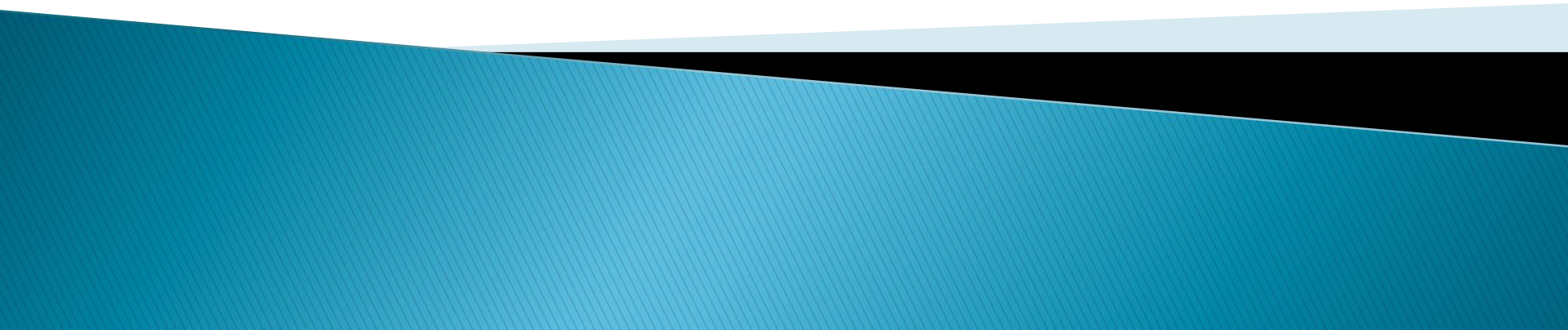



# Inheritance and Polymorphism – I

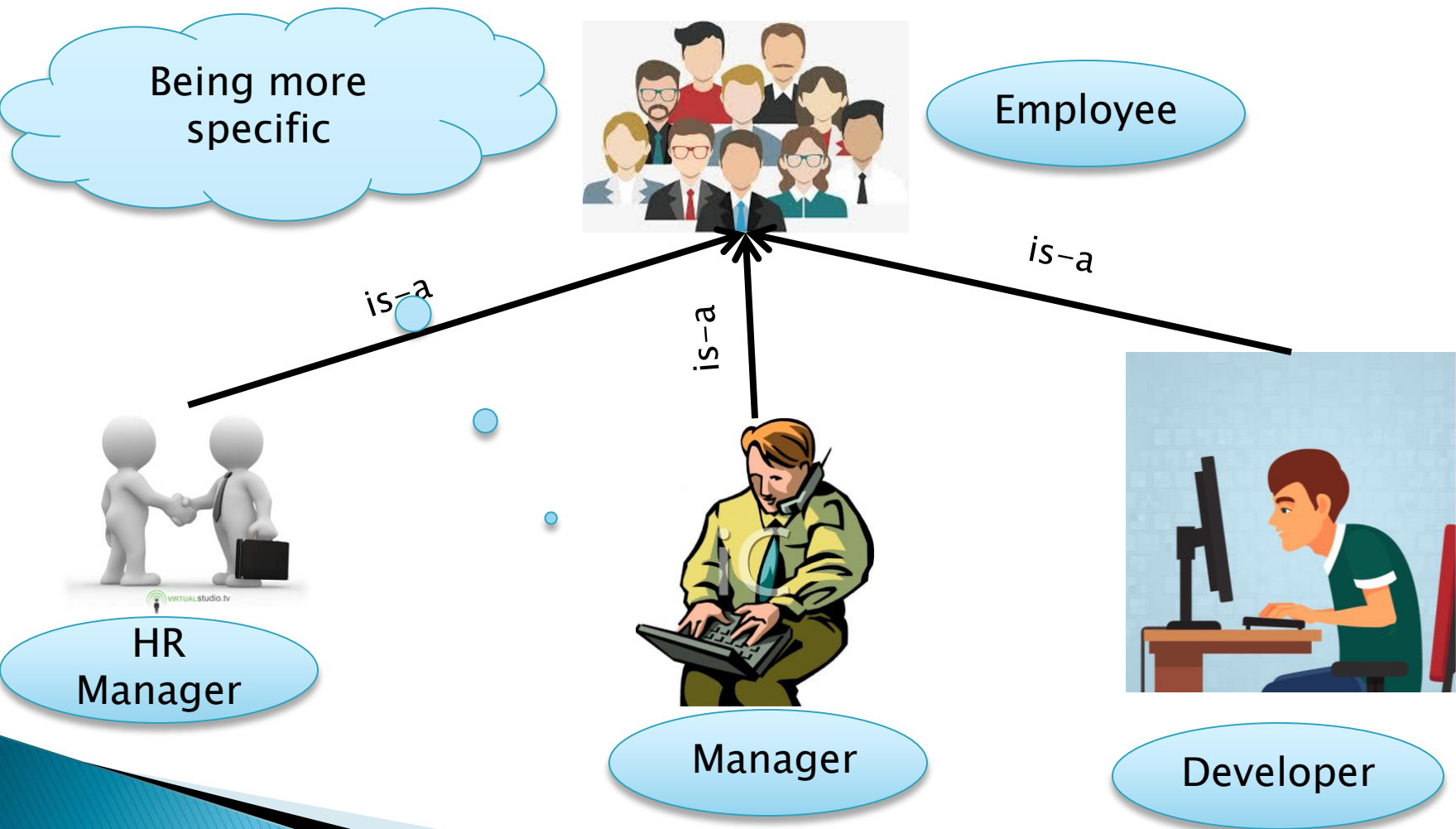
Session 4



# Contents

- ▶ Inheritance & its types.
  - ▶ Constructing “is-a” kind of relationship.
  - ▶ Using “super” & “protected” keywords.
  - ▶ Concept of Polymorphism & its use.
  - ▶ Difference between method overloading & overriding.
  - ▶ Usage of “final” keyword with method and class.
  - ▶ Concept of “Object” class with methods like toString() & equals().
- 

