Structured Programming

Lecture5

Dr. Obead Alhadreti

Outline

- Inheritance
- extends Keyword
- Types of Inheritance
- IS-A Relationship
- super keyword

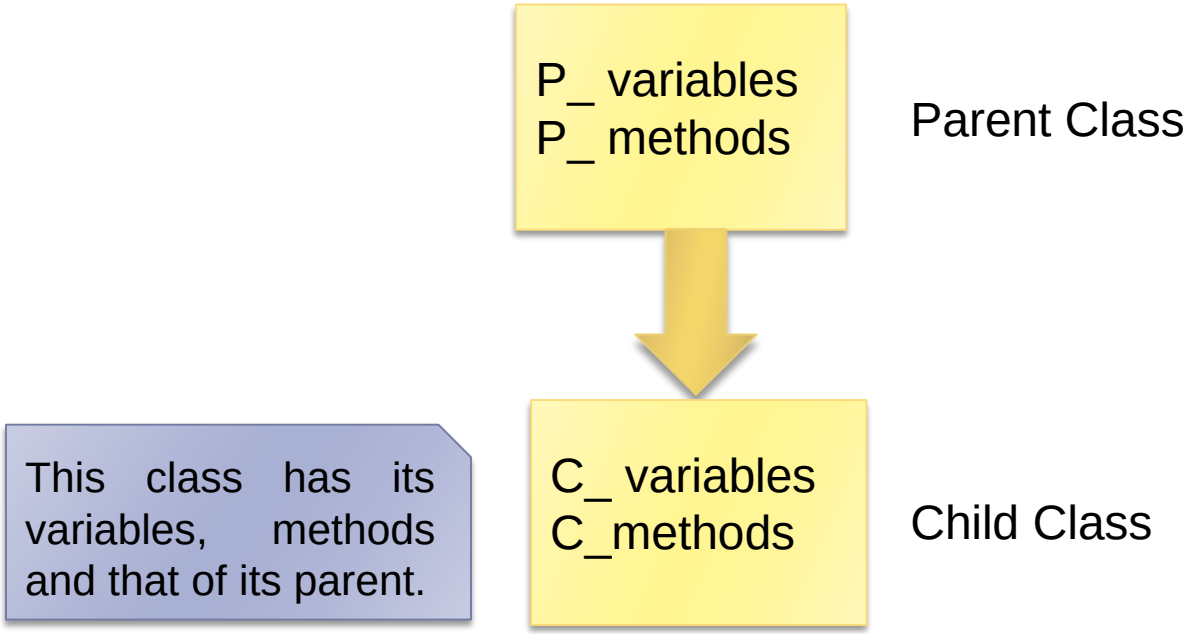
Inheritance

- One of the advantages of an Object-Oriented programming language is code reuse. We can do code reuse in java by the vimplementation of inheritance.

- **Inheritance** can be defined as the process where a new class acquires the members (methods and variables) of an existing class, and possibly embellishing them with new or modified capabilities.

Inheritance

- The new class which inherits the members of an existing class is known as subclass (derived class, child class) and the class whose members are inherited is known as superclass (base class, parent class).



Advantages of Inheritance

1. Minimizing the amount of duplicate code.

2. A better organization of code and smaller, simpler compilation units.

3. Making application code more flexible to change because classes that inherit from a common superclass can be used interchangeably.

4. Increasing the likelihood that a system will be implemented and maintained effectively.

extends Keyword

- **extends** is the keyword used to inherit the members of a class.

- To achieve inheritance, we put the keyword extends after the name of a child class, then we put the name of a parent class.

```
Class Product {
int x;
}

class Television extends Product {
}
```

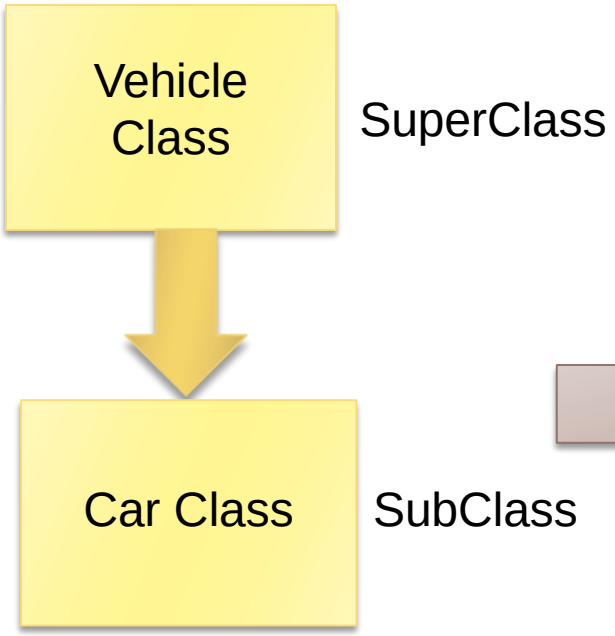
  - Class Television has a copy of the variable x of the class Product.

