INHERITANCE

Presented by:

Dr. Shivangi K. Surati

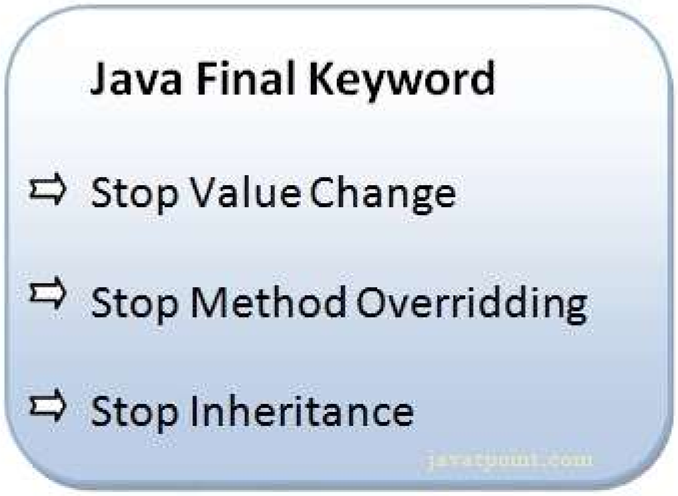
Assistant Professor,
Department of Computer Science and Engineering,
School of Technology,
Pandit Deendayal Energy University

Outline

- Final Keyword
- Inheritance
- Why Inheritance?
- Inheritance Example
- Types of Inheritance
- Using Super- two uses
- Method Overriding
- Access modifiers
- Abstract class
- Using Final with Inheritance
- Dynamic Method Dispach

Final

- Used to restrict the user.
- Final can be:
  - variable
  - method
  - Class



Java Final Keyword

⇨ Stop Value Change

⇨ Stop Method Overridding

⇨ Stop Inheritance

- If a field declared as final, the copy constructor can change it.

Final variable

- The value of final variable cannot be changed
- Initialize it when it is declared

EX:

```
class carSpeed{
        final int speed=70;   //final variable
        void changeSpeed(){
                speed=100; //compile time error
        }
        public static void main(String args[]){
                carSpeed carObj = new carSpeed();
                carObj.changeSpeed();
 }
}
```

Inheritance

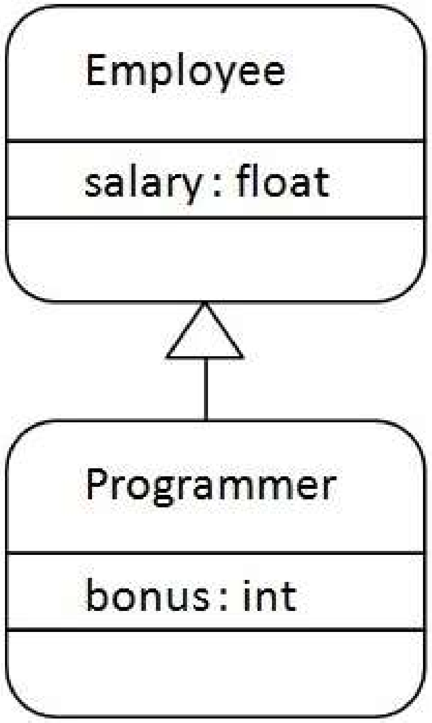
- A mechanism in which one object acquires all the properties and behaviors of a parent object
- Important feature of OOPs
- Idea: create new classes that are built upon existing classes
- Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Why Inheritance?

