Class 9 notes of chapters 1 and 2

Chapter 1. Introduction to object oriented programming concepts

Program:-

A computer program is a collection of instructions that can be executed by a computer to perform a specific task. Most computer devices require programs to function properly. A computer program is usually written by a computer programmer in a programming language.

Types of programming languages:-

1. Low level languages
2. High level languages

1. Low level languages:

Low level languages are used to write programs that relate to the specific architecture and hardware of a particular type of computer.They are closer to the native language of a computer (binary), making them harder for programmers to understand.Examples of low level language:

1. Assembly Language
2. Machine Code

Assembly Language:

Few programmers write programs in low level assembly language, but it is still used for developing code for specialist hardware, such as device drivers.It is easy distinguishable from a high level language as it contains few recognisable human words but plenty of mnemonic code.

Advantages

1. Can make use of special hardware or special machine-dependent instructions (e.g. on the specific chip)
2. Translated program requires less memory
3. Write code that can be executed faster
4. Total control over the code
5. Can work directly on memory locations
6. They are closer to the native language of a computer (binary), making them harder for programmers to understand.

Machine Language:-

Machine language is the language understood by a computer. It is very difficult to understand, but it is the only thing that the computer can work with. All programs and programming languages eventually generate or run programs in machine language. Machine language is made up of instructions and data that are all binary numbers.

High Level Languages:-

A high-level language is any programming language that enables development of a program in a much more user-friendly programming context and is generally independent of the computer's hardware architecture.

A high-level language has a higher level of abstraction from the computer, and focuses more on the programming logic rather than the underlying hardware components such as memory addressing and register utilization.

Programming language generations:

A programming language is a set of written symbols that instructs the computer hardware to perform specific tasks. Typically, a programming language consists of a vocabulary and a set of rules (called syntax) that the programmer must learn.

1. 1st generation of programming languages
2. Second generation programming languages
3. Third generation programming languages
4. 4th generation programming languages
5. Fifth generation programming languages

Programming Paradigms: -

The term programming paradigm refers to a style of programming. It does not refer to a specific language, but rather it refers to the way you program.There are lots of programming languages that are well-known but all of them need to follow some strategy when they are implemented. And that strategy is a paradigm.

1. Procedural programming or modular programming:-

Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program’s execution, any given procedure might be called at any point, including by other procedures or itself.

Languages used in Procedural Programming:

FORTRAN, ALGOL, COBOL, BASIC, Pascal and C.

Object Oriented Programming:

It can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object oriented programming, computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Languages used in Procedural Programming:

Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Objective-C, Dart, Swift, Scala.

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. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.In Java, we use method overloading and method overriding to achieve polymorphism.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.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. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.
Advantage of OOPs over Procedure-oriented programming language
1) OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.
2) OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.
3) OOPs provides the ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

Ch 2.Introduction to Java.

Introduction to Java

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This tutorial gives a complete understanding of Java. This reference will take you through simple and practical approaches while learning Java Programming language.

Features of Java: -

Object Oriented − In Java, everything is an Object. Java can be easily extended since it is based on the Object model

Platform Independent − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

Simple − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

Secure − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Architecture-neutral − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Portable − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

Robust − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

Applications of Java Programming: -

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be Write Once, Run Anywhere.

Multithreaded − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

Interpreted − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

High Performance − With the use of Just-In-Time compilers, Java enables high performance.

Distributed − Java is designed for the distributed environment of the internet.

Dynamic − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

What is Java Bytecode?

Java bytecode is the instruction set for the Java Virtual Machine. It acts similar to an assembler which is an alias representation of a C++ code. As soon as a java program is compiled, java bytecode is generated. In more apt terms, java bytecode is the machine code in the form of a .class file. With the help of java bytecode we achieve platform independence in java.

JAVA VIRTUAL MACHINE
Java virtual machine is a platform-independent abstract machine that provides a runtime environment in which the Java byte code is executed.
It is a part of Java runtime environment that converts the Java bytecode into machine-readable language. The main method that we have in a Java program is actually called by the Java virtual machine.

Following are a few tasks that Java virtual machine does.

- Loads the code
- Code verification
- Execution of the code
- It provides the run-time environment for the applications
- Memory area
- Register set
- Provides a garbage collection heap
- Reporting of the fatal errors
- Provides a class file format

Java compilation process:-

- We create a program using program editor(eg. Notepad) and save it as filename.java file in specified directory.
- Then we compile it using javac command(eg. javac filename.java), A java compiler then convert .java file into .class file which is called as bytecode and this bytecode is platform independent can run on different platforms eg.Windows, Linux,MAC etc.
- If any bugs found during compilation that should be fixed first then compiled again, If there is not any bugs found then bytecode is created and handed over to JVM(java virtual machine) for interpretation.
- JVM interpret class file by running java space class name command(eg. java filename) as given in figure.
- JVM has three components to interpret program first: Bytecode verifier which verifies bytecode, second: Class loader which loads required classes from java class library, third one is a JIT(just in time) compiler which compiles class file and convert it into machine readable format and then hand over to operating system.
- If there any bugs(mostly logical) found during interpretation(by JVM) then one have to edit program to fix bugs and compile it again.If not any more bugs then final output will be displayed on screen.