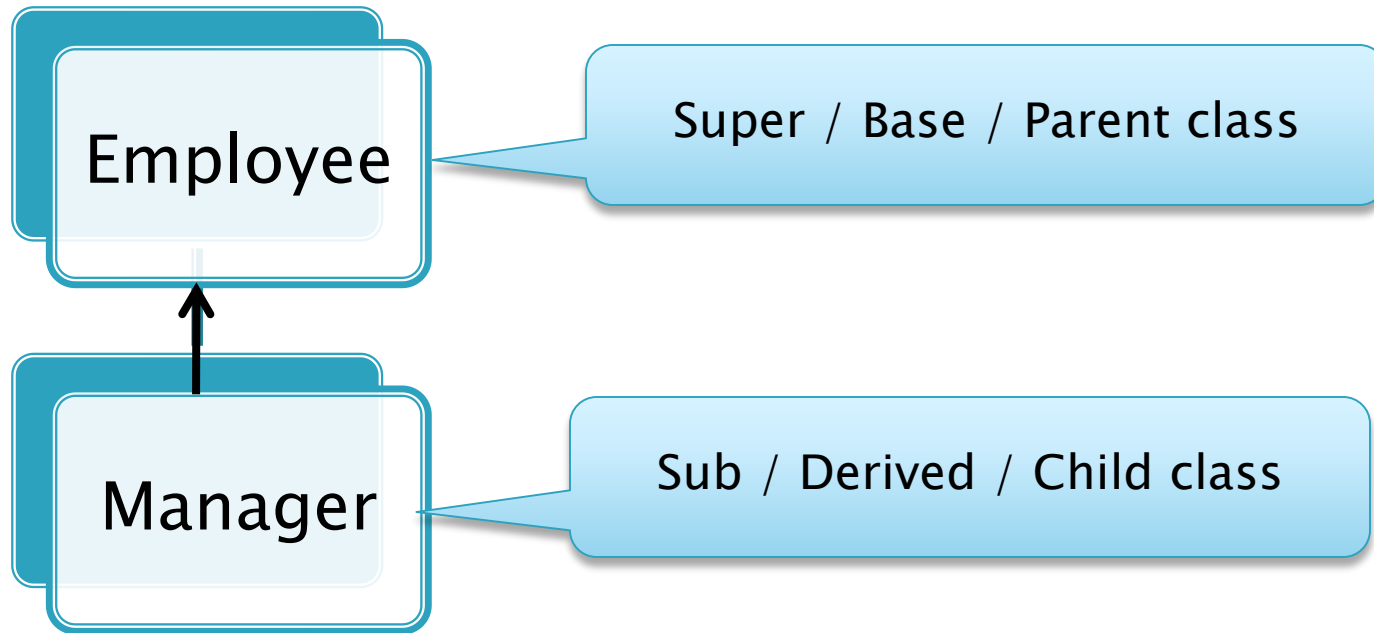
# Inheritance



# Inheritance

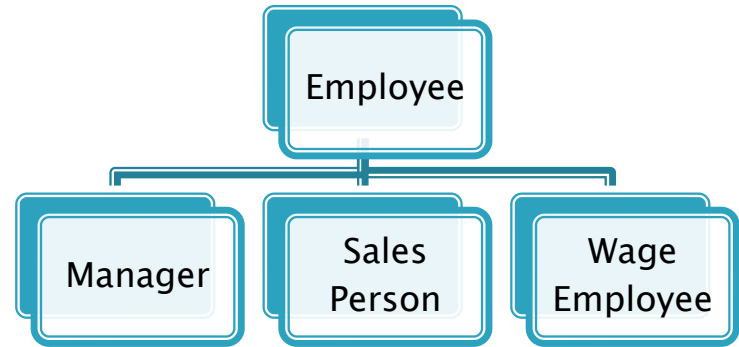
- ▶ Inheritance is one of the major pillar of Object Orientation.
- ▶ It creates “is-a” kind of relationship.
- ▶ Why to use inheritance?
  - Reusability
  - Extensibility.

# Inheritance



# Inheritance

- ▶ This is “is-a” kind of relationship
- ▶ More than one class can inherit attributes and method of single super class
- ▶ A sub class can be super class of another class



