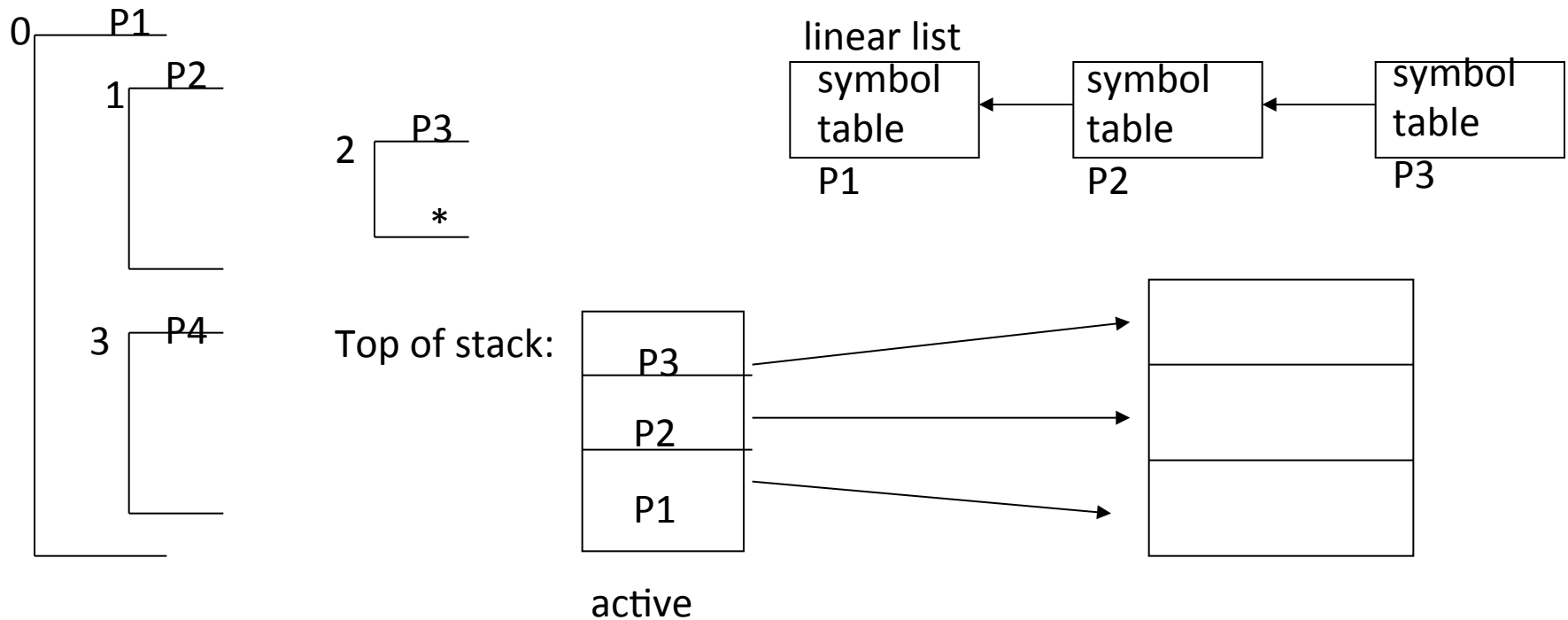




# Symbol Tables with Classes

# Recall Using a Symbol Table

- To process a portion of the program that creates a scope (block statements, function calls, classes, etc.)
  - Enter a new scope.
  - Add all variable declarations to the symbol table.
  - Process the body of the block/function/class.
  - Exit the scope.
- Much of semantic analysis is defined in terms of recursive AST traversals like this.



Search from top of stack

When table not needed, delete the pointer to its symbol table else keep pointers to table - code generation, debugging - - delete only from active

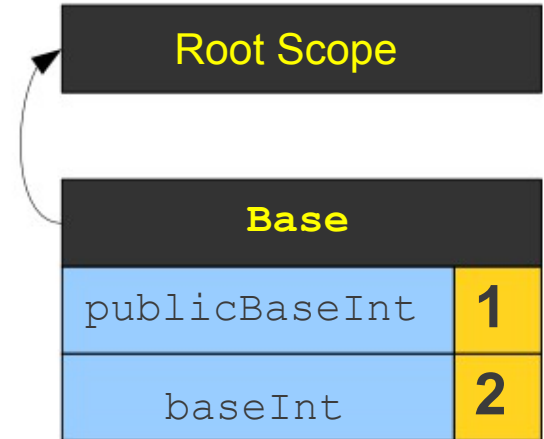
# Scoping with Inheritance

Root Scope

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}
```

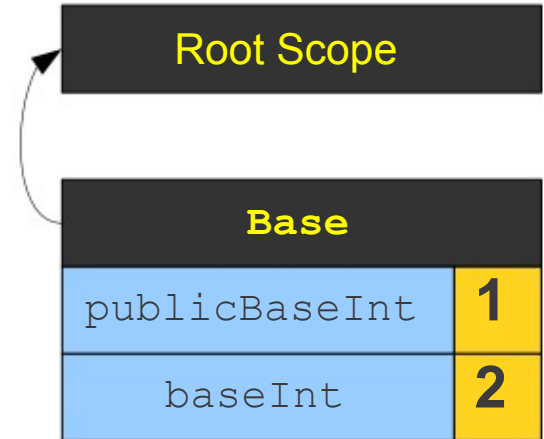
# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}
```



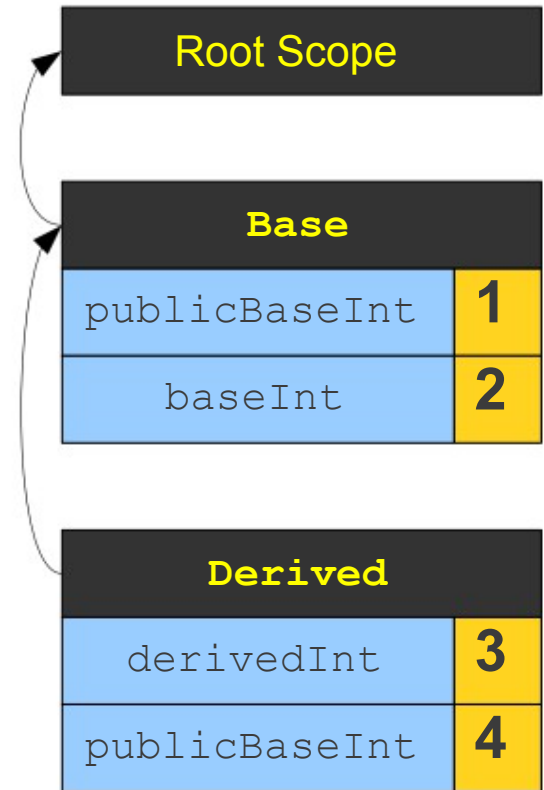
# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```



