

UNIT III – INHERITANCE AND POLYMORPHISM

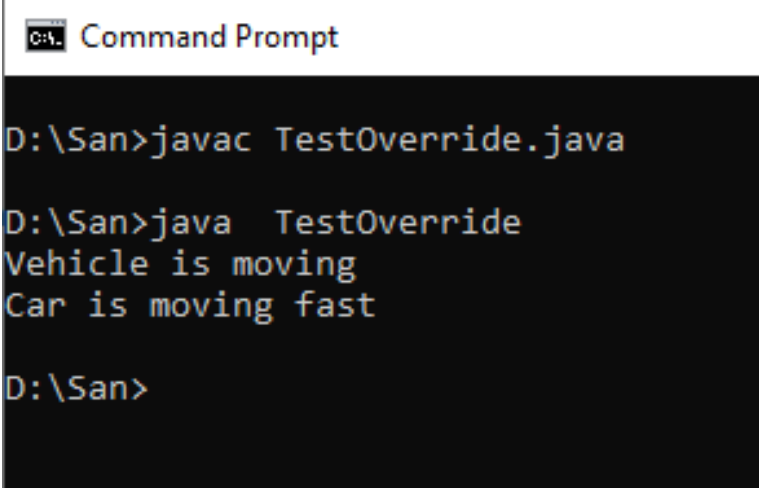
TYPES OF INHERITANCE IN JAVA (SINGLE, MULTILEVEL, HIERARCHICAL):

METHOD OVERRIDING:

- Occurs when a subclass provides its own specific implementation for a method that is already defined in its superclass.
- To be considered an override, the method in the subclass must have:
 - ✓ The exact same method name as in the superclass.
 - ✓ The exact same number, type, and order of parameters.
 - ✓ A compatible return type (same or a subtype of the superclass method's return type).

```
public class TestOverride
{
    public static void main(String[] args)
    {
        Vehicle v = new Vehicle();
        v.move(); // Output: Vehicle is moving
        Car c = new Car();
        c.move(); // Output: Car is moving fast
    }
}
```

Output:



```
Command Prompt

D:\San>javac TestOverride.java

D:\San>java TestOverride
Vehicle is moving
Car is moving fast

D:\San>
```

RUN-TIME POLYMORPHISM:

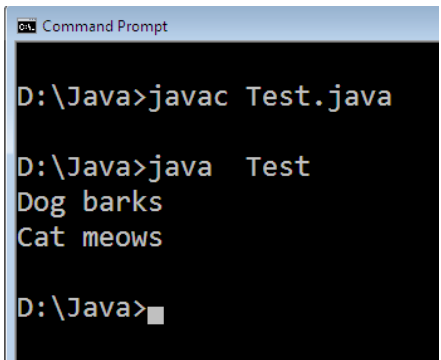
- Runtime polymorphism is a feature where the call to an overridden method is resolved at runtime, not compile time.
- This is possible because the actual object type (not the reference type) decides

which method to run.

- We have a parent class reference pointing to a child class object (upcasting).
- We call a method on the reference.
- The JVM uses dynamic method dispatch to decide at runtime which method to run.

```
class Animal
{
    void sound()
    {
        System.out.println("Animal sound");
    }
}
class Dog extends Animal
{
    @Override
    void sound()
    {
        System.out.println("Dog barks");
    }
}
class Cat extends Animal
{
    @Override
    void sound()
    {
        System.out.println("Cat
meows");
    }
}
public class Test
{
    public static void main(String[] args)
    {
```

Output:



```
Command Prompt
D:\Java>javac Test.java
D:\Java>java Test
Dog barks
Cat meows
D:\Java>
```

```

Animal ref; // Parent reference
ref = new Dog();
ref.sound(); // "Dog barks" (Dog's method runs)
ref = new Cat();
ref.sound(); // "Cat meows" (Cat's method runs)
    }
}

```

SUPER KEYWORD:

- It is a reference variable used within a subclass to refer to the immediate parent class object.
- It plays a crucial role in inheritance by allowing subclasses to interact with their superclass's members.
- `super()` must be the first statement in a subclass constructor if it is used to call a parameterized superclass constructor.
- The `super` keyword cannot be used in static methods because static methods belong to the class, not to a specific object instance, and `super` refers to instance members of a superclass.

