2.1 Java as a Programming Platform

Java is a versatile and powerful programming platform that allows developers to create a wide range of software applications. It's not just a single programming language, but rather a complete environment that includes a few important components.

1. **Java Programming Language:** At its core, Java is a high-level, object-oriented programming language. It provides a foundation for writing code that can run on different types of devices and operating systems. Java's syntax is similar to C and C++, making it relatively easy for developers to learn.
2. **Java Virtual Machine (JVM):** This is a crucial part of the Java platform. When you write Java code, it's compiled into a special type of code called bytecode. This bytecode is not specific to any particular computer or operating system. Instead, it's designed to be executed by the JVM, which translates it into machine code that's specific to the device it's running on. This "write once, run anywhere" capability is one of Java's defining features.
3. **Java Runtime Environment (JRE):** The JRE is a package that includes the JVM along with some libraries and other components that are necessary to run Java applications. When you install Java on your computer, you're actually installing the JRE.
4. **Java Development Kit (JDK):** This is a more comprehensive package that includes not only the JRE, but also the tools needed to develop Java applications. It includes a compiler that turns your Java source code into bytecode, along with other utilities for testing and debugging.
5. **Java Standard Library:** This is a collection of pre-written code (libraries) that provide commonly used functions and data structures. It saves developers a lot of time because they don't have to reinvent the wheel every time they write a new program.

Java's platform independence is one of its standout features. Once a program is compiled into bytecode, it can run on any device or operating system that has a compatible JVM. This makes Java especially popular for web-based applications, as it allows them to run in a browser on any device.

Furthermore, Java's object-oriented nature encourages clean and modular code, making it easier to maintain and extend software over time. It's widely used in various domains, including web development, mobile app development (especially for Android), enterprise-level applications, and more. Java's robustness, security features, and extensive community support contribute to its enduring popularity in the world of software development

2.2 History of Java

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of Java starts with the Green Team. Java team members (also known as Green Team), initiated this project to develop a language for digital devices such as set-top boxes, televisions, etc. However, it was best suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

Since then, Java has become integral to a multitude of domains including internet programming, mobile devices, gaming, and e-business solutions. Its versatility and wide-ranging applicability continue to make it a cornerstone of modern software development.

2.3 Java Buzzwords

- The authors of Java used 11 buzzwords for summarizing their "White Paper" explaining their design goals and accomplishments. Java has many advanced features, a list of key features is known as Java Buzz Words. These buzzwords are Simple, Object Oriented, Distributed, Robust, Secure, Architecture-Neutral, Portable, Interpreted, High Performance, Multithreaded and Dynamic.

- **Simple**
  - Java was designed to be **easy** for the professional programmer **to learn and use effectively.**
  - Java is a **cleaned up version of the syntax for C++** as there is no need for header files, pointers, structures, unions, operator overloading, virtual base classes and so on.
  - Since Java **inherits the C/C++ syntax and many of the object oriented features of C++** learning Java will require very little effort for programmers with understanding of basic concepts of object oriented programming language and experience with C++.

- **Object Oriented**
  - Simply stated, object oriented is a **technique for programmin**g that **focuses on the data** in the form of objects and on the **interfaces** to that object.
  - The object oriented features of Java are essentially derived from C++.
  - The major difference between Java and C++ lies in multiple inheritance, which Java has replaced with the simpler concept of interfaces.

- **Distributed**
  - Java has an extensive **library of routines for handling TCP/IP protocols** like HTTP and FTP so Java is **designed for** the **distributed environment** of the Internet.
  - Java applications **can open and access objects across the network via URLs** with the same ease as when accessing a local file system.
  - Its simple in Java for tasks like opening a socket connection which make the communication through the network possible.
  - Java servlets makes server side processing in Java extremely efficient.
  - Java have support for **Remote Method Invocation (RMI)** which enables a program to **invoke methods across a network** facilitating communication between distributed objects.

- **Robust**
  - Java is intended for writing programs that must be reliable in variety of ways and in variety of environments .
  - To increase reliability, Java puts a lot of emphasis on early checking for possible problems during compile time later dynamic checking during runtime thus decreasing the chance of occurrence of errors.
  - This makes sure that written program will behave in a predictable way under diverse conditions.
  - Java also provides memory management and exception handling features which helps to eliminate the errors due to manual handling of memory and exceptions.

- **Secure**
  - Java is intended to be used in networked/distributed environments
  - Java applications are confined to Java execution environment so java enables the construction of virus free, tamper free systems
  - Applications developed using Java provide confidence that no harm will be done and that no security will be breached when using them

- As, the java code runs inside Java Virtual Machine (JVM), following attacks are impossible:
  - ✓ Overrunning the runtime stack
  - ✓ Corruption memory outside its own process space
  - ✓ Reading or writing files without permission

- **Architecture Neutral**
  - In early days, programmers faced problem of written program not running or malfunctioning due to operating system upgrades, processor upgrades and changes in core system resources.
  - To solve this problem, Java designers developed Java Virtual Machine with goal write once, run anywhere, any time, forever.
  - In Java, the compiler (generates an architecture neutral object file format so that the complied code is executable on many processors and operating systems, given the presence of the Java runtime system.
  - The Java compiler does this by generating bytecode instructions which is independent of any particular computer architecture
  - The bytecode instructions are designed to be both easy to interpret on any machine and easily translatedinto native machine code on the fly
  - Interpreting bytecodes is necessarily slower than running machine instructions but by using a process called just in time compilation virtual machines can make bytecodes to execute faster by translating the most frequently executed bytecode sequences into machine code.

- **Portable**
  - Different operating system and computer architecture are present in networked and distributed environment and programs written in one system must run as expected in any other system present in the environment.
  - This was a major problem faced by application developer and the designers of Java wanted to solve this problem and made Java specification aspects **implementation-independent** unlike C and C++
  - For example, an " in Java is always a 32 bit integer but " in C/C++ can mean a 16 bit integer, a 32 bit integer, or any other size that the compiler vendor likes.

- **Interpreted**
  - Java enables creation of cross platform programs by compiling into an intermediate representation called Java bytecode.
  - This code can be executed on any system that implements the Java Virtual Machine.
  - JVM is responsible for interpreting the bytecode for the native system where it resides.

- **High Performance**
  - Most previous attempts at cross platform solutions have done so at the expense of performance.
  - Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just in time compiler.
  - Java run time systems that provide this feature lose none of the benefits of the platform independent code.
  - Just in time compiler can provide sophisticated optimization such as by monitoring which code is executed frequently it can optimize just that code for speed.

- **Multithreaded**
  - The benefits of multithreading are better interactive responsiveness and real time behaviour.
  - Java was designed to meet the real world requirement of creating interactive, networked programs.
  - To accomplish this, Java supports **multithreaded programming** which allows to write programs that do many things simultaneously.
  - The Java run time system also provides solution for multi process synchronization that enables to construct smoothly running interactive systems.
  - But Java leaves the implementation of multithreading to the underlying operating system or a thread library which makes multithreading platform dependent.

- **Dynamic**
  - Java was designed to adapt to an evolving environment
  - Java programs have significant amount of run-time type information that is used to verify and resolve accesses to objects at run time
  - This makes it possible to **dynamically link code in safe and practical manner.**

2.4 Java Virtual Machine

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

It is:

1. A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
2. An implementation Its implementation is known as JRE (Java Runtime Environment).
3. Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.