Superclasses and Subclasses

- A subclass inherits all the members (fields, methods, and nested classes) from its superclass.
- A subclass can add its own variables and methods, or override existing behavior from superclass.
- If a class try to inherit from a non-existing class, you will get an error message java.lang.ExceptionInInitializerError
- A subclass can be a superclass of **future** subclasses.

Superclasses and Subclasses

- An object of a subclass can be treated as an object of its superclass. That is why, using the object of the subclass you can access the members of a superclass.

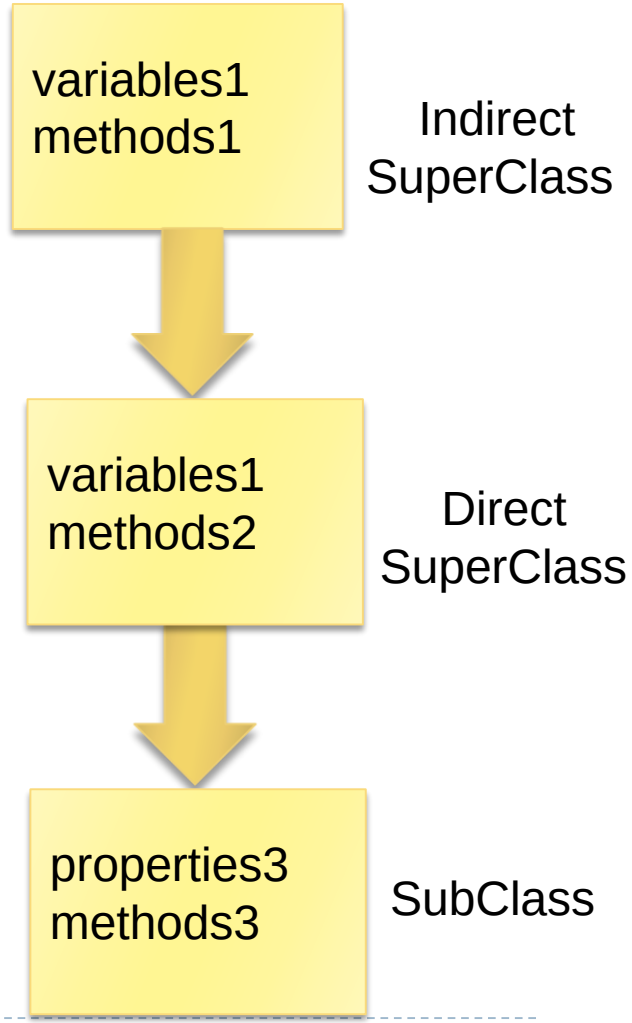


Honda **is a** car &
Honda is a vehicle

Object of sub is also an object of super

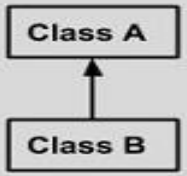
Direct & Indirect Superclasses

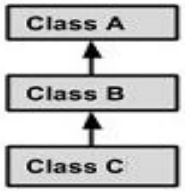
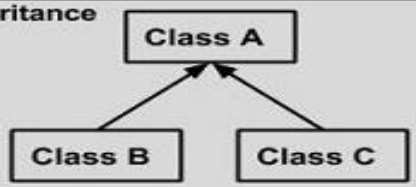
- The direct superclass is the superclass from which the subclass explicitly inherits.
- An indirect superclass is any class above the direct superclass in the class hierarchy.

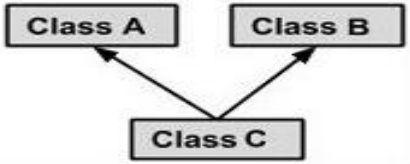


Types of Inheritance

- There are various types of inheritance as demonstrated below.