# Type of inheritance

Simple or  
Single  
inheritance

Employee

Manager

Simple or  
Single  
inheritance

Employee

Manager

Sales  
Person

Wage  
Employee

Employee

Manager

Sales  
Manager

Multilevel  
inheritance

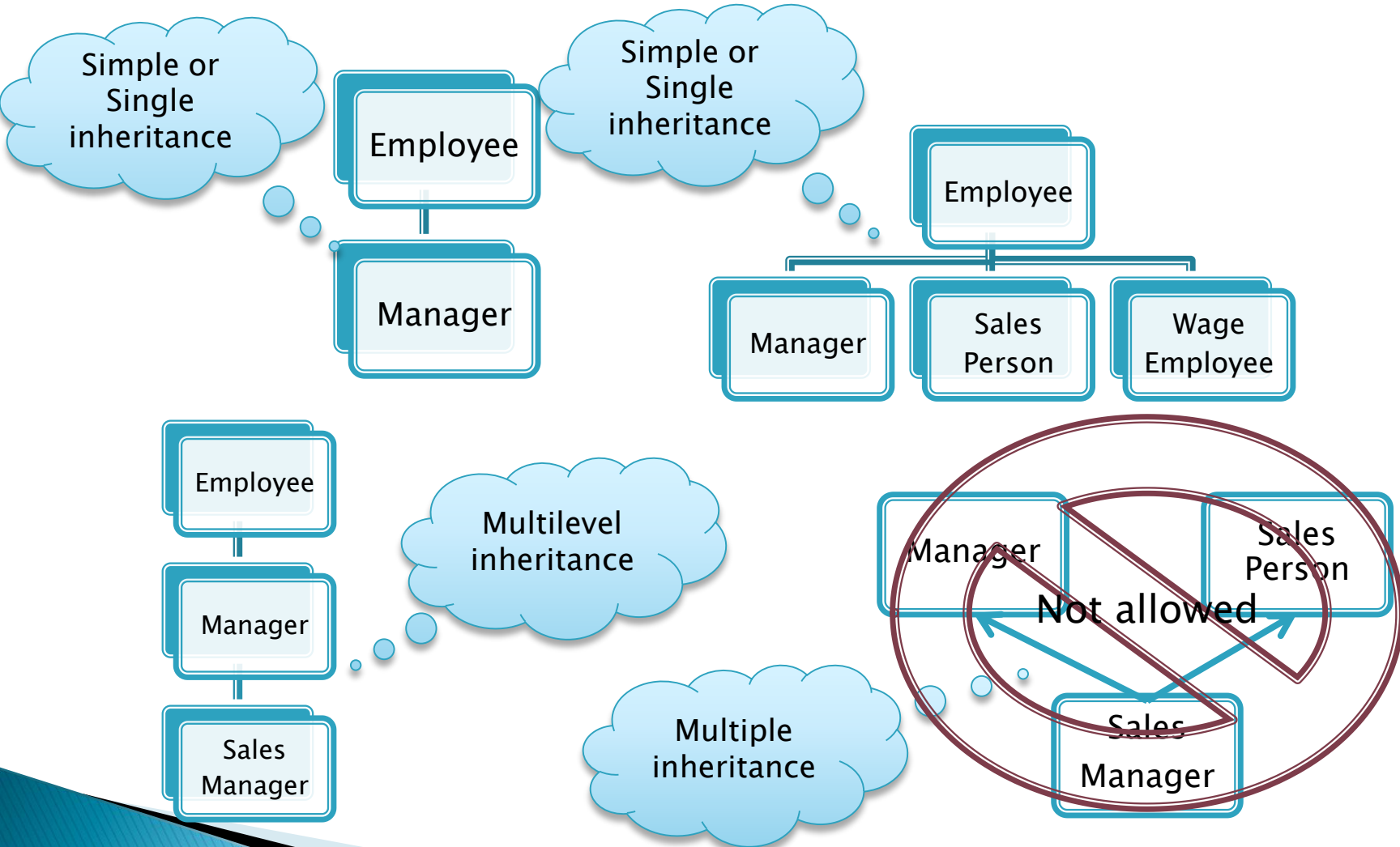
Multiple  
inheritance

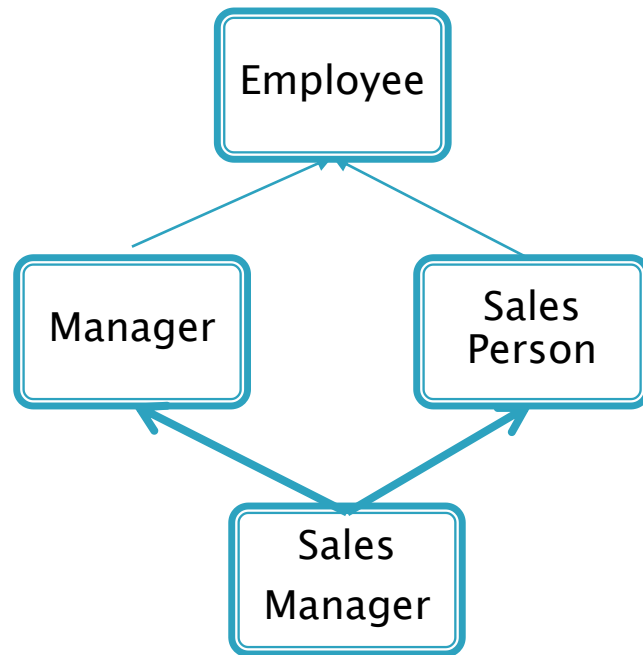
Manager

Sales  
Person

Not allowed

Sales  
Manager



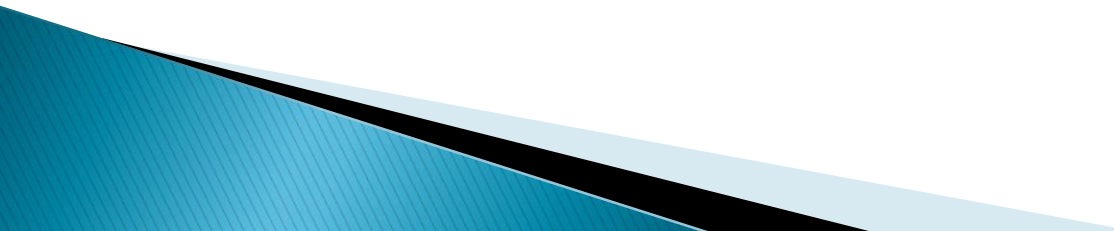




# Example of Inheritance

- ▶ Demonstrate class Employee & Manager

# Derived class constructor

- ▶ Constructors are invoked in the sequence of
  - ▶ Super → Sub
  - ▶ When Manager object is created constructor sequence will be
  - ▶ Employee → Manager
- 

# super keyword

- ▶ Calls the base class method