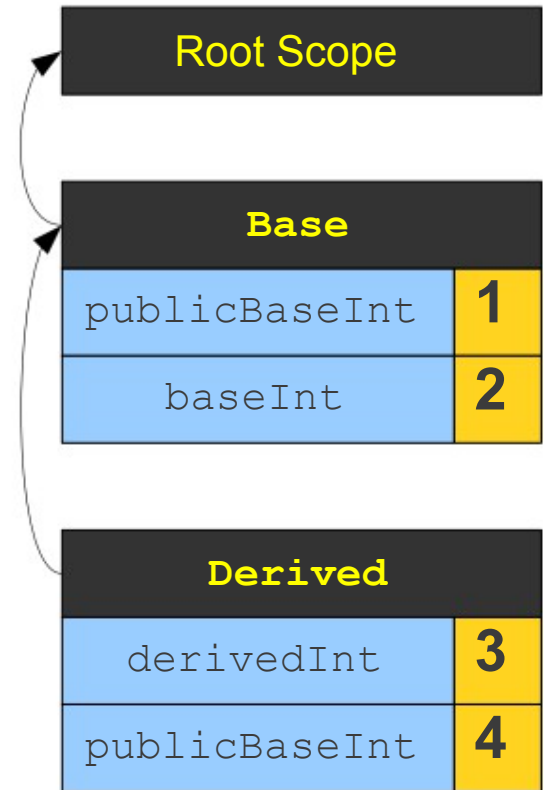

# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```



# Scoping with Inheritance

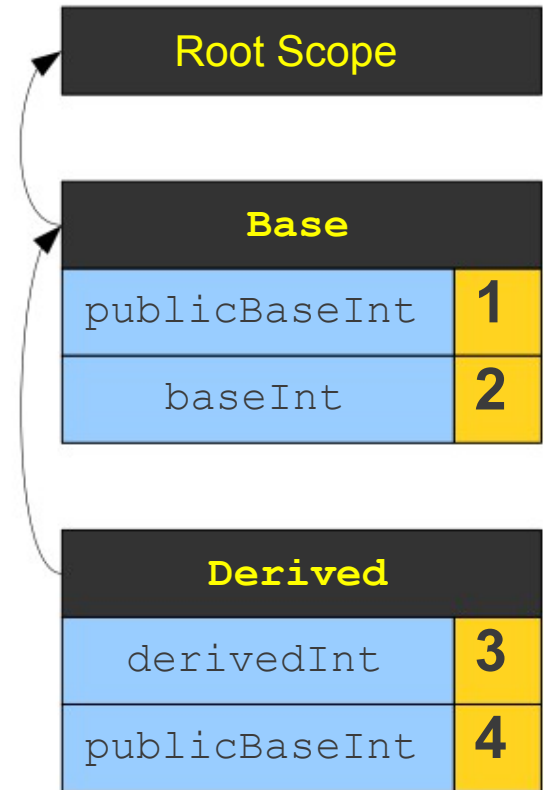

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```





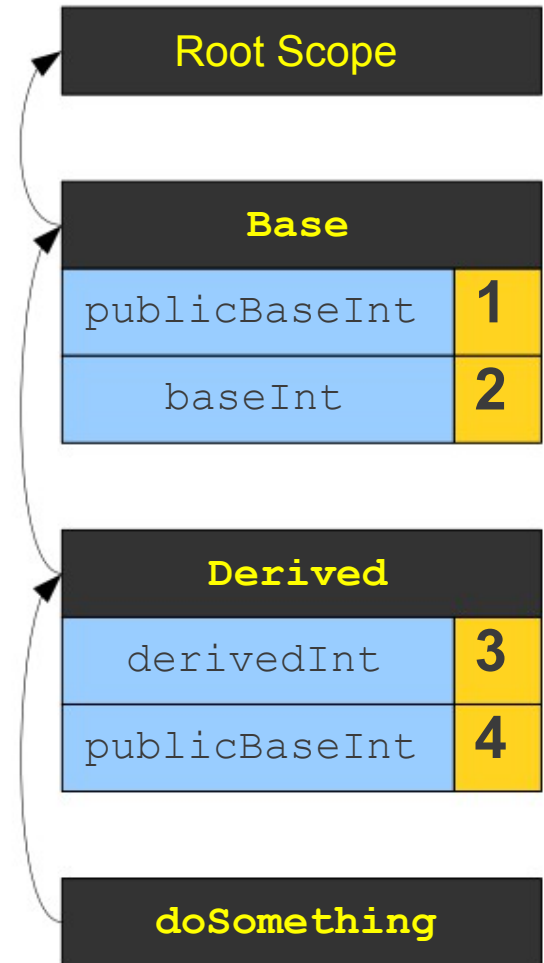

# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```



# Scoping with Inheritance

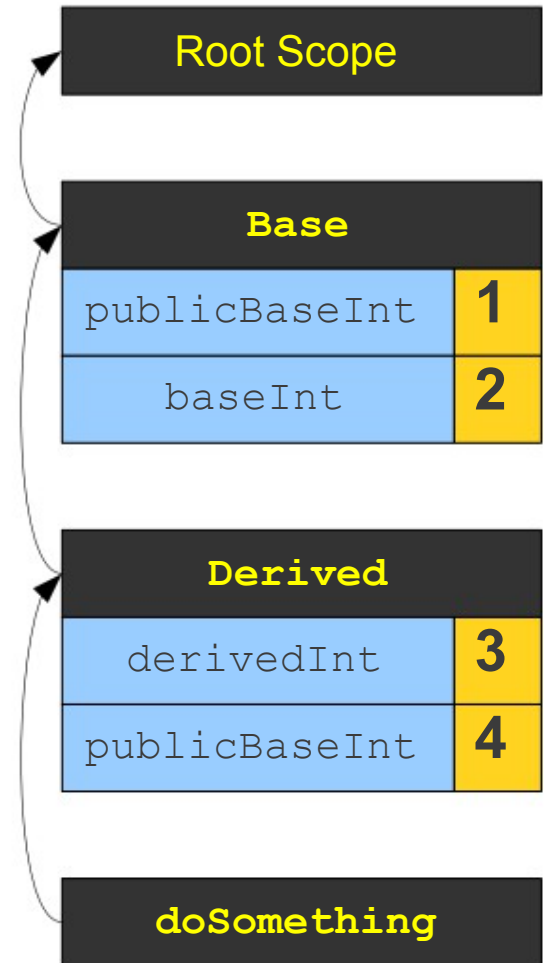
```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

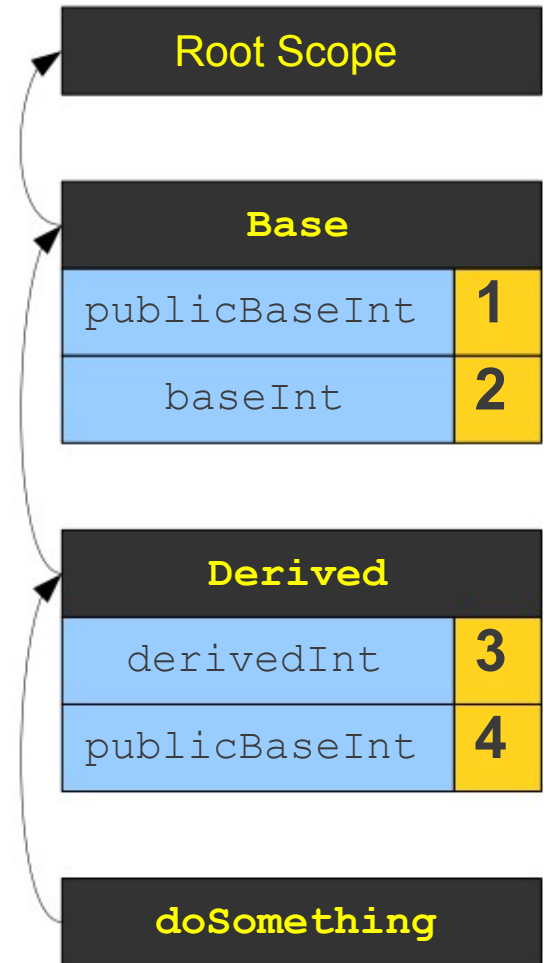
>



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

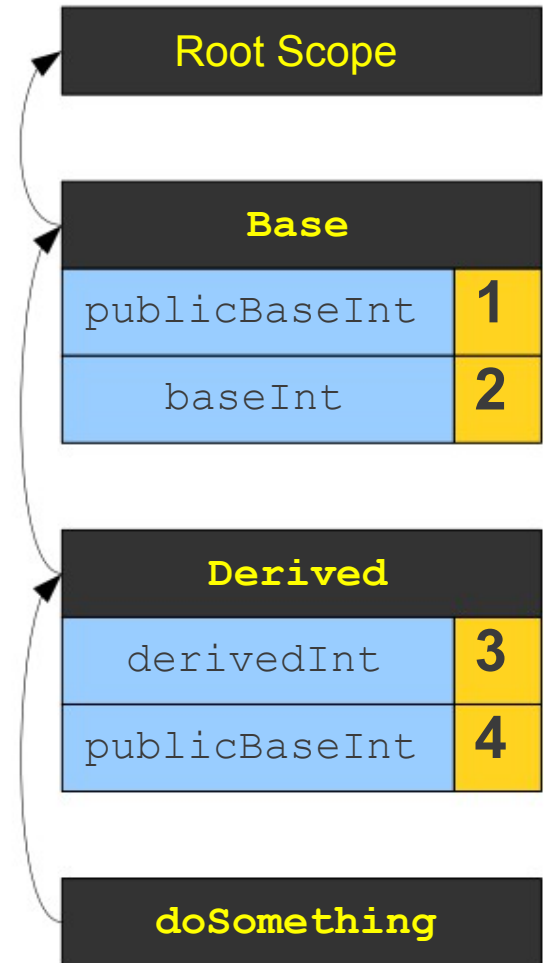
**> 4**



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

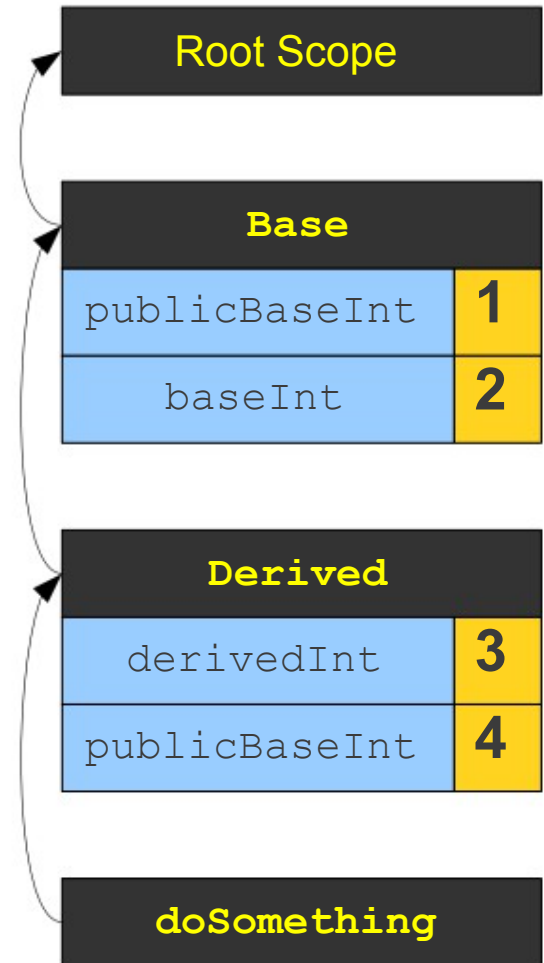
**> 4**



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

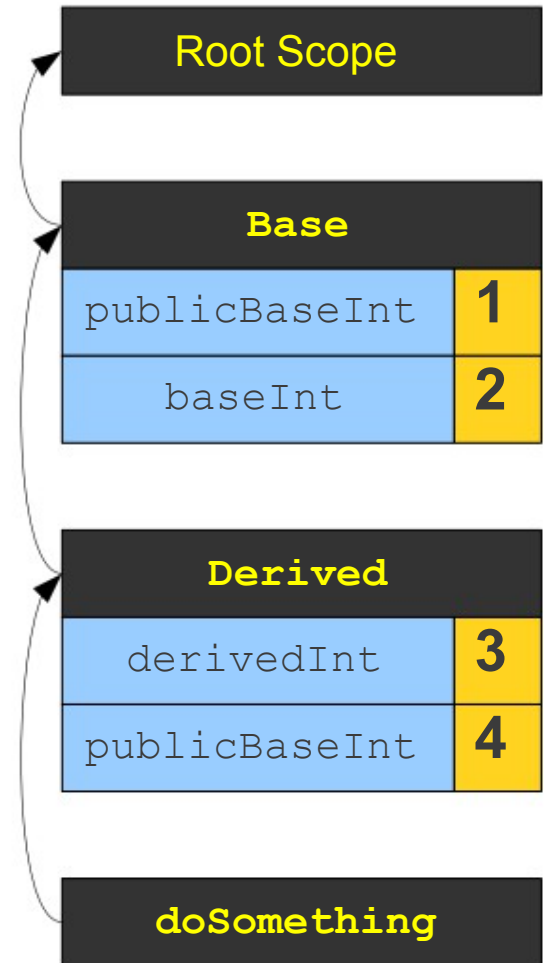
```
> 4  
  2
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