- Reusability: to reuse methods and fields of the parent class
- Method overriding (Run-time polymorphism)

- Define superclass- general aspects of an object
- Inherit superclass to form specialized classes
- Each subclass simply adds its own attributes

Inheritance Example



- Relationship: Programmer IS-A Employee

Simple Program

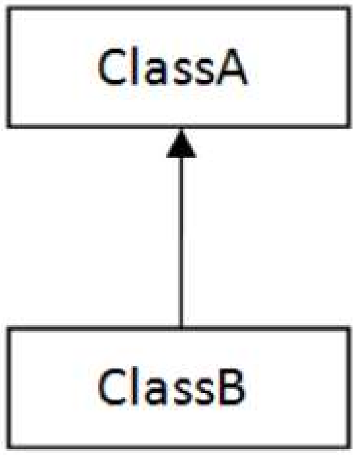
Inheritance program.doc (**Simple Example**)

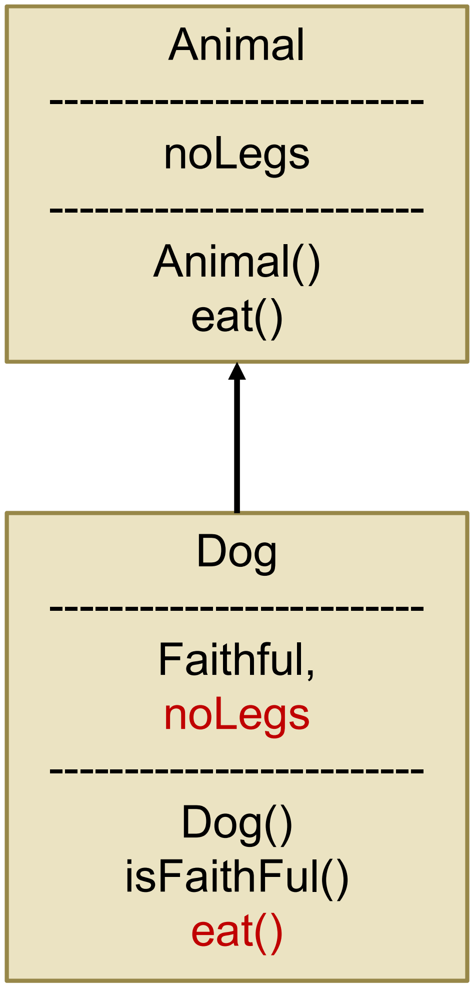
(Refer this file for types of inheritance also).

Types of Inheritance

1) Single Inheritance

Inheritance program.doc
(**Single Inheritance**)



