1. **Explain the keywords this and super.**

In Java, the keywords **`this`** and **`super`** serve important roles in object-oriented programming, particularly in relation to object instances and inheritance. Here's a detailed explanation of each:

`this` Keyword

Definition: The `this` keyword refers to the current instance of a class. It is used within instance methods or constructors to refer to the calling object.

`super` Keyword

Definition: The `super` keyword refers to the superclass (parent class) of the current object. It is used to access members (methods and variables) of the parent class.

2. **Explain the concepts of garbage collection and finalize ().**

arbage collection in Java is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

The Java **finalize() method** of Object class is a method that the Garbage Collector always calls just before the deletion/destroying the object which is eligible for Garbage Collection to perform clean-up activity. Clean-up activity means closing the resources associated with that object like Database Connection, Network Connection, or we can say resource de-allocation. Remember, it is not a reserved keyword. Once the finalize() method completes immediately, Garbage Collector destroys that object.

Finalization: Just before destroying any object, the garbage collector always calls finalize() method to perform clean-up activities on that object. This process is known as Finalization in Java.

3. Explain the purpose of package and how to import package.

A **package** in Java is a namespace that organizes a set of related classes and interfaces. It serves several important purposes:

1. **Namespace Management**: Packages help avoid naming conflicts by grouping related classes and interfaces under a unique namespace. For example, you can have two classes with the same name in different packages.
2. **Access Control**: Packages allow you to control access levels for classes, methods, and variables. Classes in the same package can access each other's package-private and protected members, while classes in different packages may have restricted access.
3. **Organizational Structure**: Packages provide a way to organize classes and interfaces into a hierarchical structure, making it easier to manage large applications.
4. **Reusability**: By packaging classes and interfaces, you can create reusable components that can be imported into other applications or modules.

In java, the import keyword used to import built-in and user-defined packages. When a package has imported, we can refer to all the classes of that package using their name directly. The import statement must be after the package statement, and before any other statement