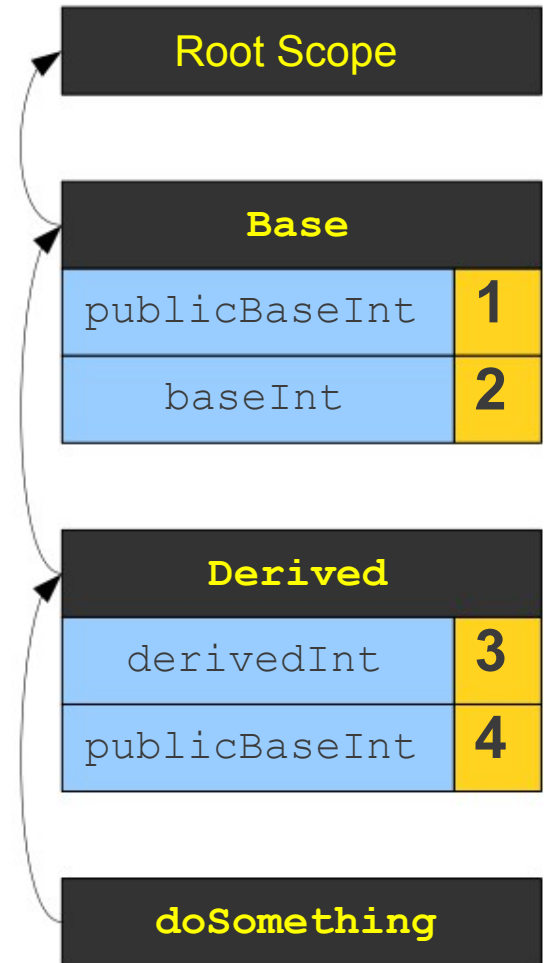
```
> 4  
  2
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

```
> 4  
  2  
  3
```

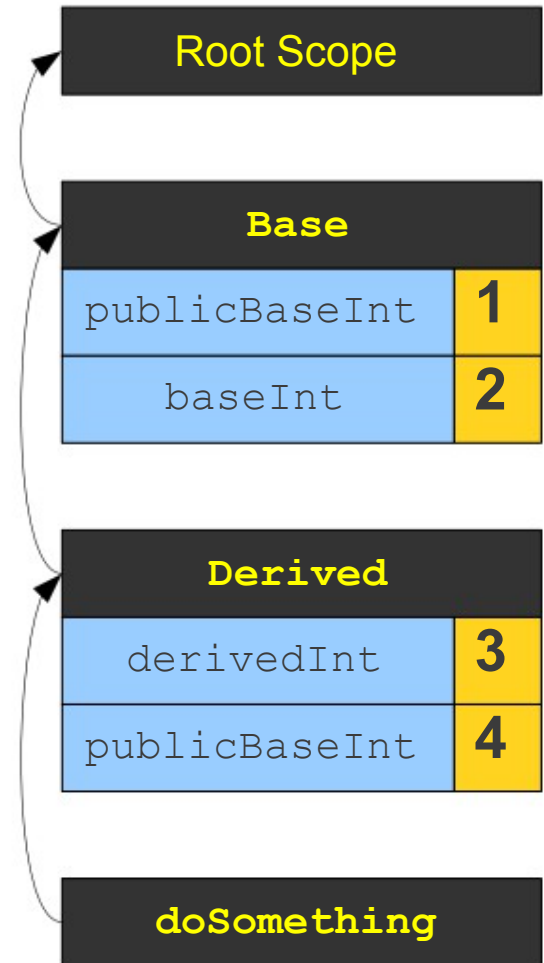




# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

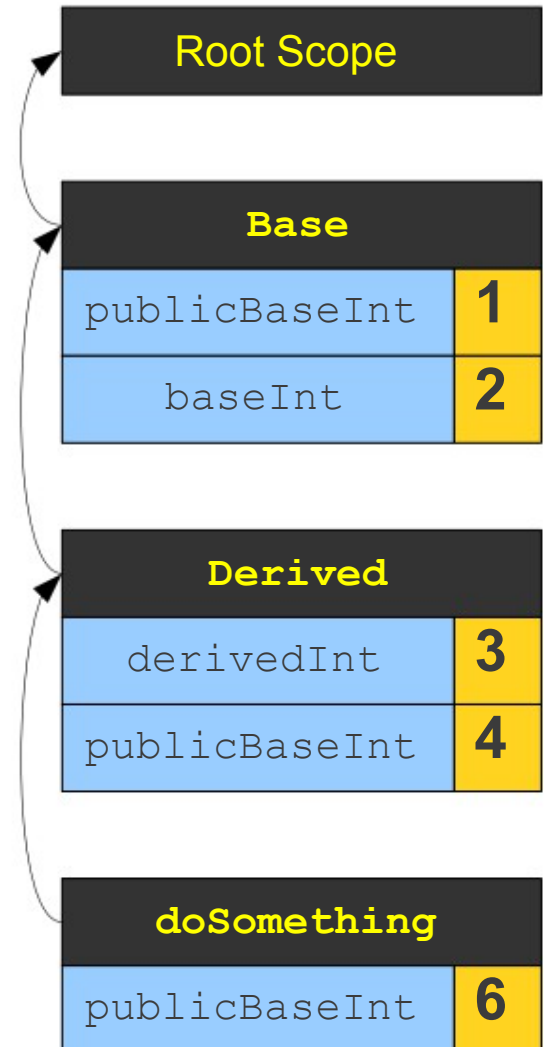
```
> 4  
  2  
  3
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

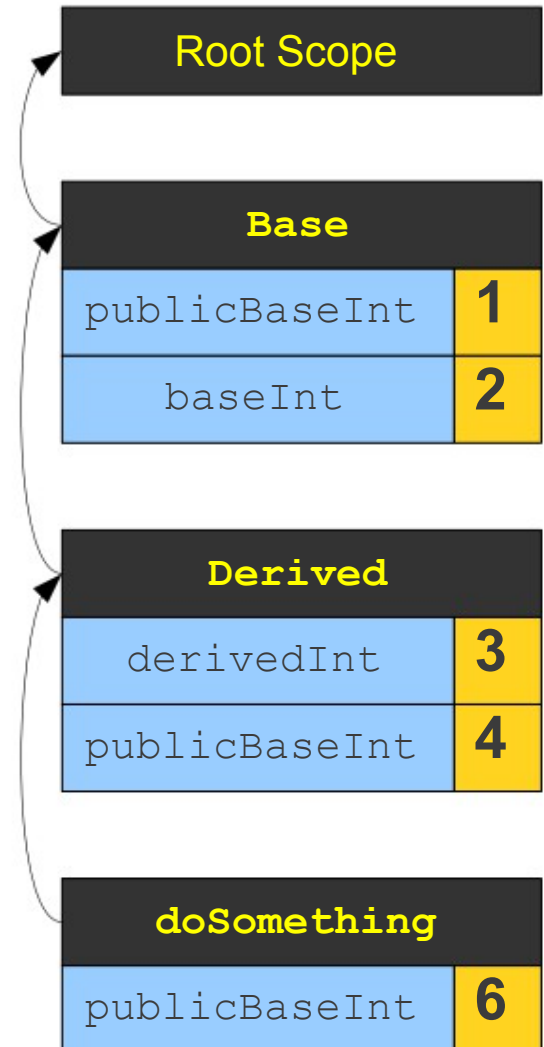
```
> 4  
  2  
  3
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

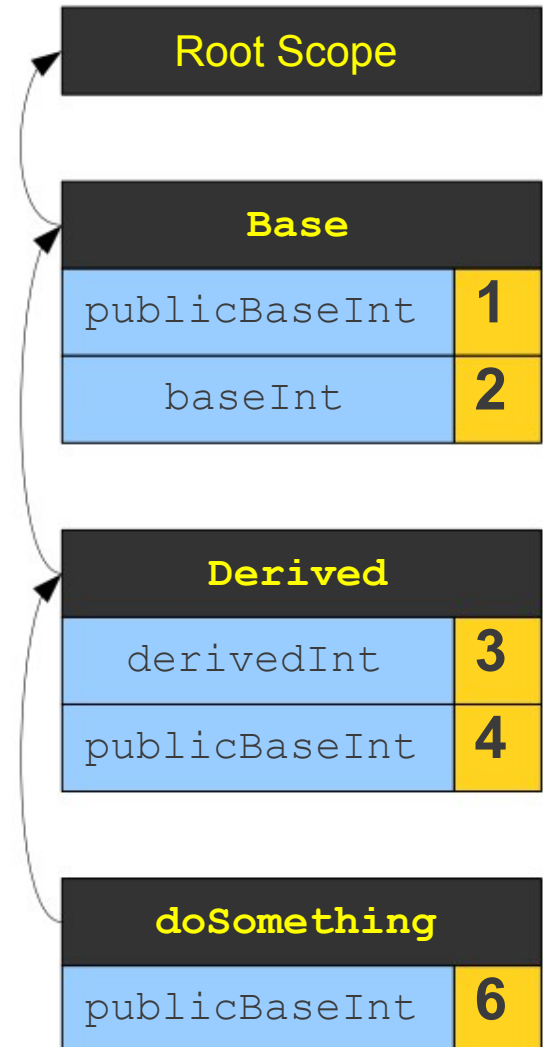
```
> 4  
  2  
  3
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

