1. **What is Java?**
2. **Java is a platform Independent language?** (Video)
3. **What is Class?**
   1. Template (objects are instances of these template)
4. **Difference between JDK, JVM, JRE?** (video)
5. **What is Refactoring?** (video)
6. [**What is Encapsulation?**](https://www.geeksforgeeks.org/encapsulation-in-java/) (video)
7. [**What is Polymorphism? Way to acheive it with Examples**](https://www.geeksforgeeks.org/polymorphism-in-java/)
8. [**Difference between Method Overloading and Method Overriding**](https://beginnersbook.com/2014/01/difference-between-method-overloading-and-overriding-in-java/)
9. [**What is Abstraction?**](https://www.geeksforgeeks.org/abstraction-in-java-2/) (video)
10. [**Difference Between Encapsulation and Abstraction**](https://www.geeksforgeeks.org/difference-between-abstraction-and-encapsulation-in-java-with-examples/)
11. [**Association, Aggregation and Composition?**](https://www.geeksforgeeks.org/association-composition-aggregation-java/)
12. [**Access Modifiers**](https://www.geeksforgeeks.org/access-modifiers-java/)
13. [**Multiple Inheritance and Daimond Problem**](https://www.geeksforgeeks.org/java-and-multiple-inheritance/)
14. **What is Java Wrapper Classes Why we use them?** (video)
15. **Difference between String, StringBuilder, StringBuffer?** (google)
16. **Difference between Array and ArrayList? (google)**
17. **What is Inheritance? Why we use it?**
18. **What is Abstract Class?**
19. **What is Abstract Function?**
20. **Why we Use Abstract Class?**
21. **What is Interface? (very Important)**
22. **Difference between Abstract class and Interface? (very Important)**
    1. **Similarity:** Both of them are used to achieve abstraction.
    2. **Final Variables:** Variables declared in a java interface are by default final. An abstract class may contain non-final variables.
    3. **Types of variables:** Interface can have only static and final variables. Abstract classes can have final, non-final, static and non-static variables.
    4. **Implementation:** Abstract class can provide the implementation of interface. Interface can’t provide the implementation of abstract class. That means abstract class can inherit interface but interface can’t inherit abstract class.
    5. **Keyword used:** Java interface is implemented using “implements” keyword, while an abstract class can be extended using “extends” keyword.
    6. **Multiple Implementation:** An interface can extend another java interface only, while an abstract class can extend another class as well as multiple java interfaces.
    7. **Accessibility:** Members of Java interface are public by default. A java abstract class can access specifier is default.
23. **When to use what?**
    1. Consider using abstract classes if any of these statements apply to your situation:
       1. In java application, there are some related classes that need to share some lines of code then you can put these lines of code within abstract class and this abstract class should be extended by all these related classes.
       2. You can define non-static or non-final field(s) in abstract class, so that via a method you can access and modify the state of Object to which they belong.
       3. You can expect that the classes that extend an abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).
    2. Consider using interfaces if any of these statements apply to your situation:
       1. It is total abstraction, All methods declared within an interface must be implemented by the class(es) that implements this interface.
       2. A class can implement more than one interface. It is called multiple inheritance.
       3. You want to specify the behaviour of a particular data type, but not concerned about who implements its behavior.
24. **What is Collections? Why we use it?**
25. **Difference between ArrayList, LinkedList, Vector?**
26. **What is Generics? And Why do we use it?**
27. **What is Functional Programming?**
28. **What is a Lambda Expression?**
29. **What is Optional Class?**
30. **What is Functional Interface?**
31. **Behind the Scene of Functional Interface?**
32. **What is Functional References? Example**
33. **Advantages and Disadvantages of Using Functional Programming?**
34. **What is Thread?**
35. **What is Multithreading? Why we use it?**
36. **Ways to create thread?**
37. **Example Program to create thread.**
38. **States of Thread.**
39. **How to set priority in thread.**
40. **What is synchronized in terms of thread?**
41. **What is join, sleep, yield in thread?**
42. **Why we need to Executor service?**
43. **What is executor service?**

Java is:-

* High Level
* Object-oriented
* Secure
* Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java.

**Applications**:-

Approximately 3 billion devices run java. There are many devices where java is currently used. Some of them are:-

* Desktop Applications such as acrobat reader, media player, antivirus, etc.
* Web Applications such as irctc.co.in, javatpoint.com, etc.
* Enterprise Applications such as banking applications.
* Mobile Applications
* Games etc.

**Features of Java:-**

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

**Difference between C++ and Java:-**

**Compilation Flow:-**

**At compile time**

* At compile time, java file is compiled by java compiler (it does not interact with OS) and converts the java code into bytecode.

