

# Core Java 8 and Development Tools

Lesson 06 : Inheritance and  
Polymorphism

# Lesson Objectives

- After completing this lesson, participants will be able to -
  - Understand concept of Inheritance and Polymorphism
  - Implement inheritance in java programs
  - Implement different types of polymorphism



# What is Inheritance?



**Basic TV**



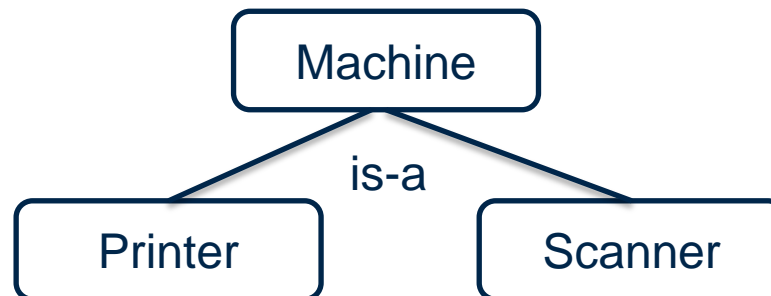
**Smart TV**



Smart TV's are inherited from basic television which apart from multimedia functionality of TV allows us to do more like streaming video contents from Internet.

# What is Inheritance?

- Inheritance allows programmers to reuse of existing classes and make them extendible either for enhancement or alteration
- Allows creation of hierarchical classification
- Advantage is reusability of the code:
  - A class, once defined and debugged, can be used to create further derived classes
- Extend existing code to adapt to different situations
- Inheritance is ideal for those classes which has “is-a” relationship
- “Object” class is the ultimate superclass in Java

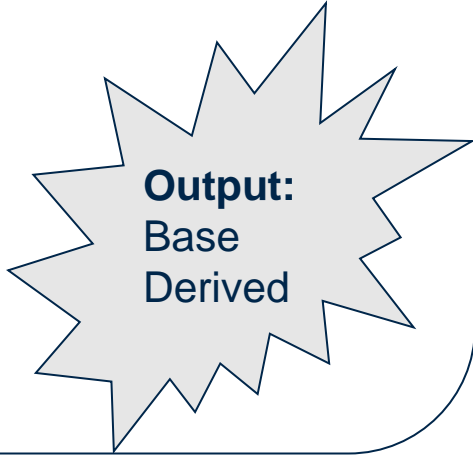


# Using super Keyword

- The super keyword is used to refer instance of its direct superclass
- There are two uses of super keyword:
  - Calling parent class constructor
  - Call member of parent class

# Inheritance : Example

```
class Base {  
    public void baseMethod() {  
        System.out.println("Base");  
    }  
}  
class Derived extends Base {  
    public void derivedMethod() {  
        super. baseMethod ();  
        System.out.println("Derived");  
    }  
}  
class Test {  
    public static void main(String args[]){  
        Derived derived=new Derived();  
        derived. derivedMethod();  
    }  
}
```



**Output:**  
Base  
Derived

# instanceOf Operator

- The instanceof operator compares an object to a specified type
- Checks whether an object is:
  - An instance of a class.
  - An instance of a subclass.
  - An instance of a class that implements a particular interface.
  - Example : The following returns true:

```
new String("Hello") instanceof String;
```

# Demo

- Inheritance
  - Basic Inheritance
  - Using super Keyword
  - Use of instanceof keyword





## What is Polymorphism?

- Poly meaning “many” and morph means “forms”
- It's capability of method to do different things based on the object used for invoking method



- Polymorphism also enables an object to determine which method implementation to invoke upon receiving a method call
- Java implements polymorphism in two ways
  - Method Overloading
  - Method Overriding

### Method Overloading

- Two or more methods within the same class share the *same* name. Parameter declarations are different
- You can overload Constructors and Normal Methods

```
class Box {  
    Box(){  
        //1. default no-argument constructor  
    }  
    Box(dbl dblValue){  
        // 2. constructor with 1 arg  
    }  
    public static void main(String[] args){  
        Box boxObj1 = new Box(); // calls constructor 1  
        Box boxObj2 = new Box(30); // calls constructor 2  
    } }
```

## Constructor Overloading

- If class has more than one constructors, then they are called as overloaded constructors.
- When constructors are overloaded, they must differ in
  - Number of parameters
  - Type of parameters
  - Order of parameters



A same model car can be constructed in different ways as per the requirement. Few of them are basic, while the other differ in fuel type, color, engine power, CNG, A.C. and or other accessories.



- In a class hierarchy, when a method in a subclass has the same *name* and *type signature* as a method in its super class, then the subclass method overrides the super class method
- Overridden methods allow Java to support run-time polymorphism



Normal Swap Machine



Chip card Machine which **overrides** the card reading for better security

# Demo

- Polymorphism
  - Method Overloading
  - Method Overriding



# @Override Annotation

- The @Override annotation informs the compiler that the element is meant to override an element declared in a superclass
- It applies to only methods

```
public class Employee {  
    @override  
    public String toString() {  
        //statements  
        return "EmpName is:"+empname;  
    }  
}
```

- Above code will throw a compilation error as it is not overriding the toString method of Object Class

# Final Modifier

- Final Modifier : Can be applied to variables, methods and classes
- Final variable:
  - Behaves like a constant; i.e. once initialized, it's value cannot be changed
  - Example: `final int i = 10;`
- Final Method:
  - Method declared as final cannot be overridden in subclasses
  - Their values cannot change their value once initialized
  - Example:

```
class A {  
    public final int add (int a, int b) { return a+b; }  
}
```
- Final class:
  - Cannot be sub-classed at all
  - Examples: *String* and *StringBuffer* class



A class or method cannot be abstract & final at the same time.

# Lab

- Lab 2: Inheritance and Polymorphism





# Summary

- In this lesson, you have learnt about:
  - Inheritance
  - Using super keyword
  - InstanceOf Operator
  - Method & Constructor overloading
  - Method overriding
  - @override annotation
  - Using final keyword
  - Best Practices



# Review Question

- Question 1: Which of the following options enable parent class to avoid overriding of its methods.
  - extends
  - Override
  - Final
- Question 2: When you want to invoke parent class method from child, it should be written as first statement in child class method
  - True/False
- Question 3: Which of the following access specifier enables child class residing in different package to access parent class methods?
  - private
  - public
  - Final
  - Protected