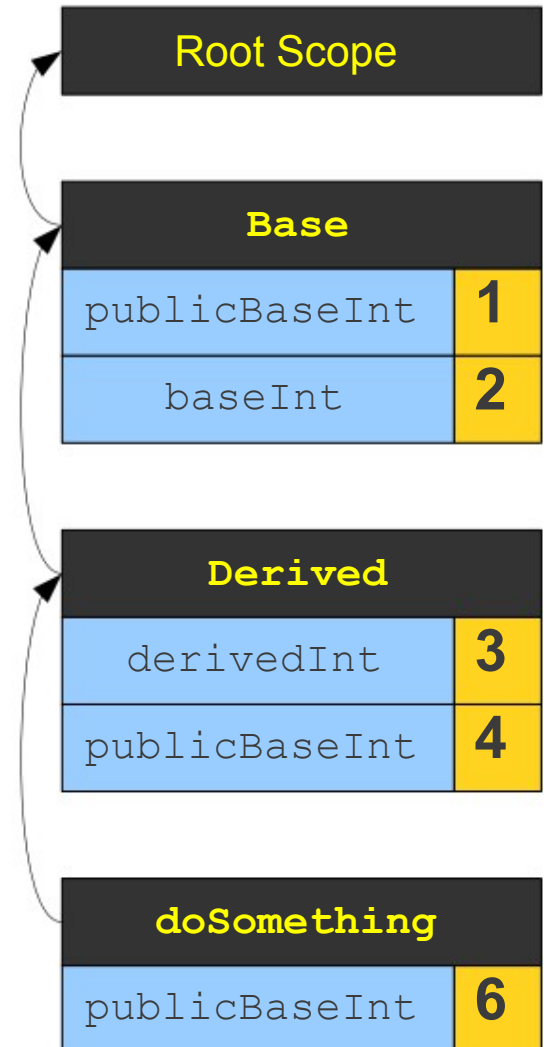
```
> 4  
  2  
  3  
  6
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected int baseInt = 2;  
}  
  
public class Derived extends Base {  
    public int derivedInt = 3;  
    public int publicBaseInt = 4;  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

```
> 4  
  2  
  3  
  6
```



# Scoping with Inheritance

```
public class Base {  
    public int publicBaseInt = 1;  
    protected  
}  
  
public class Derived {  
    public  
    public  
  
    public void doSomething() {  
        System.out.println(publicBaseInt);  
        System.out.println(baseInt);  
        System.out.println(derivedInt);  
  
        int publicBaseInt = 6;  
        System.out.println(publicBaseInt);  
    }  
}
```

Typically the scope would also contain the names “**Base**” and “**Derived**,” along with the function name “**doSomething**.” For simplicity, I've left these out.

```
> 4  
  2  
  3  
  6
```

Root Scope

Base

publicBaseInt

1

baseInt

2

Derived

derivedInt

3

publicBaseInt

4

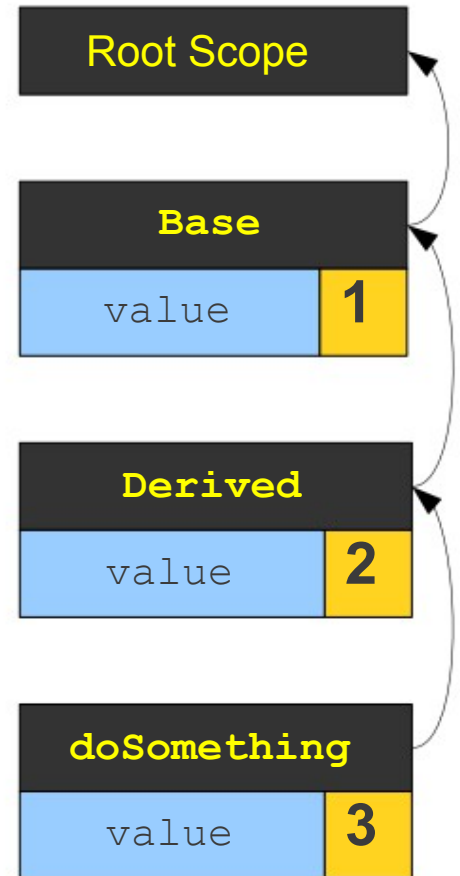
doSomething

publicBaseInt

6

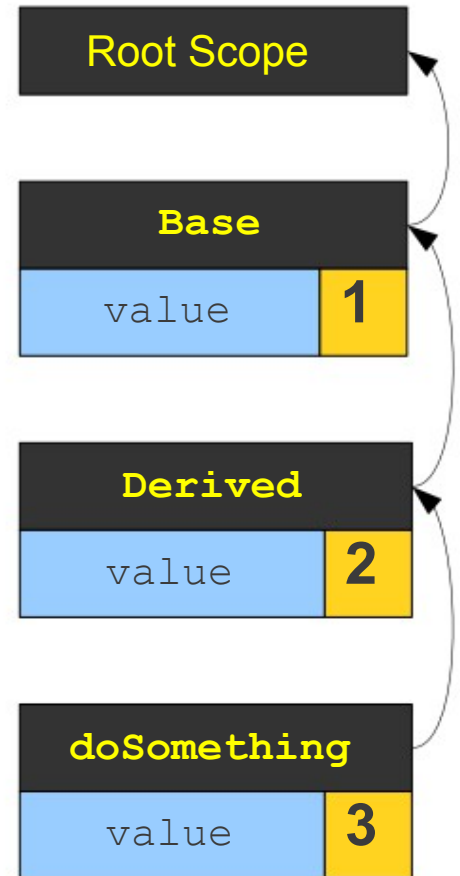
# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



# Explicit Disambiguation

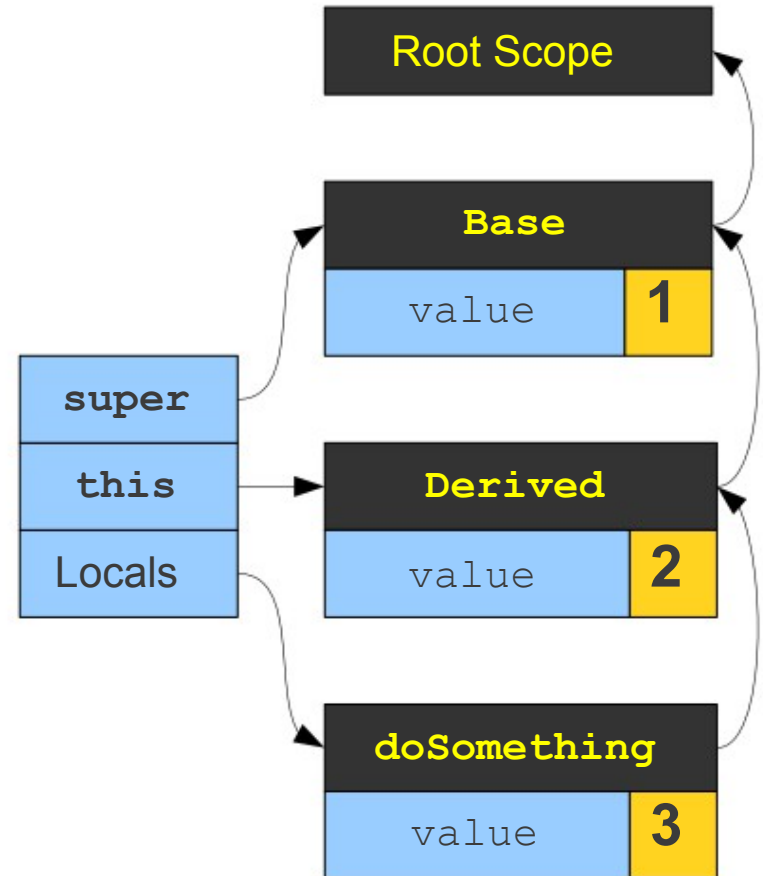
```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```





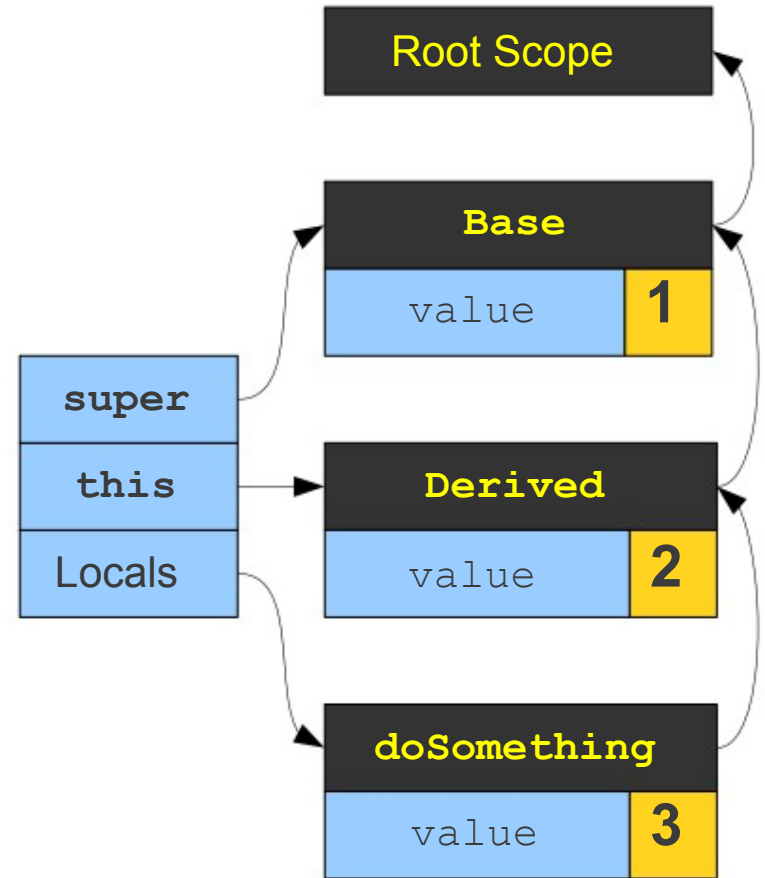
# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



