Chapter 7: Java Inheritance

Inheritance:

Q) What is Inheritance?

A) It is the mechanism of deriving one class from the other.

Q) What is superclass ?

A) The class from which the other class is derived is called as superclass.

Q) What is subclass?

A) The class which is derived from the superclass is called as subclass.

The subclass is a specialized version of the superclass or it can be said that the subclass is the extended version of the superclass.

Uses of Inheritance:

In Java, we can create inheritance relationships by extending a class.

The most common reasons for using inheritance are:

- To promote code reuse
- To use polymorphism.

Polymorphism:

- The term polymorphism comes from two Greek words,
- poly means many and morphs means forms.
- Polymorphism allows us to perform various operations by using the same method.
- In Java it is possible to use a single method to perform different functions by changing the implementation of the method.
- Polymorphism can be static or dynamic:
    - In static, also known as early binding, the binding is performed during compilation time.
    - In dynamic, also known as late binding, the binding occurs during runtime, depending on the type of object.

Overloading V/S Overriding:

Points	Overloading	Overriding
Arguments:	We must change the type, number or sequence of arguments of overloaded methods.	We must not change the type, number or sequence of arguments in the argument list.
Access Modifier:	We can change the access modifier of an overloaded method.	We can change the access modifier of the overridden method that is less restrictive than the superclass version of the method.
Return Type:	We can change the return type of an overloaded method.	We cannot change the return type of overridden method
Declaration Context:	A method can be overloaded in the same class or in a subclass.	A method can only be overridden in a subclass.
Method call resolution:	At compile time, the *declared type of the reference is used* to determine which method will be executed at runtime.	The runtime type *of the* reference, i.e., the type of the object referenced at *runtime,* determines which method is selected for execution.