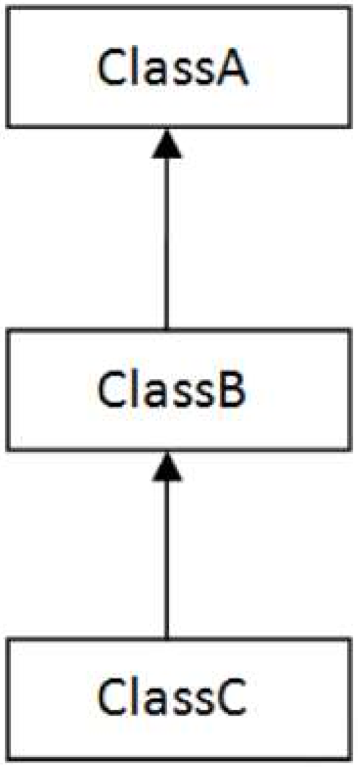


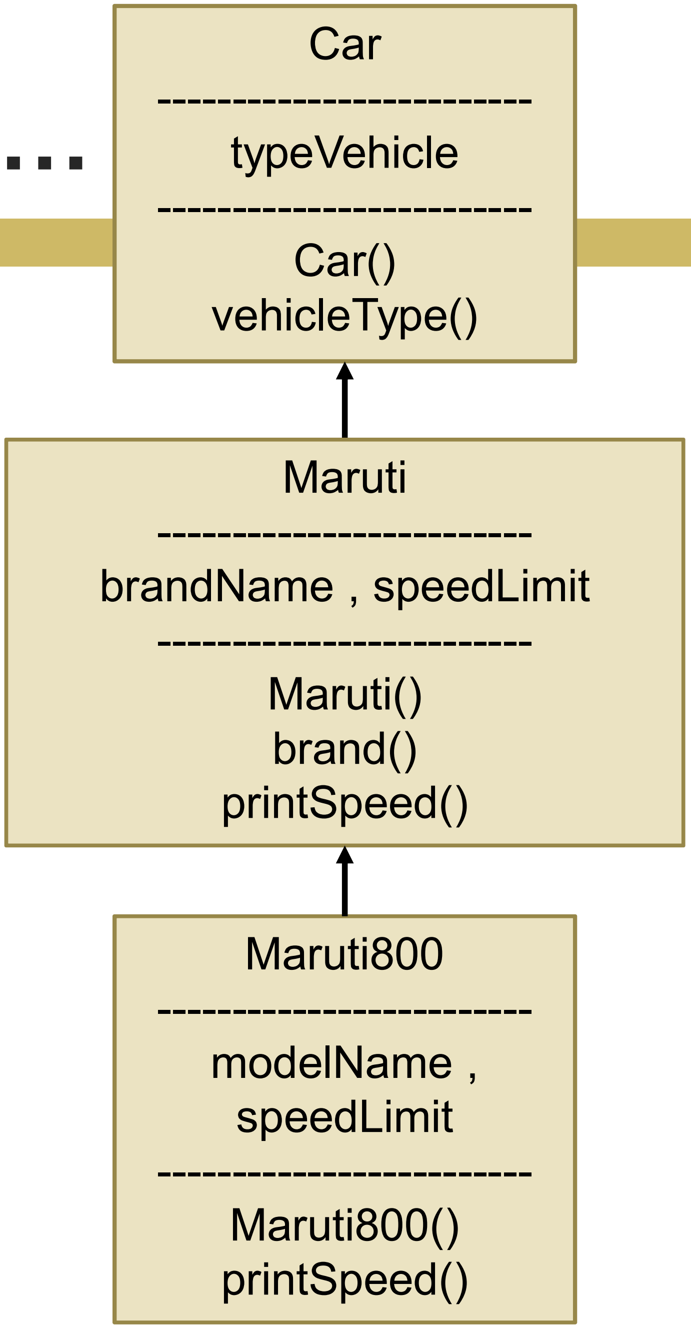
Types of Inheritance…

2) Multilevel Inheritance

Inheritance program.doc

(Multilevel Inheritance)