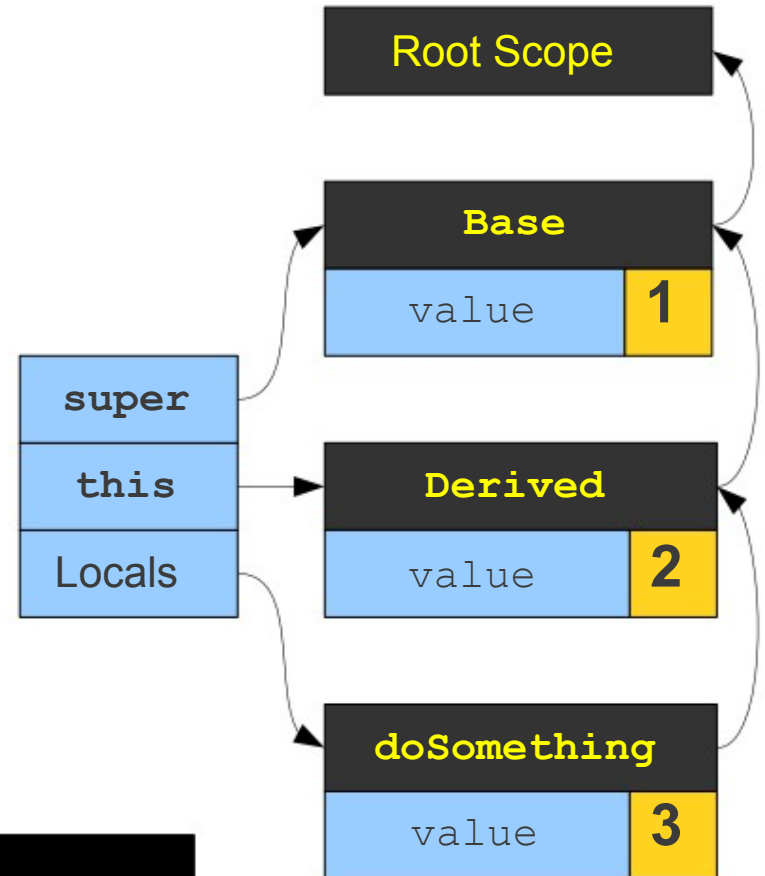
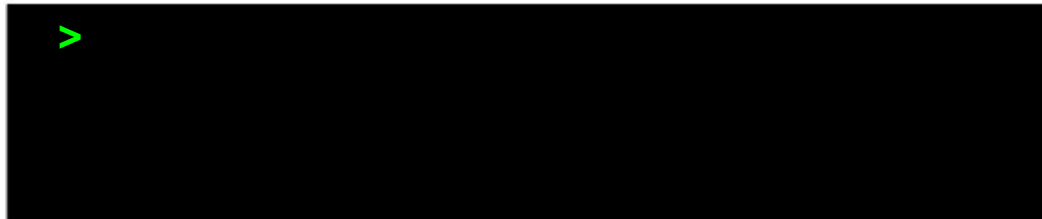
# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



# Explicit Disambiguation

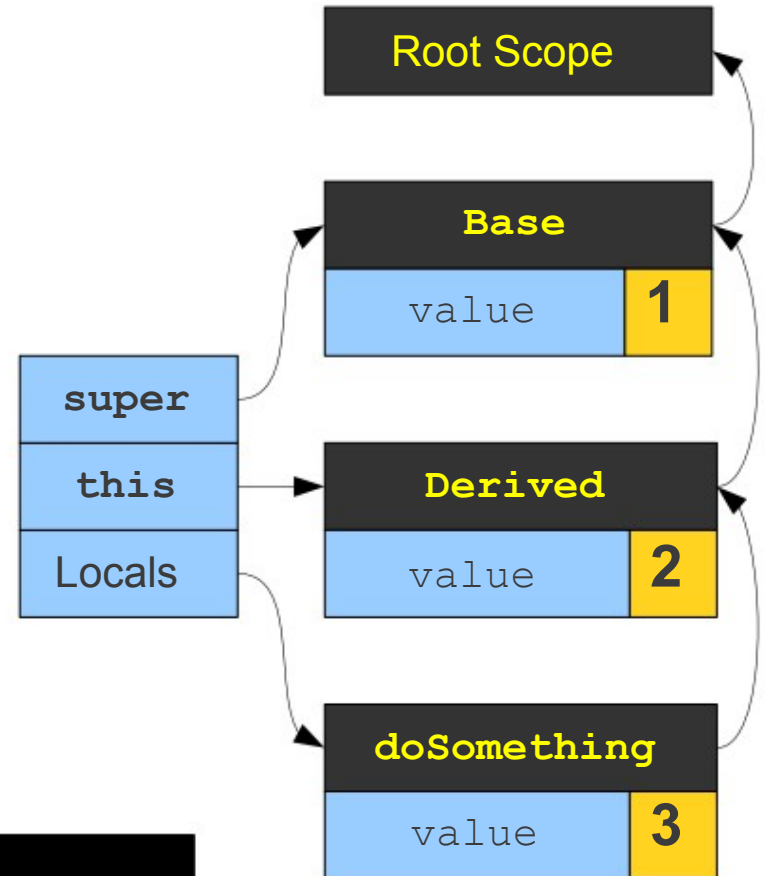
```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

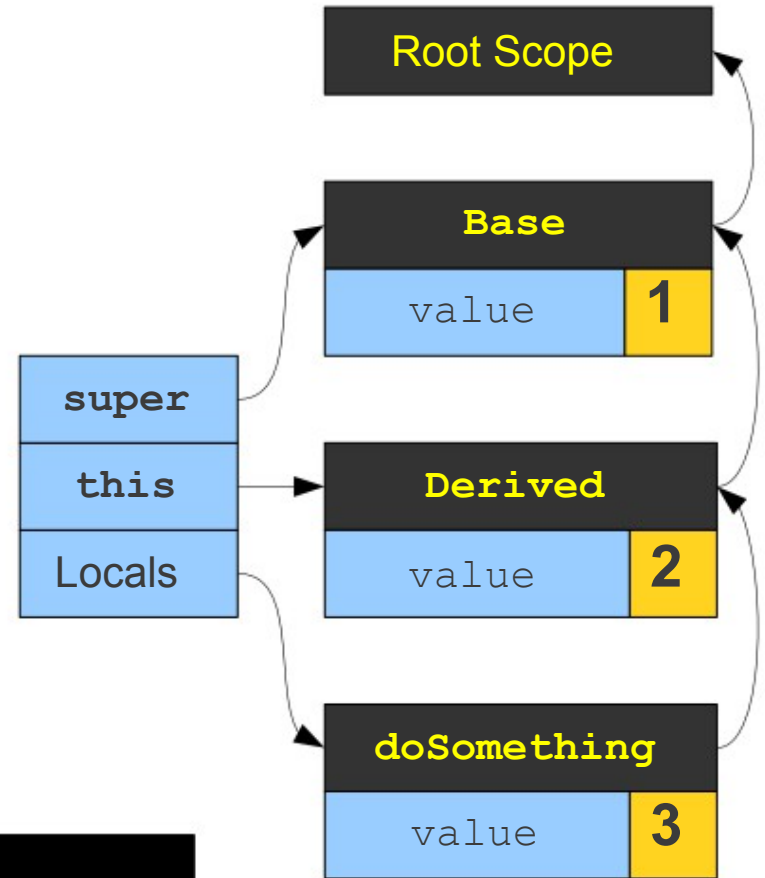
```
>
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