Single Inheritance 	public class A {
	.....
	}
	public class B **extends** A {
	.....
	}

Multi Level Inheritance 	public class A { .....} public class B **extends** A {.....} public class C **extends** B {..... }
Hierarchical Inheritance 	public class A { .....} public class B **extends** A {.....} public class C **extends** A {..... }

Multiple Inheritance 	public class A { .....}
	public class B {.....}
	public class C **extends** A,B {
	.....
	} // Java does not support mutiple Inheritance

Example

CollegeRecords.java × | **Employee.java** × | Academic.java ×

Source | History

```java
package collegerecords;

public class Employee {

    public String name;
    public int x;

    public void diplay ()
{
    System.out.println ("I am working at the college of computer");
}


}
```

CollegeRecords.java × | **Employee.java** × | **Academic.java** ×

Source | History

```java
package collegerecords;

public class  Academic extends Employee {

}
```

Example

CollegeRecords.java | Employee.java | Academic.java

Source | History

```java
package collegerecords;

public class CollegeRecords {

    public static void main(String[] args) {

        Academic a1 = new Academic ();

        a1.diplay();

        a1.x = 12;

        System.out.println("x: " +a1.x);
    }

}
```

Notifications | Output - CollegeRecords (run)

```
run:
I am working at the college of computer
x: 12
BUILD SUCCESSFUL (total time: 0 seconds)
```

IS-A Relationship

- IS-A is a way of understating the relationship between objects. It is a way of saying: This class (and its object) is a type of that class (and its objects).

- Let us see the following example:

```
public class Animal {
}

public class Mammal extends Animal {
}

public class Reptile extends Animal {
}

public class Dog extends Mammal {
}
```

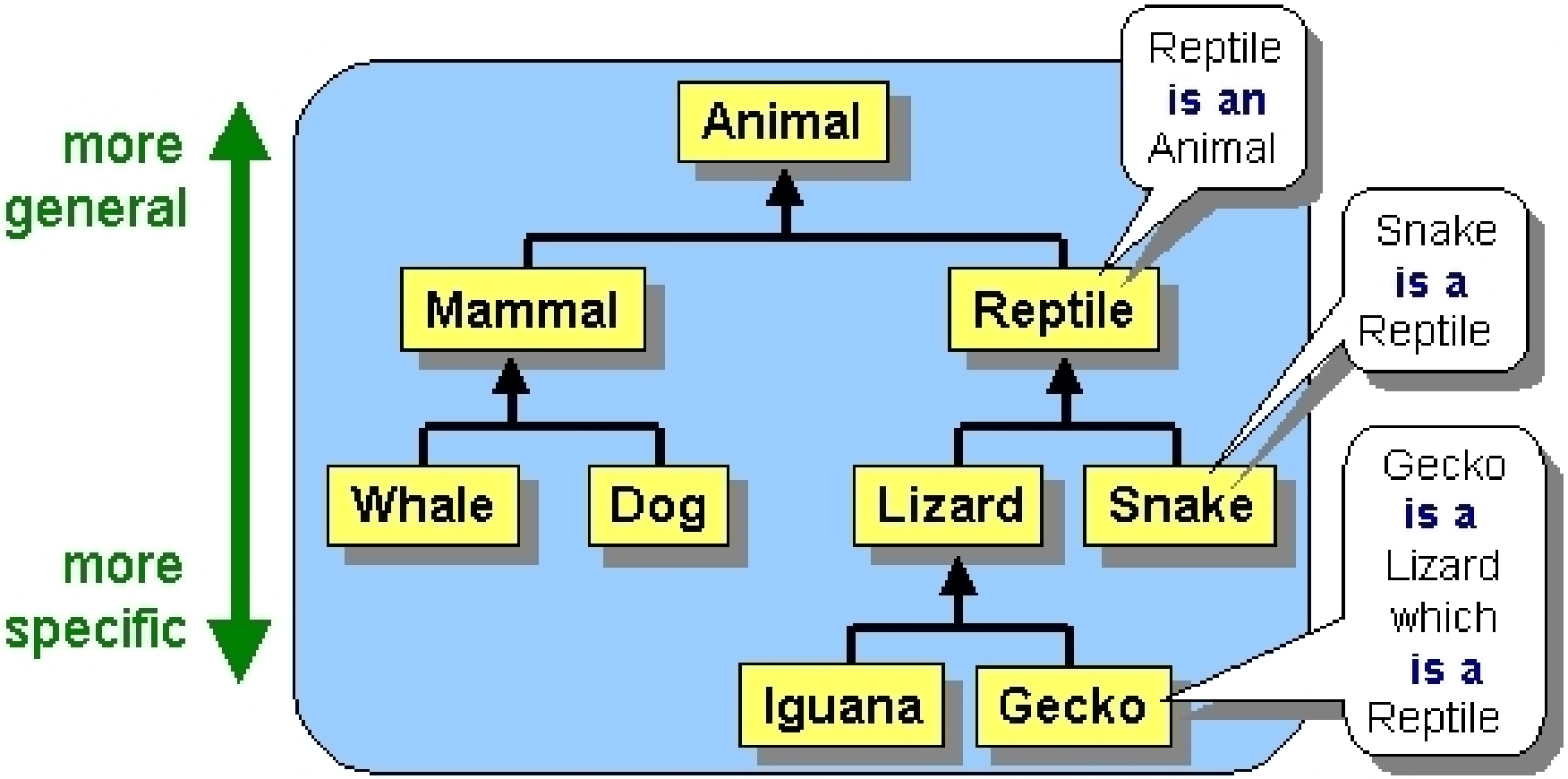
IS-A Relationship (cont't)