**At runtime**

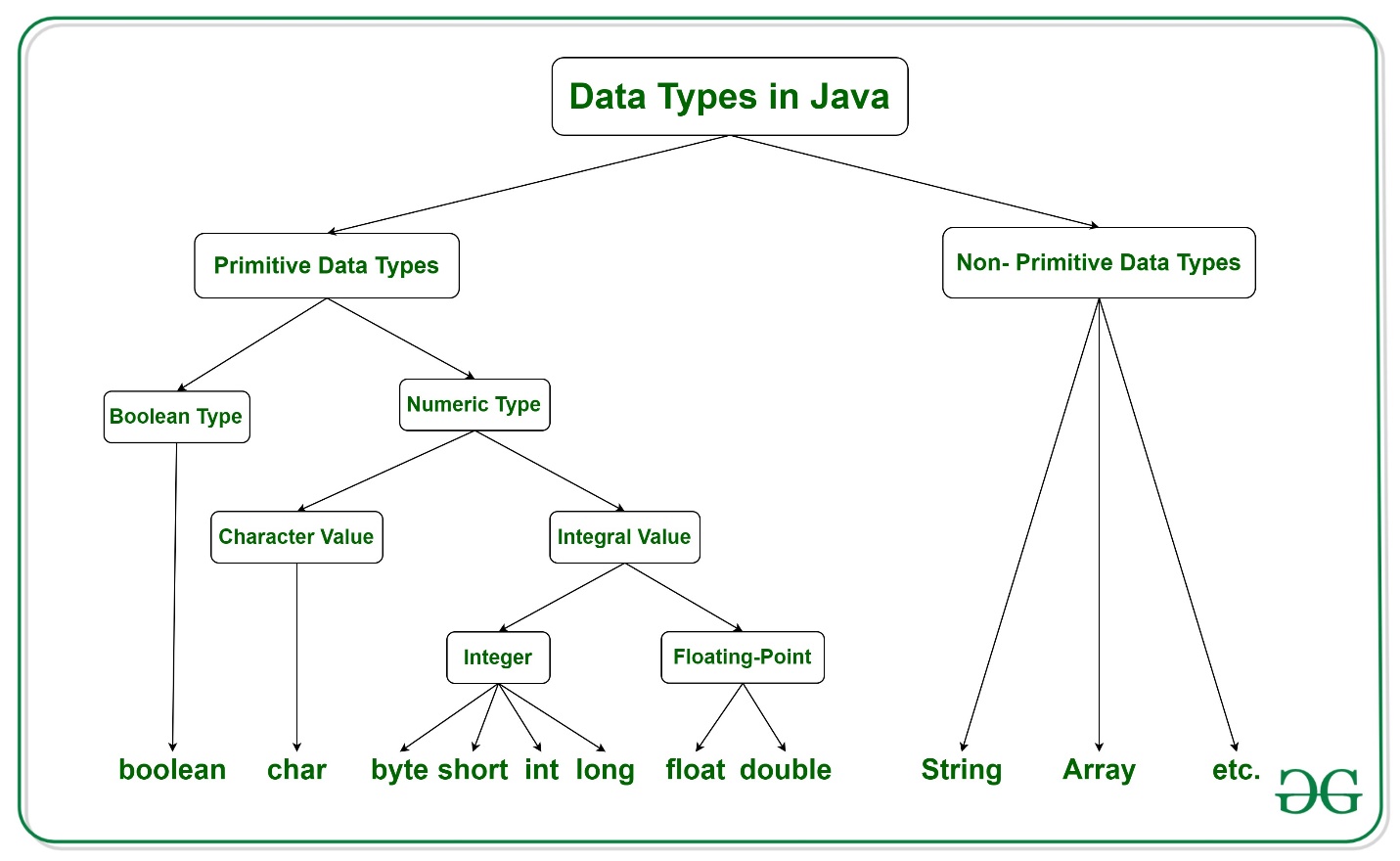
* **Classloader:** is the subsystem of JVM that is used to load class files.
* **Bytecode Verifier:** checks the code fragments for illegal code that can violate access right to objects.
* **Interpreter:** read bytecode stream then execute the instructions.

**What is JVM, JRE, JDK and difference between them.**

**Types of Variables:-**

* Local Variables
* Instance Variables
* Static Variables

**Data Types in Java**:-



Java has two categories of data:

* **Primitive Data Type:** such as Boolean, char, int, short, byte, long, float and double
* **Non-Primitive Data Type or Object Data type:** such as String, Array, etc.

There are **8 primitive data types**:

1. Boolean
2. Byte
3. Short
4. Int
5. Long
6. Float
7. Double
8. Char

There are **Non-primitive data types**:

1. String
2. Class
3. Object
4. Interface
5. Array

**Operators:-**

* Unary Operator, (++, --, !, ~)
* Arithmetic Operator,
* Shift Operator,
* Relational Operator,
* Bitwise Operator,
* Logical Operator,
* Ternary Operator and
* Assignment Operator