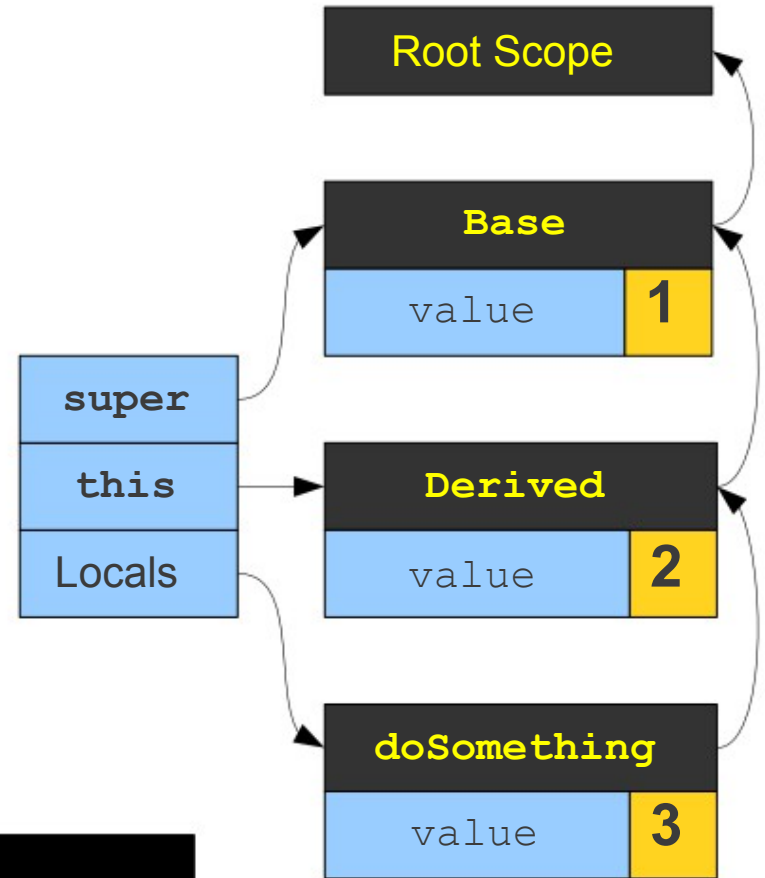
> 3



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

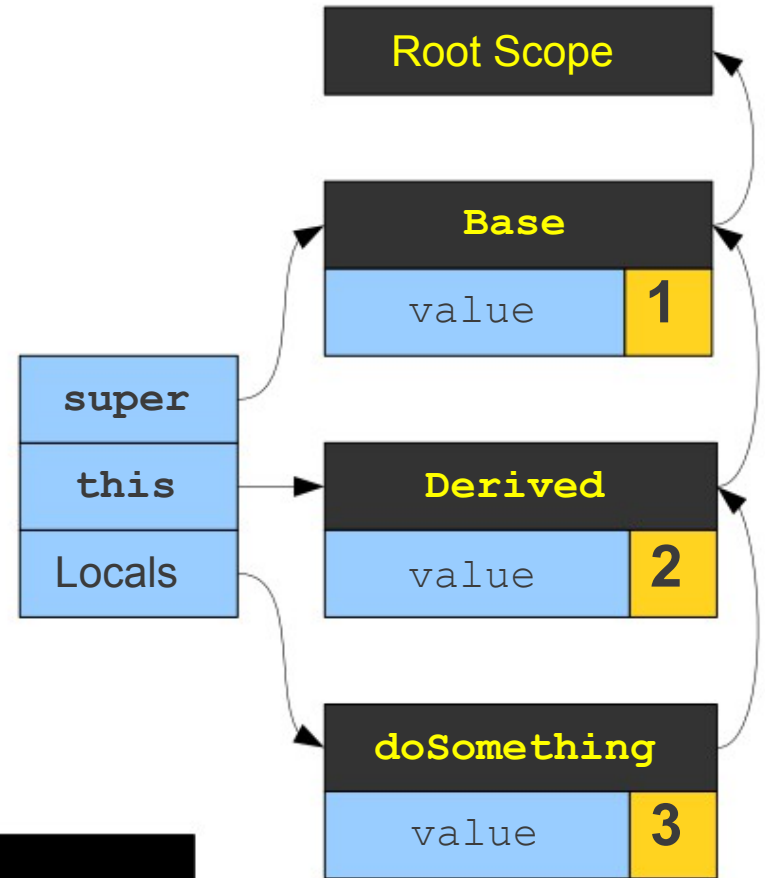
> 3



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

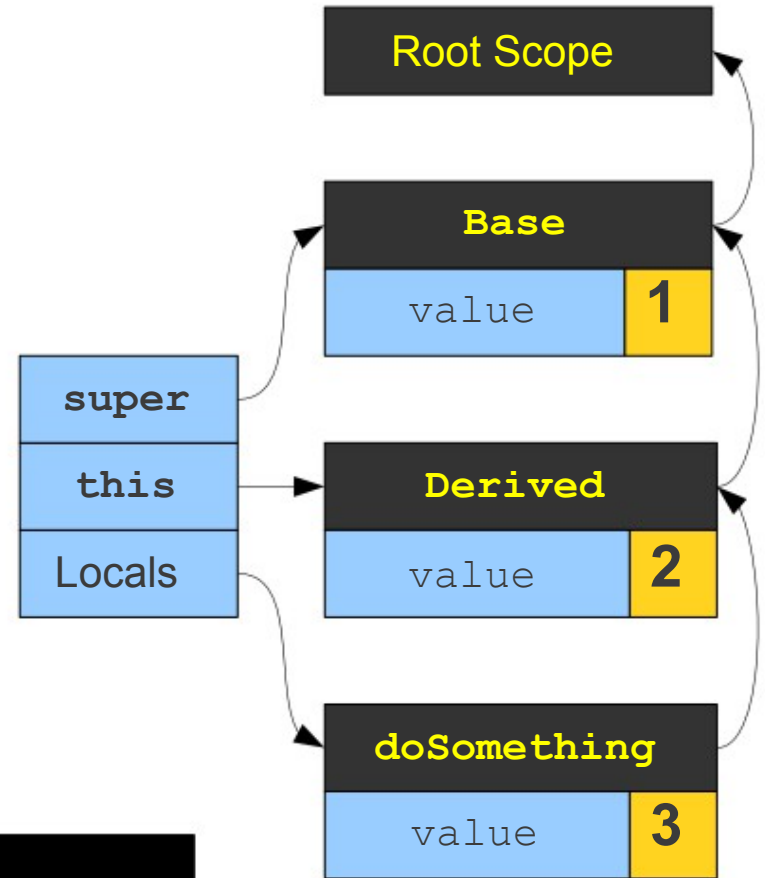
```
> 3  
  2
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

```
> 3  
  2
```

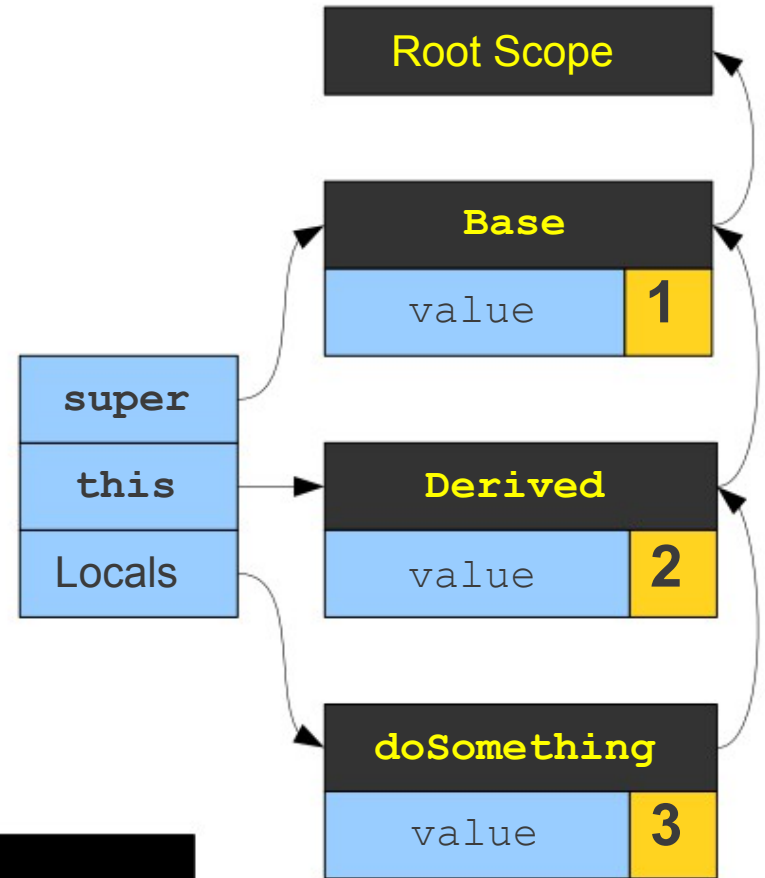




# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

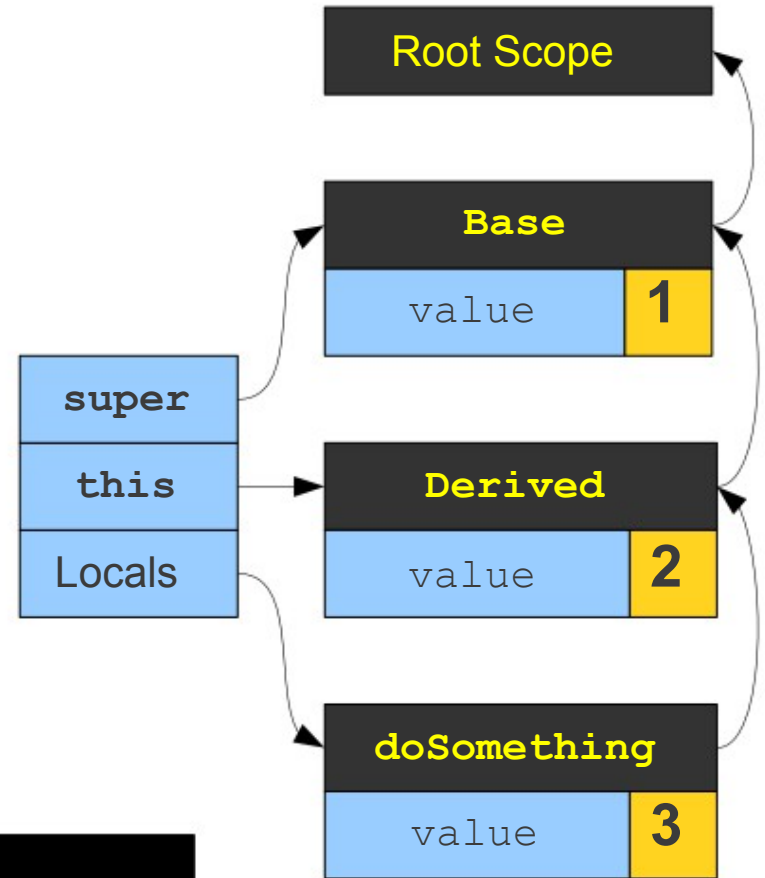
```
> 3  
  2  
  1
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
    public int value = 2;  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

```
> 3  
  2  
  1
```

