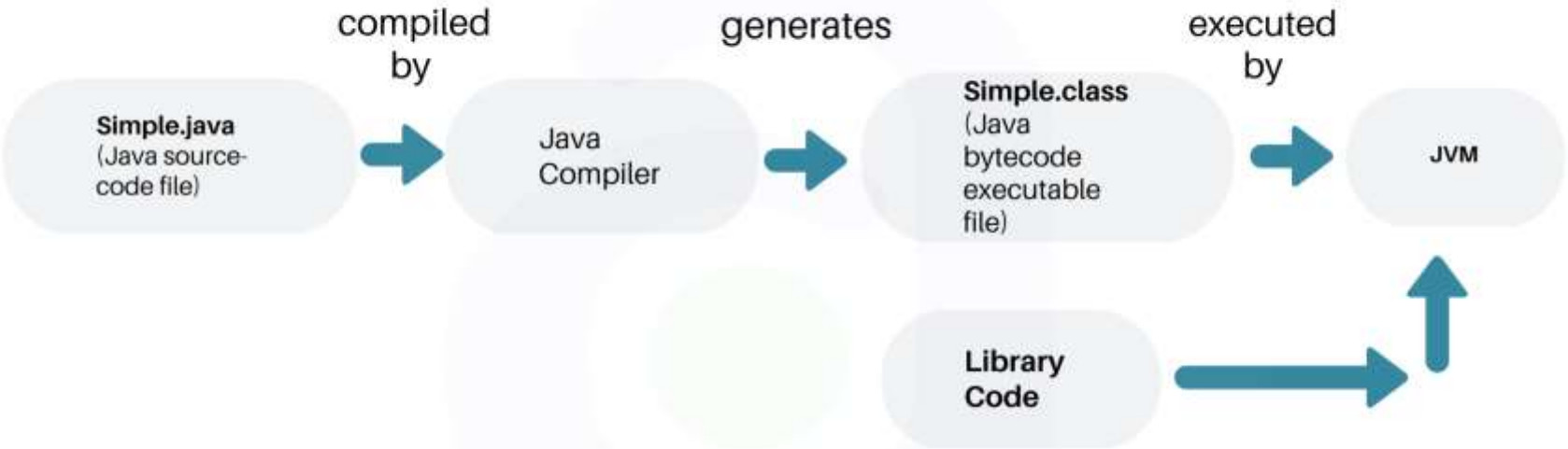
  If your Java program is run through the console, you can pass the input parameter, and main() method takes it as input.

- **System.out.println();** This statement is used to print text on the screen as output, where the **system** is a predefined class, and **out** is an object of the **PrintWriter** class defined in the system.

  The method **println** prints the text on the screen with a new line. You can also use **print()** method instead of println() method.

  All Java statement ends with a semicolon.

Compilation and Execution



Java source code is translated into bytecode.

Compilation and Execution

- Java source code is **compiled** into Java **bytecode** and Java bytecode is interpreted by the **JVM (JAVA Virtual Machine).**

- Your Java code may use the code in the Java library. The JVM executes your code along with the code in the library.

Compilation and Execution

To **execute** a Java program is to run the program's bytecode. You can execute the bytecode on any platform with a JVM, which is an interpreter.

It translates the individual instructions in the bytecode into the target machine language code one at a time rather than the whole pro- gram as a single unit.

Each step is executed immediately after it is translated.

JVM,JRE and JDK

- **JVM**(Java Virtual Machine) is a program that runs pre compiled Java programs, which mean JVM executes .class files (byte-code) and produces output.
- **JRE**(Java Runtime Environment) is an implementation of the JVM which actually executes Java programs. It includes the JVM, core libraries and other additional components to run applications written in Java. Java Runtime Environment is a must install on machine in order to execute pre compiled Java Programs.

- **JDK**(Java Development Kit)is needed for developing Java applications. It is a bundle of software that is used to develop Java based applications. It includes the JRE, set of API classes, Java compiler and additional files needed to write java applications.

- Conclusively, to compile and run Java program you would need JDK installed, while to run a pre compiled Java class file (byte-code) you would need JRE. JRE contains Java interpreter but not Java compiler.

Compiling and Executing java file on console

- To complie: javac filename.java
- To Execute: java filename