  - `super.methodName();`
- ▶ Calls the base class constructor
  - `super();` → Default constructor
  - `super(a, b, c)` → Parameterized constructor

# toString() & equals() methods

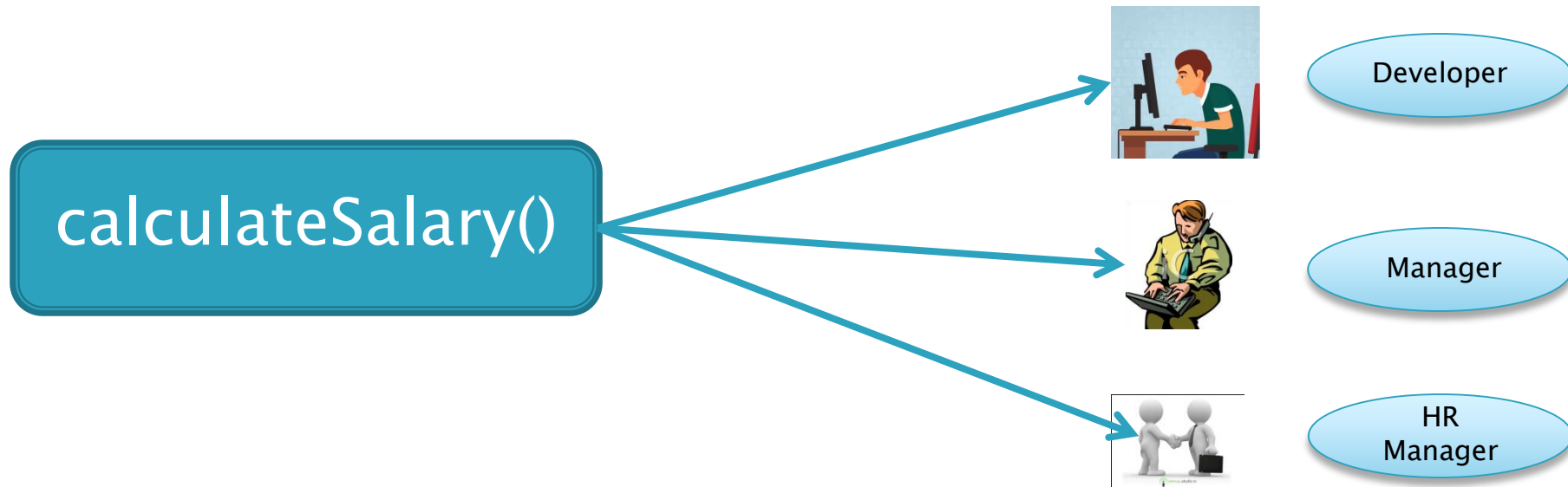
- ▶ toString() & equals() are the methods of Object class. (java.lang package.)
- ▶ *Object is cosmic super class for all datatypes in java.*

Object obj = new Employee();

- ▶ Demo of toString() & equals() method.

# Polymorphism

- ▶ It is ability of different types of related objects to respond same message (method) in their won way or different way.



# Generic Reference

- ▶ Generic reference is a reference of super class which can hold address of an object of sub class.

- ▶ eg.