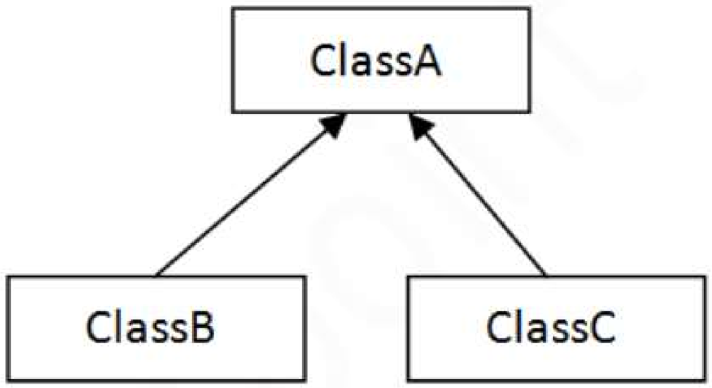


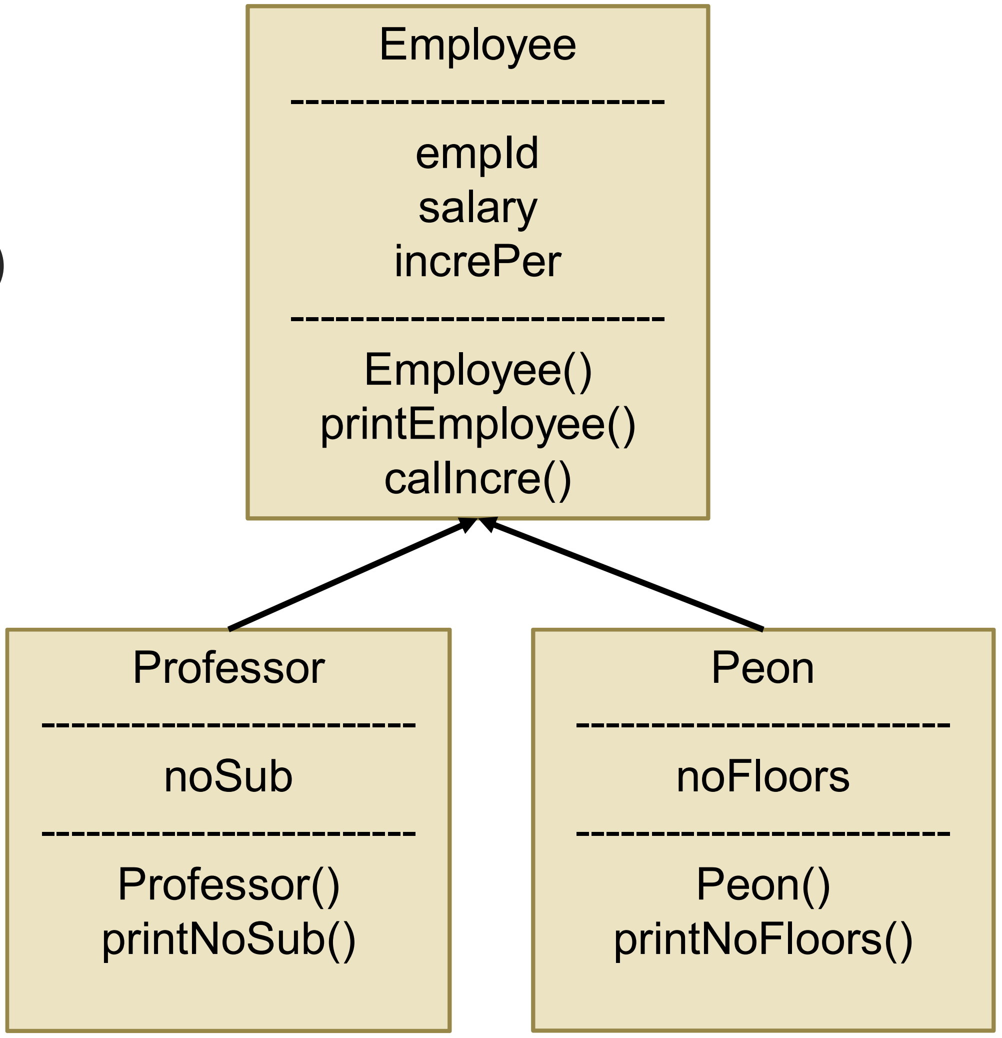
Types of Inheritance…

3) Hierarchical Inheritance

Inheritance program.doc

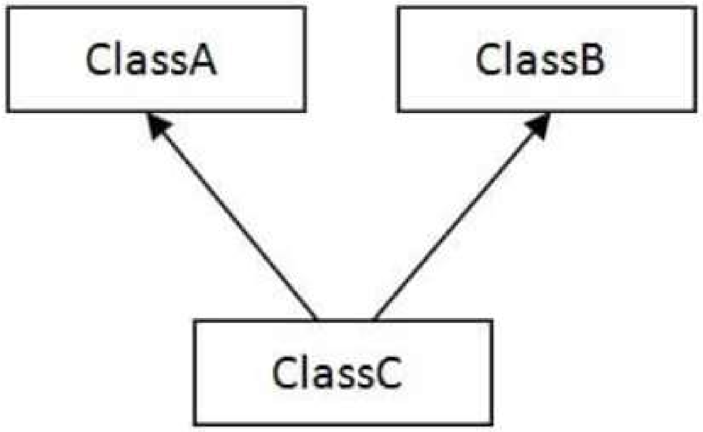
(<u>**Hierarchical Inheritance**</u>)