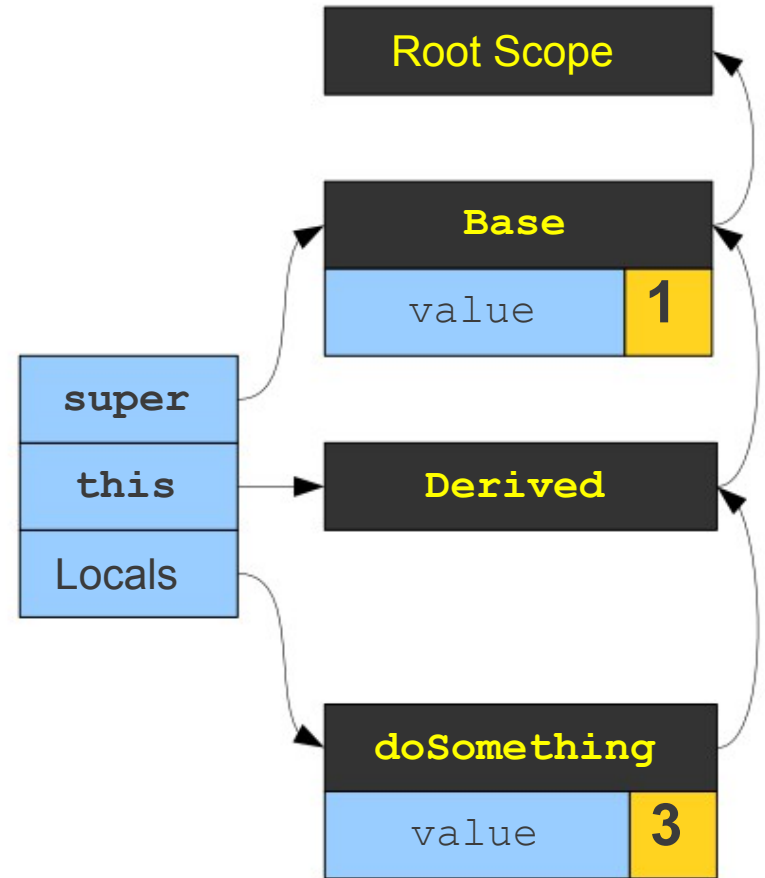


# Explicit Disambiguation – no value Declared in Derived

```
public class Base {
    public int value = 1;
}

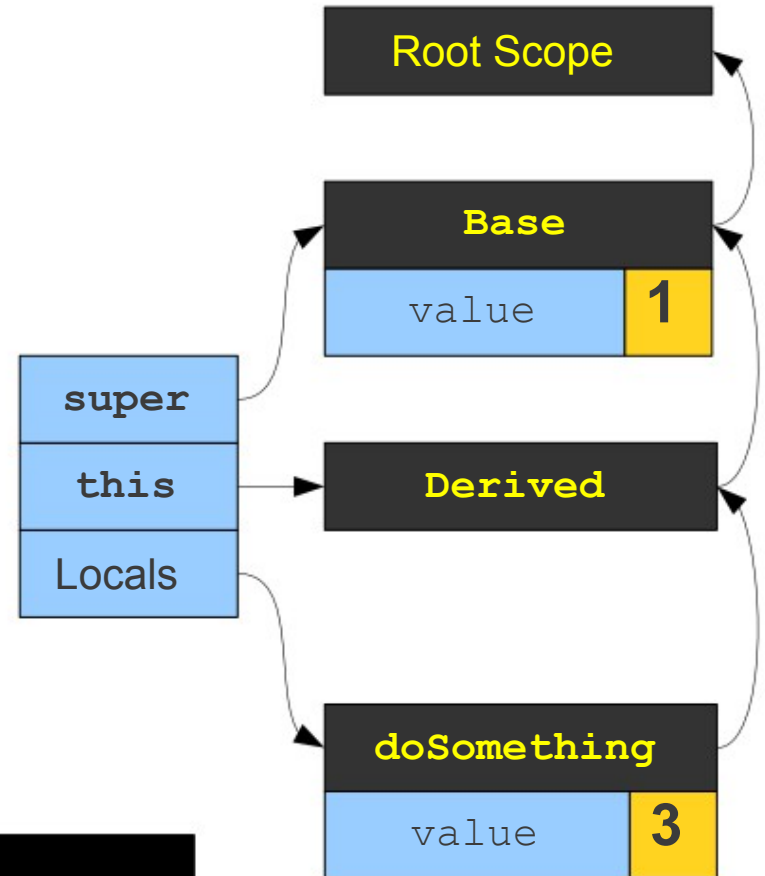
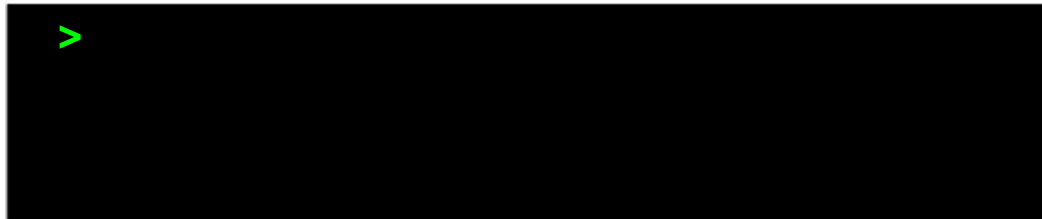
public class Derived extends Base {

    public void doSomething() {
        int value = 3;
        System.out.println(value);
        System.out.println(this.value);
        System.out.println(super.value);
    }
}
```



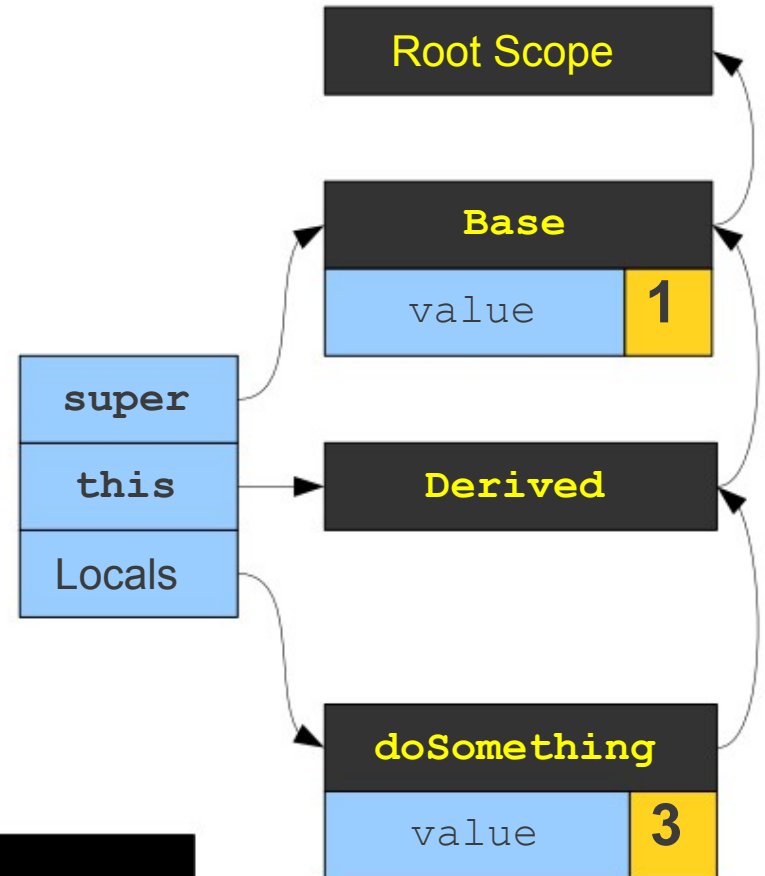
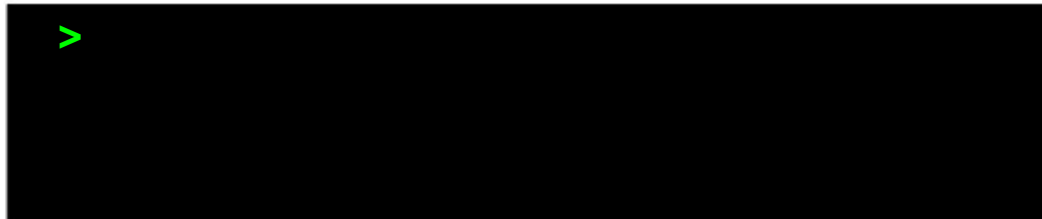
# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



# Explicit Disambiguation

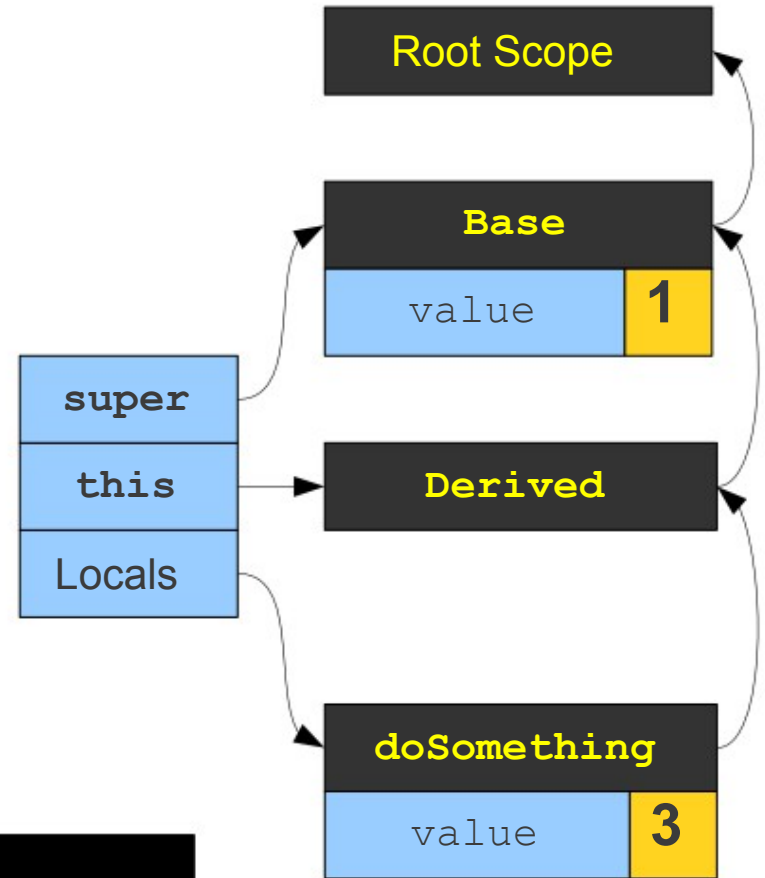
```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

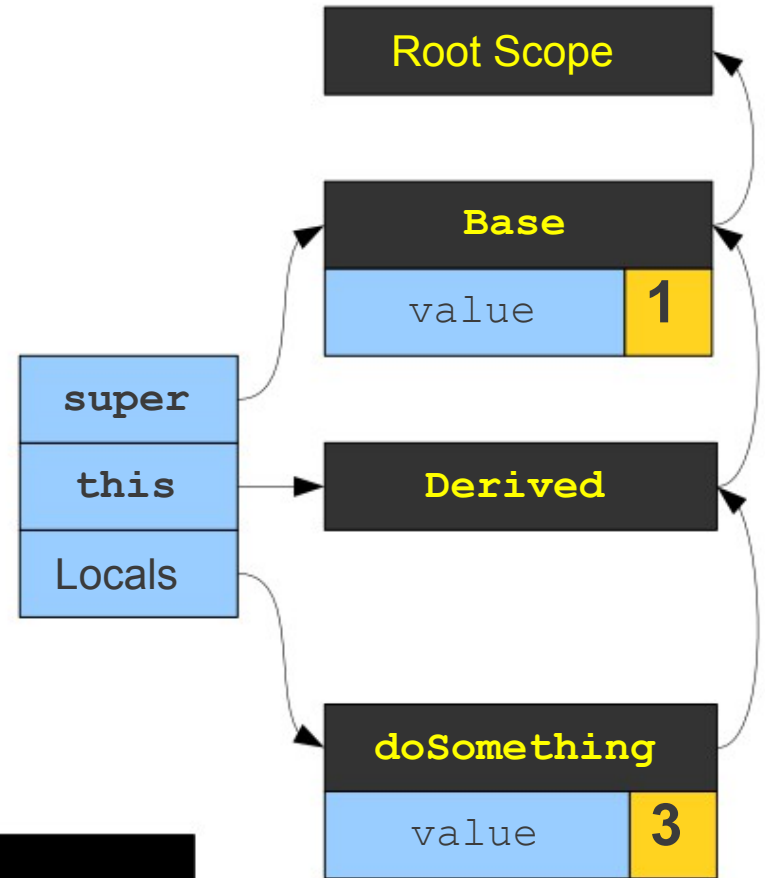
> 3



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

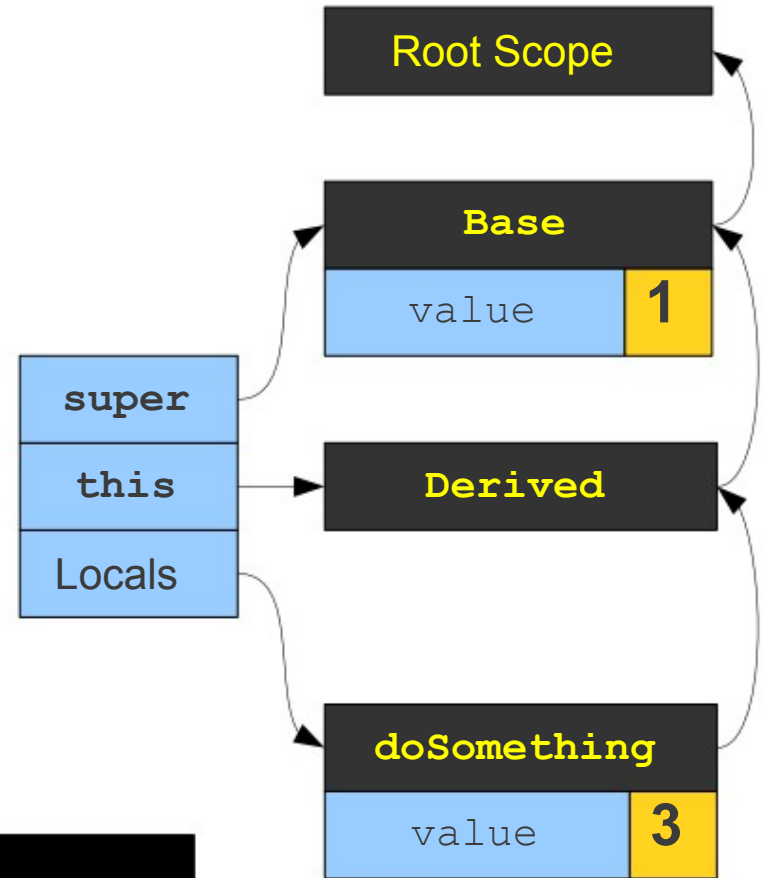
> 3



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