PROGRAM TO ILLUSTRATE `super()`:

```

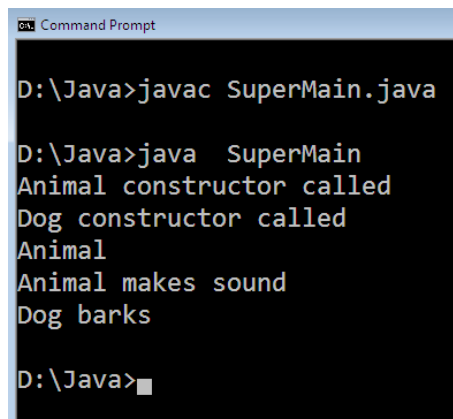
class Animal
{
    String type = "Animal";
    Animal()
    {
        System.out.println("Animal constructor called");
    }
    void sound()
    {
        System.out.println("Animal makes sound");
    }
}
class Dog extends Animal
{
    String type = "Dog";
    Dog()
    {
        super(); // calls Animal() constructor first
        System.out.println("Dog constructor called");
    }
}

```

```

void displayType()
{
    System.out.println(super.type); // Access parent variable
}
void sound()
{
    super.sound(); // Calls Animal's sound()
    System.out.println("Dog barks");
}
}
public class SuperMain
{
    public static void main(String[] args)
    {
        Dog d = new Dog();
        d.displayType();
        d.sound();
    }
}

```

Output:


```

D:\Java>javac SuperMain.java

D:\Java>java SuperMain
Animal constructor called
Dog constructor called
Animal
Animal makes sound
Dog barks

D:\Java>

```

CONSTRUCTOR CHAINING:

- It is the process where one constructor calls another constructor, either within the same class or from its parent class.
- This mechanism promotes code reuse and ensures proper object initialization.

TYPES OF CONSTRUCTOR CHAINING:**1. Within the same class:**

- This is achieved using the `this()` keyword.
- A constructor can call another overloaded constructor of the same class by passing the appropriate arguments to `this()`.
- The `this()` call must be the first statement within the constructor.

2. From a parent class (through inheritance):

- This is achieved using the `super()` keyword.
- A subclass constructor can call a constructor of its immediate superclass using

super().

- The super() call must be the first statement within the subclass constructor.
- If super() is not explicitly called, the default no-argument constructor of the superclass is implicitly invoked.

```
class A
```

```
{
    A()
    {
        this(10); // calls A(int)
        System.out.println("Default constructor of A");
    }
    A(int x)
    {
        System.out.println("Parameterized constructor of A: " + x);
    }
}
```

```
class B extends A
```

```
{
    B()
    {
        super(); // calls A()
        System.out.println("Default constructor of B");
    }
    B(int y)
    {
        this(); // calls B()
        System.out.println("Parameterized constructor of B: " + y);
    }
}
```

```
public class ConstructorChaining
```

```
{
    public static void main(String[] args)
    {
        B obj = new B(20);
    }
}
```

Output:

```
D:\Java>javac ConstructorChaining.java

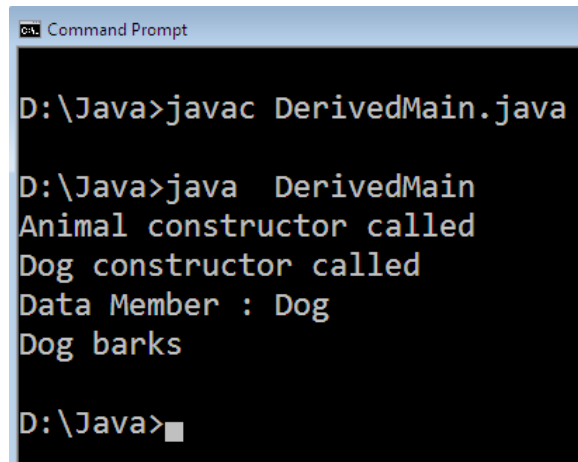
D:\Java>java  ConstructorChaining
Parameterized constructor of A: 10
Default constructor of A
Default constructor of B
Parameterized constructor of B: 20

D:\Java>
```

```
class Animal
{
    String type = "Animal";
    Animal()
    {
        System.out.println("Animal constructor called");
    }
    void sound()
    {
        System.out.println("Animal makes sound");
    }
}
class Dog extends Animal
{
    String type = "Dog";
    Dog()
    {
        System.out.println("Dog constructor called");
    }
    void displayType()
    {
        System.out.println("Data Member : "+type);
    }
    void sound()
    {
        System.out.println("Dog barks");
    }
}
public class DerivedMain
{
    public static void main(String[] args)
    {
        Dog d = new Dog();
        d.displayType();
        d.sound();
    }
}
```

```
}  
}
```

Output:



```
Command Prompt  
D:\Java>javac DerivedMain.java  
  
D:\Java>java DerivedMain  
Animal constructor called  
Dog constructor called  
Data Member : Dog  
Dog barks  
  
D:\Java>
```