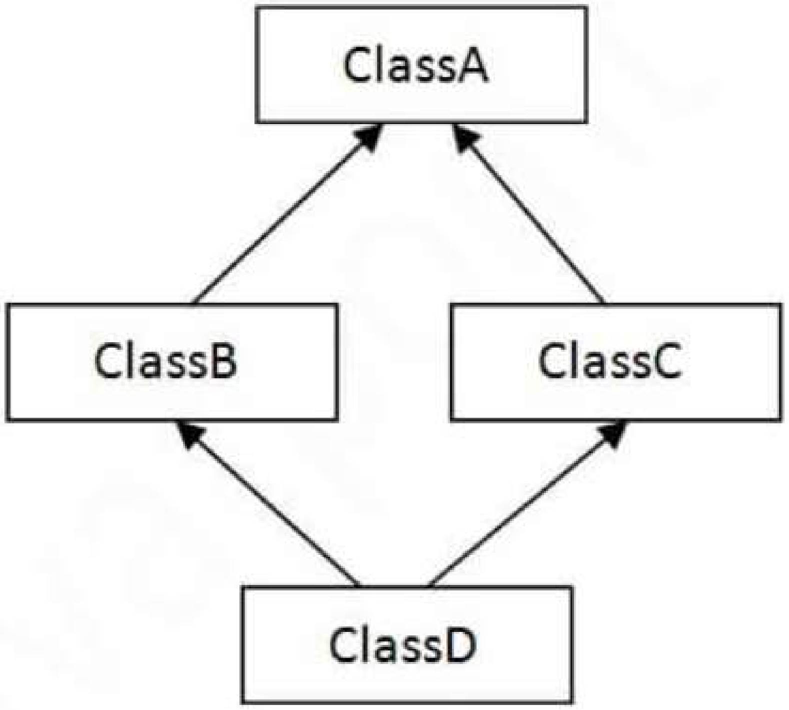


Types of Inheritance…

4) Multiple Inheritance



5) Hybrid Inheritance



These two inheritances are not supported by Java Classes !!

Using Super- two uses

(1) To call Superclass Constructors

- Syntax: super(arg-list);

- From previous programs:

Inheritance program.doc (**Hierarchical Inheritance**, Prog-3, Prog-5)

Using Super- two uses…

(2) To refer superclass member (instance variable/method)

- Syntax: super.member;

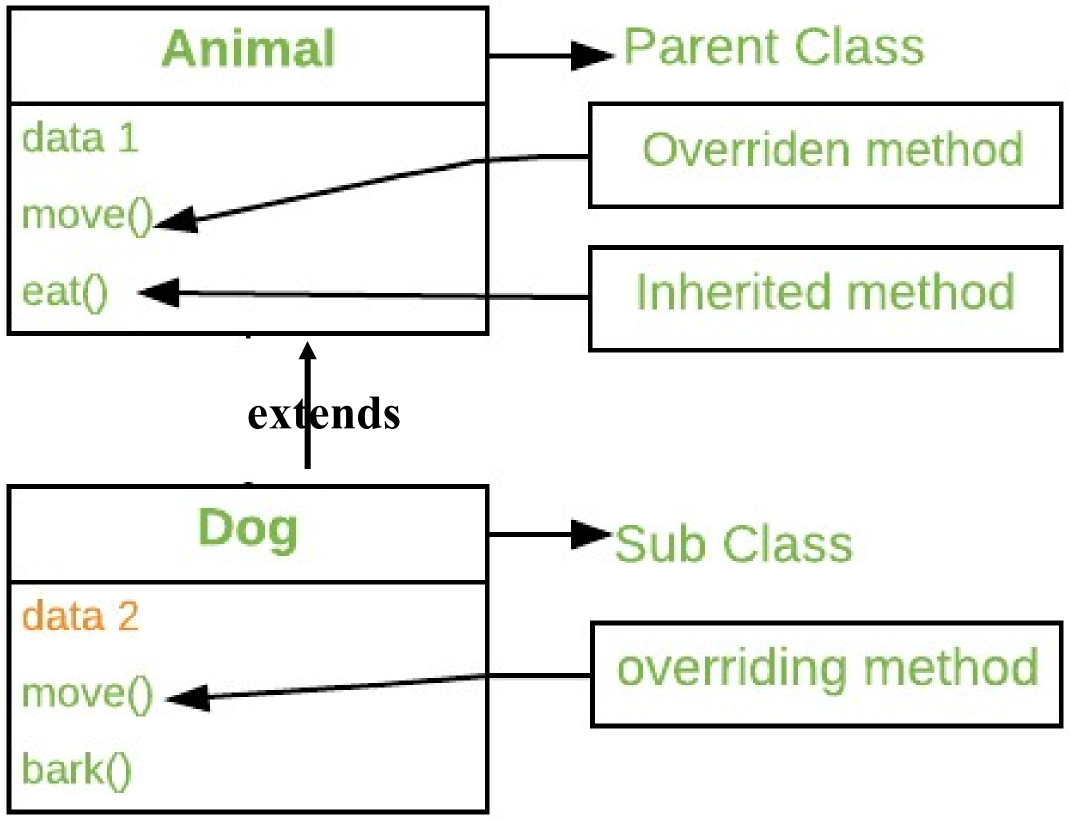
- Used when subclass members hide members by the same name in the superclass
- Inheritance program.doc (Prog-6)