- Now, based on the above example, in Object-Oriented terms, the following are true :
  - Animal is the superclass of Mammal class.
  - Animal is the superclass of Reptile class.
  - Mammal and Reptile are subclasses of Animal class.
  - Dog is the subclass of both Mammal and Animal classes.

IS-A Relationship (cont't)

- Now, if we consider the IS-A relationship, we can say :
  - Mammal IS-A Animal
  - Reptile IS-A Animal
  - Dog IS-A Mammal
  - Hence: Dog IS-A Animal as well

IS-A Relationship (cont't)



super keyword

- Following are the scenarios where the **super** keyword is used :

  1. It is used to **differentiate the members** of superclass from the members of subclass, if they have same names.

  2. It is used to **invoke the superclass** constructor from subclass.

1. Differentiating the Members

- If a class is inheriting the members of another class. And if the members of the superclass have the names same as the sub class, to differentiate these members we use super keyword as shown below:

```
super.variableName

super.methodName();
```

Example 1

CollegeRecords.java | Employee.java | Academic.java

ource History

```
package collegerecords;

public class Employee {
    public int x = 5;

    }
```

CollegeRecords.java | Employee.java | Academic.java

urce History

```
package collegerecords;

public class  Academic extends Employee {
    public int x = 20;

    public void display() {
        System.out.println("x in Academic contain: " +x);
        System.out.println("x in Academic contain: " +this.x);
        System.out.println("x in Employee contain: " +super.x);
    }
}
```

Example 1

CollegeRecords.java | Academic.java | Employee.java

Source | History

```java
package collegerecords;

public class CollegeRecords {

    public static void main(String[] args) {

        Academic a1 = new Academic ();

        a1.display();

    }

}
```

Notifications | Output - CollegeRecords (run)

```
run:
x in Academic contain: 20
x in Academic contain: 20
x in Employee contain: 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Example 2

CollegeRecords.java × | Employee.java × | Academic.java ×

ource | History

```java
package collegerecords;

public class Employee {
 public void print() {
        System.out.println("This is print() method from the class Employee");
    }

}
```

CollegeRecords.java × | Employee.java × | Academic.java ×

Source | History

```java
package collegerecords;

public class  Academic extends Employee {

    @Override
    public void print() {
        System.out.println("This is print() method from the class Academic");
    }

    public void printBoth() {
        print();
        this.print();
        super.print();
}

}
```

Example 2

CollegeRecords.java | Employee.java | Academic.java

Source | History

```java
package collegerecords;

public class CollegeRecords {

    public static void main(String[] args) {

        Academic a1 = new Academic ();

        a1.printBoth();

    }

}
```

Notifications | Output - CollegeRecords (run)

```
run:
This is print() method from the class Academic
This is print() method from the class Academic
This is print() method from the class Employee
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Invoking Superclass Constructor

- If a class is inheriting the members of another class, the subclass automatically acquires the default constructor of the superclass.

- But if you want to call an empty or a parameterized constructor of the superclass, you need to use the super keyword as shown below.

```
super()    // to call an empty constructor

super( parameter List ) // to call a parameterized constructor
```

Example

Inheritance.java | Super_class.java | Sub_class.java

Source | History

```java
package inheritance;

public class Super_class {

    int age;

    public Super_class (int age)
    {
        this.age = age;
    }

    void printInfo () {
        System.out.println ("Age:" +age);
    }

}
```

Inheritance.java | Super_class.java | Sub_class.java

Source | History

```java
package inheritance;

public class Sub_class extends Super_class {

    public Sub_class (int age) {
        super (age);
    }

}
```

Example

Inheritance.java × | Super_class.java × | Sub_class.java ×

Source | History

```java
package inheritance;

public class Inheritance {

    public static void main(String[] args) {
        Sub_class obj = new Sub_class(24);

        obj.printInfo();
    }

}
```

Notifications | Output - Inheritance (run) ×

```
run:
Age:24
BUILD SUCCESSFUL (total time: 0 seconds)
```

Assignment Three is Uploaded to the Portal of “Learning”