**Java basic Fundamentals**

1. **Java Virtual Machine (JVM):** Java is a compiled and interpreted language. Code is first compiled into an intermediate form called bytecode. JVM is responsible for executing this bytecode, making Java a platform-independent language.
2. **Object-Oriented Programming (OOP):** Java is an object-oriented language, emphasizing the use of classes and objects. This paradigm promotes concepts like encapsulation, inheritance, and polymorphism for building modular and reusable code.
3. **Classes and Objects:** In Java, everything is considered an object. A class is a blueprint for creating objects, defining attributes (fields) and behaviors (methods). Objects are instances of classes.
4. **Inheritance:** Inheritance allows a class to inherit properties and behaviors from another class. It promotes code reuse and the creation of a hierarchy of classes.
5. **Polymorphism:** Polymorphism means the ability of an object to take on multiple forms. In Java, polymorphism is achieved through method overloading (same method name, different parameters) and method overriding (providing a specific implementation in a subclass).
6. **Encapsulation:** Encapsulation is the bundling of data and methods that operate on that data into a single unit known as a class. It helps in controlling access to the internal details of a class and promotes data hiding.
7. **Abstraction:** Abstraction involves simplifying complex systems by modeling classes based on the essential features. It hides the implementation details and only exposes what is necessary.
8. **Data Types:** Java has primitive data types (int, double, char, etc.) and objects (String, Arrays) to represent and manipulate data. Each data type has a specific range and behavior.
9. **Variables:** Variables are containers for storing data values. They must be declared with a specific data type before use. Variables can be instance variables (belonging to an instance of a class) or class variables (shared among all instances).
10. **Control Flow:** Java supports standard control flow structures such as if-else statements, loops (for, while, do-while), and switch statements for conditional and iterative execution of code.
11. **Exception Handling:** Exception handling in Java allows the program to gracefully handle runtime errors. The try-catch block is used to catch and handle exceptions.
12. **Interfaces:** Interfaces in Java define a contract of methods that implementing classes must adhere to. They provide a way to achieve multiple inheritances and support abstraction.
13. **Packages:** Packages are used to organize classes into namespaces. They help in avoiding naming conflicts and provide a modular structure to the code.
14. **Garbage Collection:** Java has automatic garbage collection, where the JVM automatically reclaims memory occupied by objects that are no longer in use.
15. **Concurrency:** Java supports multithreading, allowing the execution of multiple threads concurrently. This is useful for tasks that can be performed simultaneously.

These fundamental concepts provide a solid foundation for understanding Java programming. They contribute to Java's readability, reusability, and maintainability.