Inheritance

Lecture 13

Prof. Anita Agrawal,
BITS –Pilani, K.K.Birla Goa campus

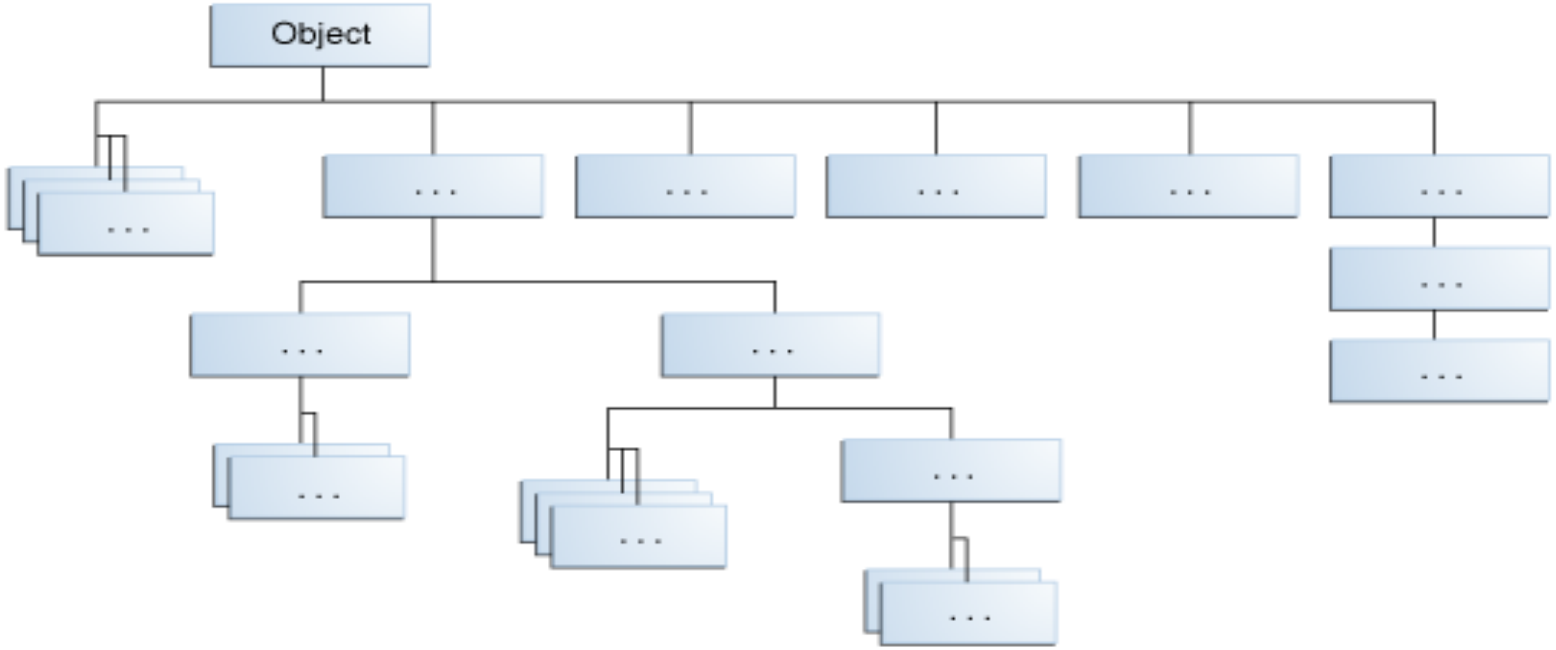
What is inheritance?

- In Java language, classes can be *derived* from other classes, thereby ***inheriting*** fields and methods from those classes.

- A class that is derived from another class: a ***subclass/derived class/extended class/ child class***).

- A class from which it is derived: ***superclass/base class/parent class***

- Except for the class Object, every other class has one and only one direct superclass (**single inheritance**).
- If there is no explicit superclass for a class, it is implicitly a subclass of the class object.
- Classes can be derived from classes which are derived from classes, ……. And are ultimately derived from class object.


Object
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .
. . .

- A subclass inherits all the *members** (**fields, methods, and nested classes**) from its superclass.

- Constructors of the superclass are not inherited by subclass, but can be invoked from the subclass.

- A subclass inherits all of the ***public*** and ***protected*** members of its parent, no matter what package the subclass is in.
- If the subclass is in the same package as its parent, it also inherits the ***package-private members*** of the parent.
- A subclass does not inherit the private members of its parent class.
- However, if the superclass has public or protected methods for accessing its private fields, these can also be used by the subclass.

Fields in subclasses

- You can use the inherited members as is, replace them, hide them, or supplement them with new members.
- The inherited fields can be used directly, just like any other fields. And can also change their values (replacing)
- You can declare a field in the subclass with the same name as the one in the superclass, thus *hiding* it (**not recommended**).
- You can declare new fields in the subclass that are not in the superclass.

Methods in subclasses

- The inherited methods can be used directly as they are.

- You can write a new *instance* method in the subclass that has the same signature as the one in the superclass, thus ***overriding*** it.

- You can declare new methods in the subclass that are not in the superclass.
- You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword **super.**

Overriding

- Inheritance1

super

- Inheritance2
- inheritance3