```
> 3  
  1
```

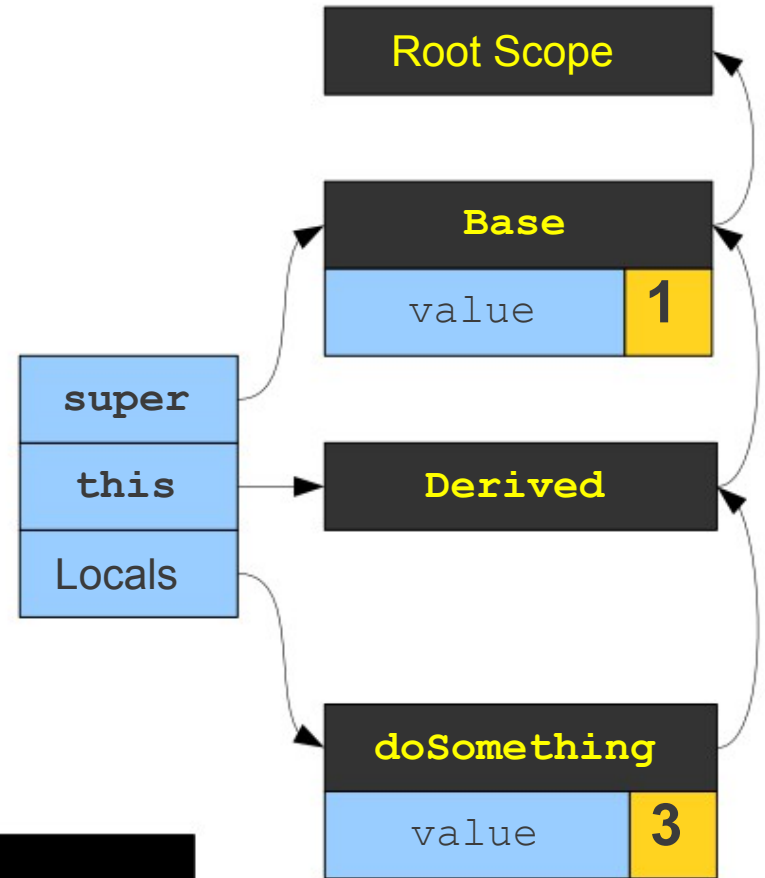




# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

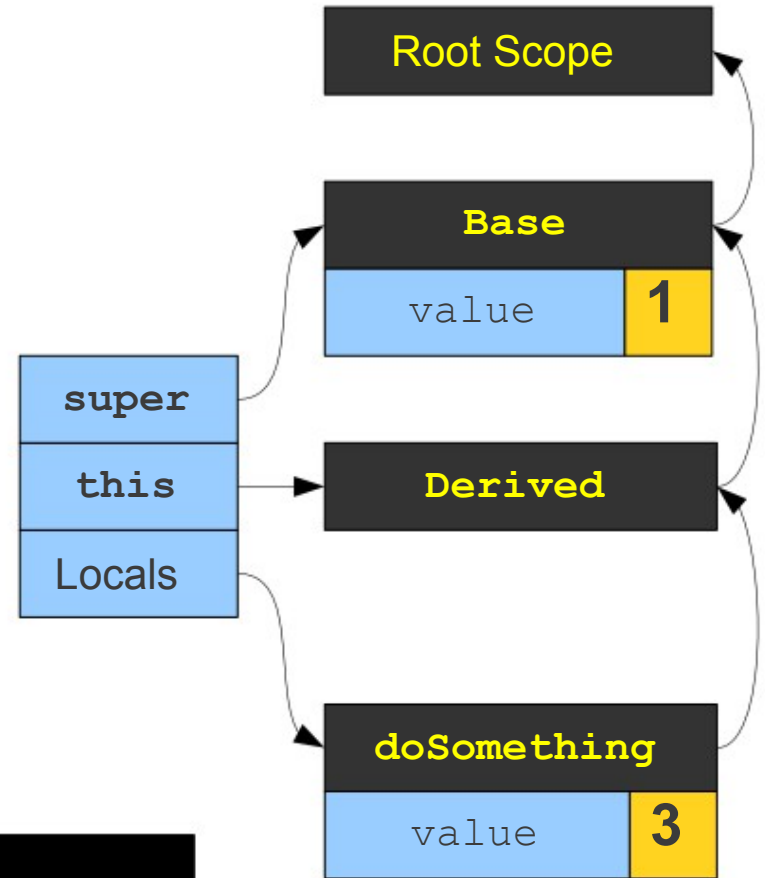
```
> 3  
  1
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

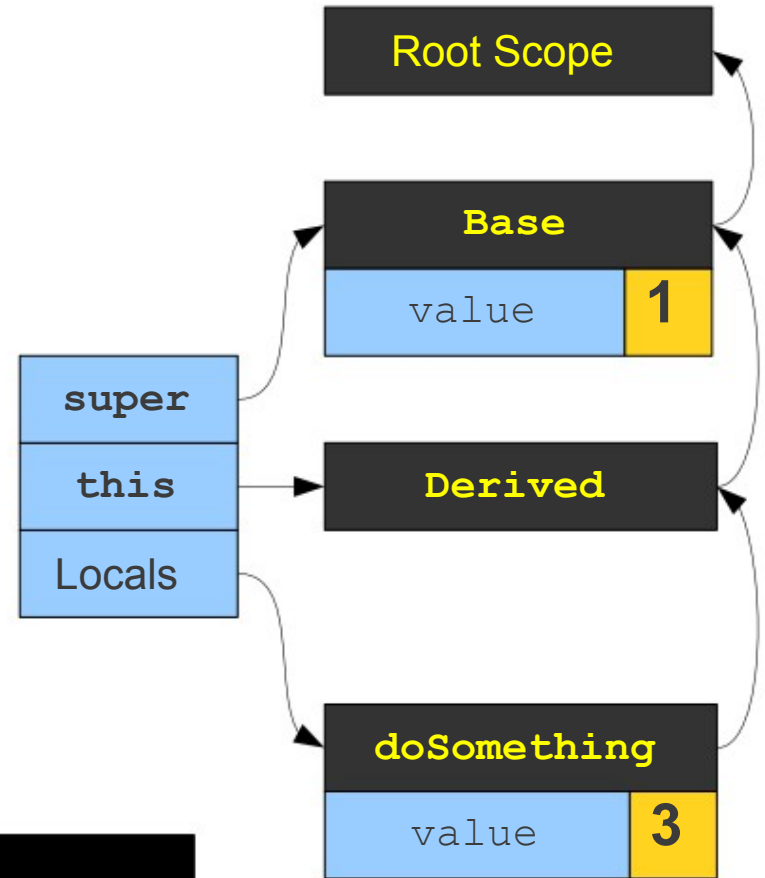
```
> 3  
  1  
  1
```



# Explicit Disambiguation

```
public class Base {  
    public int value = 1;  
}  
  
public class Derived extends Base {  
  
    public void doSomething() {  
        int value = 3;  
        System.out.println(value);  
        System.out.println(this.value);  
        System.out.println(super.value);  
    }  
}
```

```
> 3  
  1  
  1
```



# Disambiguating Scopes

- Maintain a second table of pointers into the scope stack.
- When looking up a value in a specific scope, begin the search from that scope.
- Some languages allow you to jump up to any arbitrary base class (for example, C++).