Method Overriding

- If subclass (child class) has the same method as declared in the parent class
- Rules for Java Method Overriding
  - Applicable in Inheritance (IS-A relationship)
  - The method must have the same name as in the parent class
  - The method must have the same parameter as in the parent class.

Method Overriding…

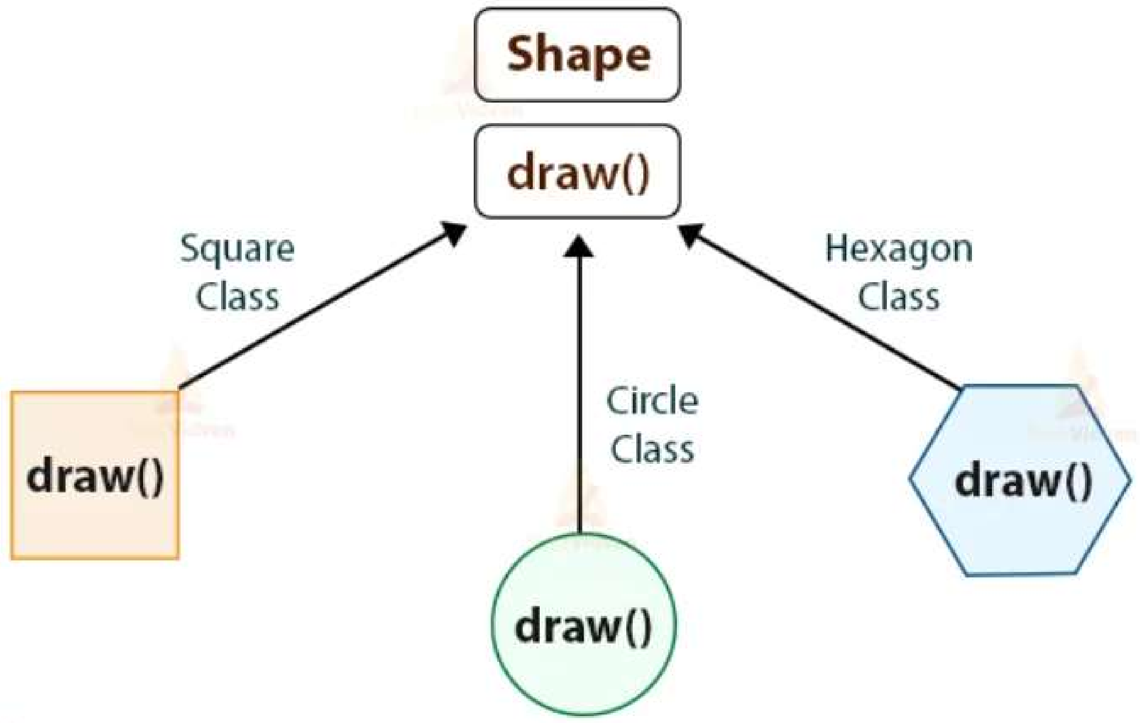
(1) Ex-1:



(2) Inheritance programs-1.doc

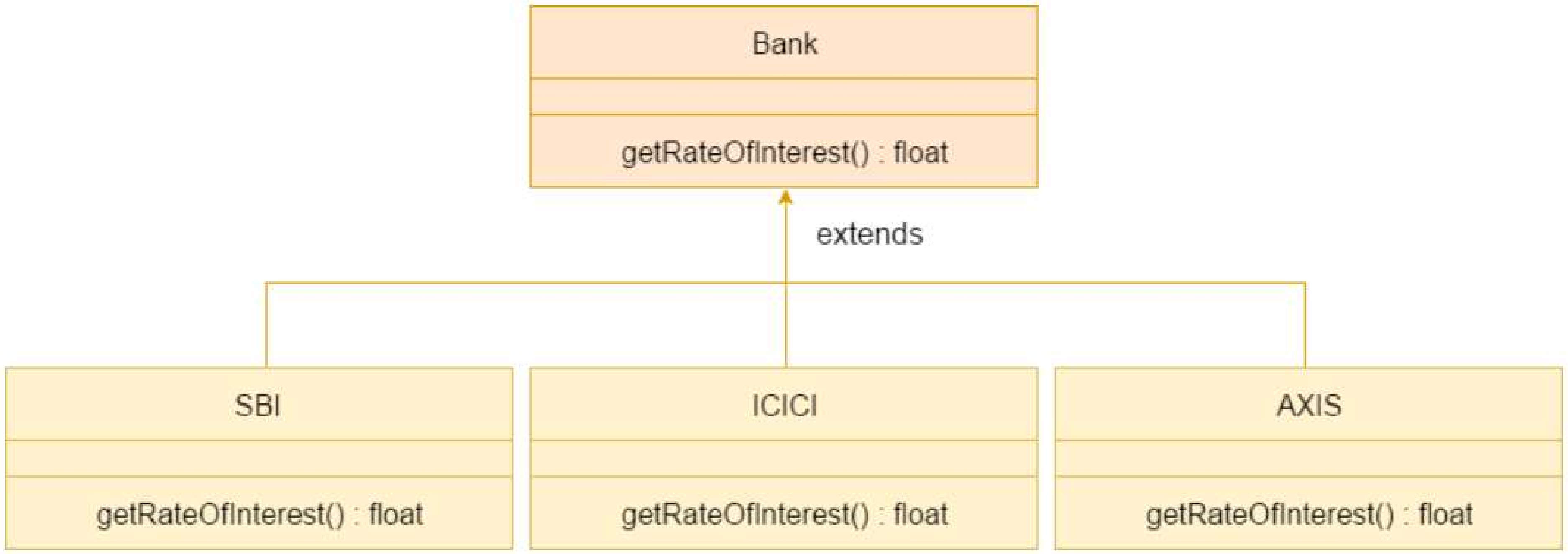
Method Overriding…

- Usage of Java Method Overriding
  - Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
  - Method overriding is used for runtime polymorphism



Method Overriding…

- Try yourself !!
  - Add the method noLockersAvail() and override it in subclasses.


Bank
getRateOfInterest() : float
extends
SBI
getRateOfInterest() : float
ICICI
getRateOfInterest() : float
AXIS
getRateOfInterest() : float

Method Overriding…

- Rules:
  - Static methods can not be overridden
  - The methods declared as ‘final’ cannot be overridden
  - Constructors cannot be overridden
  - Overriding Method must have the same return type (or subtype)
  - If lesser access in the subclass than that in the superclass, then we will get a compile-time error

Access Modifiers

- To restrict the scope of a class, constructor, variable, method, or data member
- Four types of access modifiers in Java:
  - Default – No keyword required
  - Private
  - Protected
  - Public

Access Modifiers…

- Access Level of each modifier:

Access Modifier	Access Level	Cannot be accessed from
Default	Only within the package	Outside the package
Private	Only within the class	Outside the class
Protected	Within the package and outside the package through child class	Outside the package without child class
Public	Everywhere (within the class, outside the class, within the package and outside the package	-

