1. **Explain the different types of variables and their uses in context**
   In Java, variables are categorized based on their scope, data type, and how they are stored in memory.
   Here are the main types of variables in Java:

   Instance Variables (Non-Static Fields)
   Definition: These are variables declared inside a class but outside any method, constructor, or block, and are not marked as static.
   Scope: They belong to an instance of a class. Each object of the class has its own copy of these variables.
   Initialization: If not explicitly initialized, they are automatically initialized with default values (e.g., 0 for numeric types, null for objects).
   Use Case: Instance variables store data unique to each object. For example, in a Person class, name and age could be instance variables, as each Person object has its own name and age.

   Local Variables
   Definition: These are variables declared with the static keyword inside a class, but outside any method, constructor, or block.
   Scope: They belong to the class itself rather than to any particular object. Only one copy of a static variable exists, shared among all instances of the class.
   Initialization: Automatically initialized with default values if not explicitly set.
   Use Case: Class variables store data shared across all instances of a class. For example, a static counter that tracks the number of objects created.

   Parameters
   Definition: These are variables passed to methods or constructors when they are called.
   Scope: Local to the method or constructor they are defined in.
   Initialization: Automatically initialized with the value provided by the caller.

   Use Case: Parameters are used to pass data into methods. For example, in a method that adds two numbers, the numbers would be passed as parameters

2. **Explain Encapsulation and give an example of how it can be used**
   Encapsulation is one of the core principles of object-oriented programming (OOP) and is used to bundle data (fields) and methods (functions) into a single unit, usually a class.

   This principle helps to restrict direct access to some of the object's components and can prevent the accidental modification of data.

Usage: In Java, encapsulation is like that toy box. You hide your toy (data) inside the box (class), and you let others play with it only through special helpers (methods) that you control.

3. **Explain what coupling is and where it can be used**
   Coupling in Java refers to the degree of dependency between classes or modules. It impacts how changes in one part of the code affect other parts and influences the flexibility and maintainability of the software.

   Usage: Achieve loose coupling with interfaces or dependency injection, making the system more modular and easier to modify.

4. **Explain cohesion and an example.**
   Cohesion refers to how closely related the responsibilities of a single class or module are.

   A class with high cohesion focuses on a single task or responsibility, making it easier to understand, maintain, and reuse.

   In contrast, low cohesion occurs when a class tries to do too many unrelated tasks, making it harder to manage and more prone to errors

5. **How to perfect Java syntax**
   - Practice regularly by writing and reviewing code.
   - Follow conventions: Proper naming, indentation, and commenting.
   - Use IDEs for syntax highlighting and error checking.
   - Read and refactor: Study others' code and improve your own.