JA111

Day-7

Why Inheritance?

Inheritance

- **Inheritance is a mechanism by which one object acquired properties of another object.**
- **In enhances code reusability and keep the code DRY that leans to easy maintenance and scalability of the code.**
- **It is used to define create specific implementation from a general implementation.**
- **It is used to define is-a relationship which is Unidirectional.**
- **e.g. House is a Building. But Building is not a House.**

Super class/Base class/Parent class: A class from which another class is derived.

Sub class/Derived class/Child class: A class that is derived from a superclass.

- **Types of Inheritance**

Single Level Inheritance	Multi Level Inheritance	Hierarchical Inheritance
Multiple Inheritance	Hybrid Inheritance	

Syntax for Inheritance

```
class subclass-name extends superclass-name{
   //body of class
}
```

- Access modifiers are used to work in their usual way they do not prevent any members from being inherited they control accessibility only.

Constructor calling and inheritance

- An object of sub class has all the variables and methods of super class so the object of sub class can be called object of super class.
- All instance members and class members are getting inherited in the sub-class but constructor is not inherited.
- When an object of subclass is created then constructor of super class as well as of sub class will be called and constructors are called in the order of derivation i.e. they are called from super-class to sub-class.

super keyword

- **The default constructor (means constructor with no parameter) of super class is called implicitly when object of sub class is created but if super class do not have default constructor (means it has parameterized constructor(s) only) then when object of sub class is created then we have to call super class constructor explicitly using 'super' keyword.**

- **The super keyword must be the first statement inside the sub class constructor. This restriction is only applicable when super keyword is used for constructor calling.**

- **Another use of super keyword is to access hidden instance variable of super class and to call overridden method of super class in sub class.**

Method Overriding

- If a sub class has a method whose signature (means return-type method-name(parameter-list)) is same as that of its super class (but not body) then we say that sub class has overridden the method of super class.
- Method overriding is used to redefine (not adding) the behavior of super class means overriding does not add any new method in sub class.
- If overridden method is called inside the subclass body then version defined in the subclass will be called always and to access the version of super class in the sub class body we have to use super keyword.
- when overriding method in the subclass then use @Override annotation with method signature so that compiler will check if method signature in super class and sub class is same or not. If not same then compiler will report error.

final Keyword

- To create named constant
- To prevent method overriding
- To prevent inheritance

Method Overriding

- If a sub class has a method whose signature (means return-type method-name(parameter-list)) is same as that of its super class (but not body) then we say that sub class has overridden the method of super class.
- Method overriding is used to redefine (not adding) the behavior of super class means overriding does not add any new method in sub class.
- If overridden method is called inside the subclass body then version defined in the subclass will be called always and to access the version of super class in the sub class body we have to use super keyword.
- when overriding method in the subclass then use @Override annotation with method signature so that compiler will check if method signature in super class and sub class is same or not. If not same then compiler will report error.

final Keyword

- To create named constant
- To prevent method overriding
- To prevent inheritance

The Object class

This is inbuilt class that is parent class of all classes in java i.e. all members of Object class are available to all classes of java. The Object class has no variables.

Some of the important methods of Object class are-

1. public final Class getClass()
2. public int hashCode()
3. public boolean equals(Object o)
4. public String toString()
5. protected Object clone()
6. protected void finalize()
7. void notify()
8. void notifyAll()
9. void wait()
10. void wait(long timeout)
11. void wait(long timeout, int nanos)