Access Modifiers…

Access Modifier	Within class	Within package subclass	Within package Non subclass	Outside package by subclass only	Outside package Non subclass
Private	Yes	No	No	No	No
Default	Yes	Yes	Yes	No	No
Protected	Yes	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes	Yes

Inheritance programs-1.doc

Find outputs

- Inheritance programs-1.doc
- Ex-3 and Ex-4 shows use of final with Inheritance

Abstract Class

- Abstraction
  - a process of hiding the implementation details and showing only functionality to the user
  - Focus on
    - What the object does
    - Not how it does
- Define a superclass that declares the structure of given abstraction
- Superclass only defines a generalized form shared by all subclasses- subclass will fill in the details
- Determines nature of methods that subclass must implement
- Superclass has no meaningful instructions

Abstract Class…

- A class that is declared abstract
  - may or may not include abstract methods
  - must be declared with an abstract keyword
- Abstract classes cannot be instantiated
- Abstract classes can be subclassed
  - the subclass usually provides implementations for all of the abstract methods in its parent class
- It can have abstract and non-abstract methods
- It can have constructors and static methods also
- It can have final methods

Abstract Class…

- Syntax of abstract class

  abstract class A{}

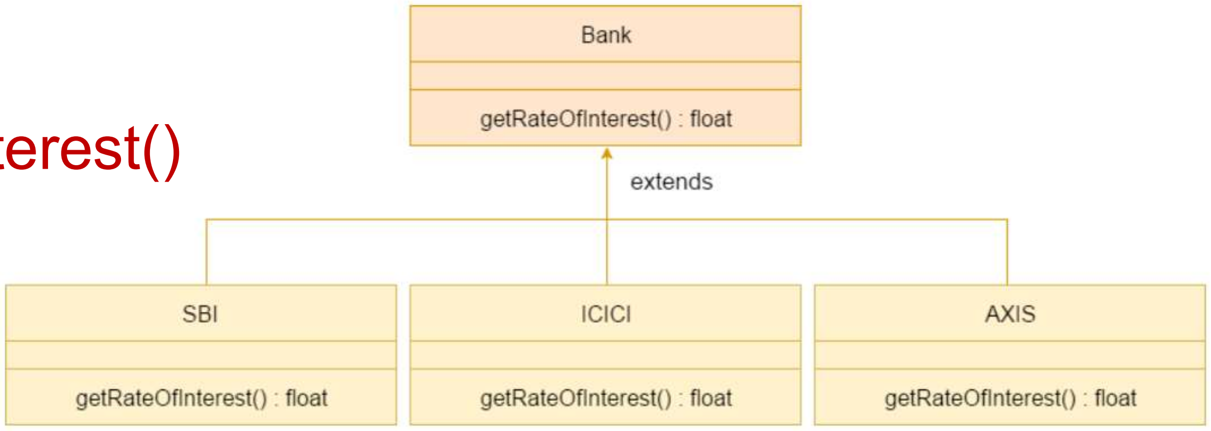
- Syntax of abstract method

  abstract void printData(); //no method body and abstract

- Rule:
  - If there is an abstract method in a class, that class must be abstract

Abstract Class Example

```
abstract class Bank{
        abstract float getRateOfInterest();
}
class SBI extends Bank{
        float getRateOfInterest()
                {return 7;}
}
class TestBank{
public static void main(String args[]){
      SBI b=new SBI();
      System.out.println("Rate of Interest is:+ b.getRateOfInterest());
      Bank ob=new Bank();  //?
}}
```


Bank
getRateOfInterest() : float
extends
SBI
getRateOfInterest() : float
ICICI
getRateOfInterest() : float
AXIS
getRateOfInterest() : float

Dynamic Method Dispatch

- One of the powerful concepts of Java
- Achieved through method overriding
- A call to an overridden method is resolved at run time
- Implements run time polymorphism
- Superclass reference can refer to a subclass object
- Determines which version of that method to execute based upon
  - **The type of the object being referred to during call**

Example

- Inheritance programs-1.doc

Questions

- Difference between method Overloading and Method Overriding in java?
- What if constructor is made private?