Employee emp;

Manager mgr = **new Manager()**;

emp = mgr;

OR

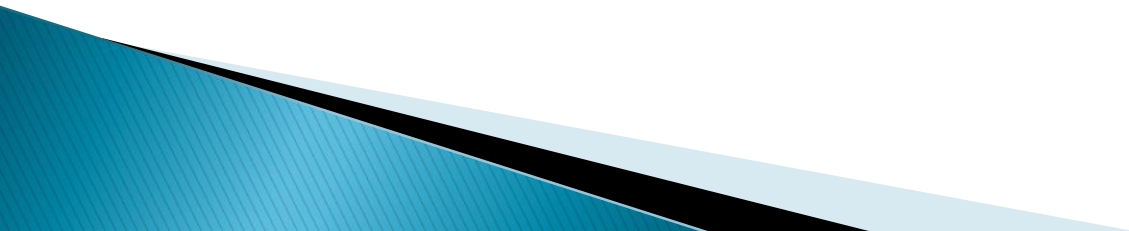
Employee emp = new Manager();

Reference of Base class / Generic Reference

Object of Derived class.

# Why polymorphism

- ▶ Polymorphism allows to design and implement system that are more easily extensible and maintainable.



# super keyword

- ▶ super keyword refers to super class.
- ▶ When one want to call any method of super class, can be called like `super.method();`.
- ▶ Calling constructor of Super class `super();`



# protected keyword

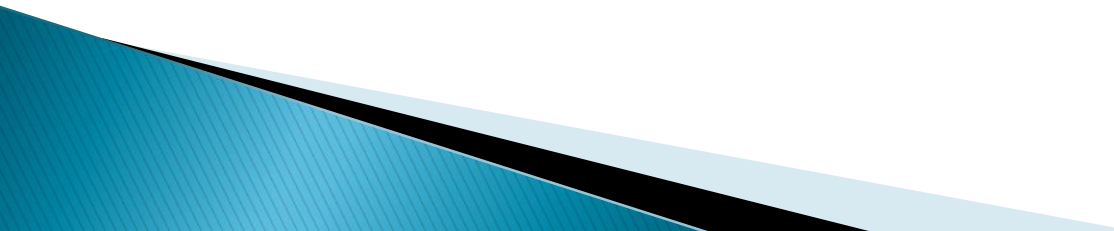
- ▶ protected keyword is used in case of inheritance.
- ▶ Accessibility of protected member is anywhere in the package and outside of package only with inherited class.

# Method overriding

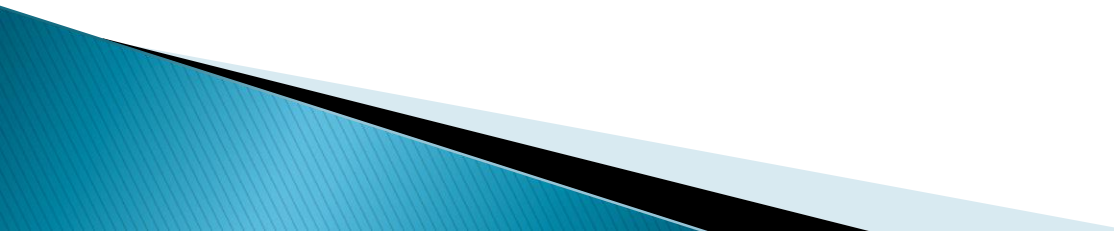
- ▶ These are the methods with same name, same signature and same **return type** where scope is different.

	Method Overloading	Method overriding
Scope	Generally in same class	In different classes (Inherited)
Purpose	Single method name can be used in different ways.	Message is same but action needs to be different.
Signature	Different for every method	Same for every method
Return type	Can be same or different as it is not considered.	Same for all methods.

# Binding

- ▶ Binding is an association with function call with an object calling the function.
  - ▶ Types of Binding
    - Compile time / Static / Early Binding: Association of function call with an object at compile time
    - Static binding or polymorphism is achieved using method overloading in java.
    - Run time / Dynamic / Late Binding: Association of call with an object at run time.
    - Dynamic binding is achieved using method overriding in java.
- 

# final keyword.

- ▶ final keyword can be used in 3 different ways.
    1. Used with variable, variable will become constant
    2. Used with class, class will become sealed. Can't inherited.
    3. Used with method, method cannot be overridden.
- 

Thank  
you!

