

# Compiler Support: Construction in Java

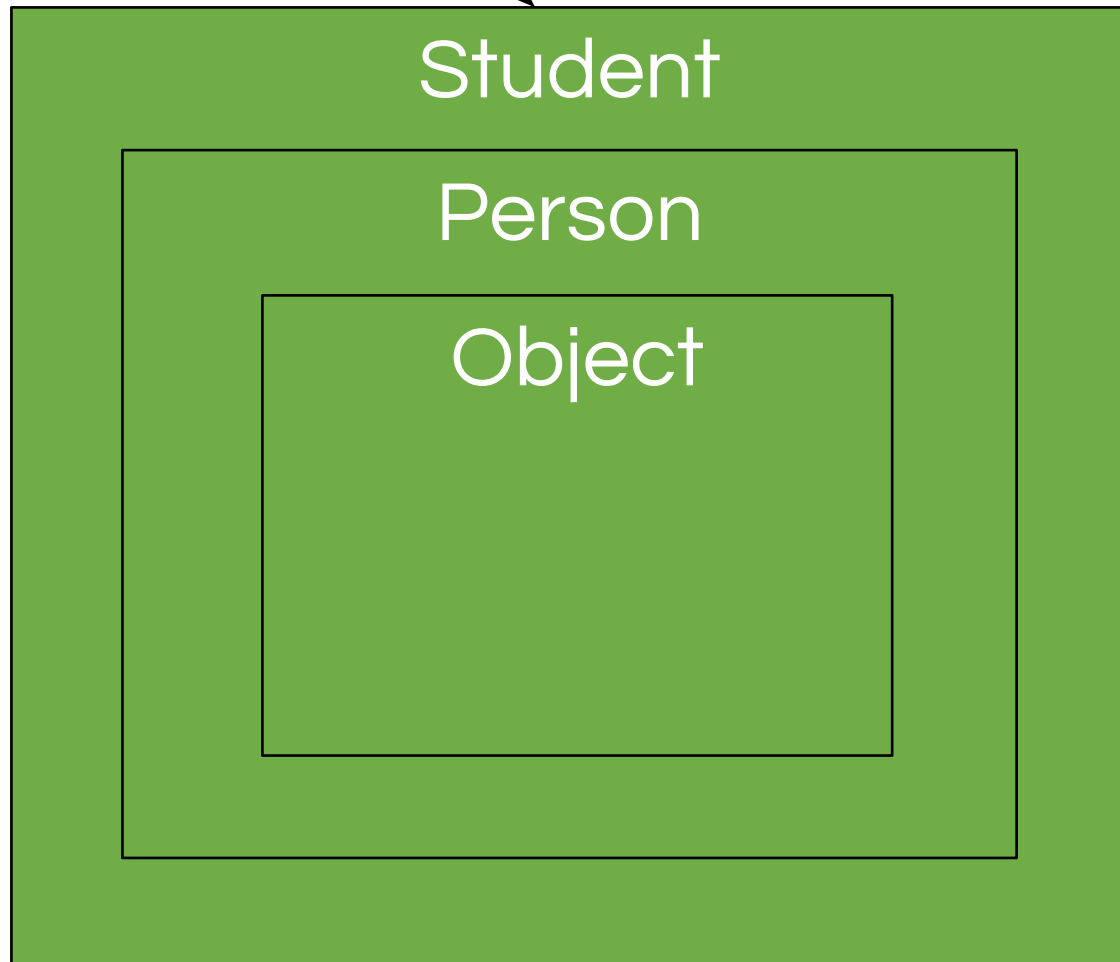


This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)  
by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

## By the end of this video you will be able to...

- Discuss how the Java Compiler ensures object construction occurs from the inside out

```
Student s = new Student();
```



**Wait, I don't  
remember  
extending  
Object...**

**Your Code**



```
graph TD; A[Your Code] --> B[Java Compiler]
```

A diagram illustrating the compilation process. A white box with a blue border labeled 'Your Code' has a black arrow pointing to a solid blue box labeled 'Java Compiler'.

Java  
Compiler

**Your Code**

```
graph TD; A[Your Code] --> B[Java Compiler]; B --> C[Bytecode];
```

A flowchart illustrating the compilation process. It starts with a box labeled 'Your Code' at the top left. An arrow points from this box to a central box labeled 'Java Compiler'. Another arrow points from the 'Java Compiler' box to a box labeled 'Bytecode' at the bottom right. The 'Java Compiler' box is blue, while the others are white with blue borders. The entire diagram is set against a white background with a blue, yellow, and green vertical bar on the left side.

Java  
Compiler

**Bytecode**

```
public class Person
{
    private String name;
}
```

```
public class Person
{
    private String name;
}
```

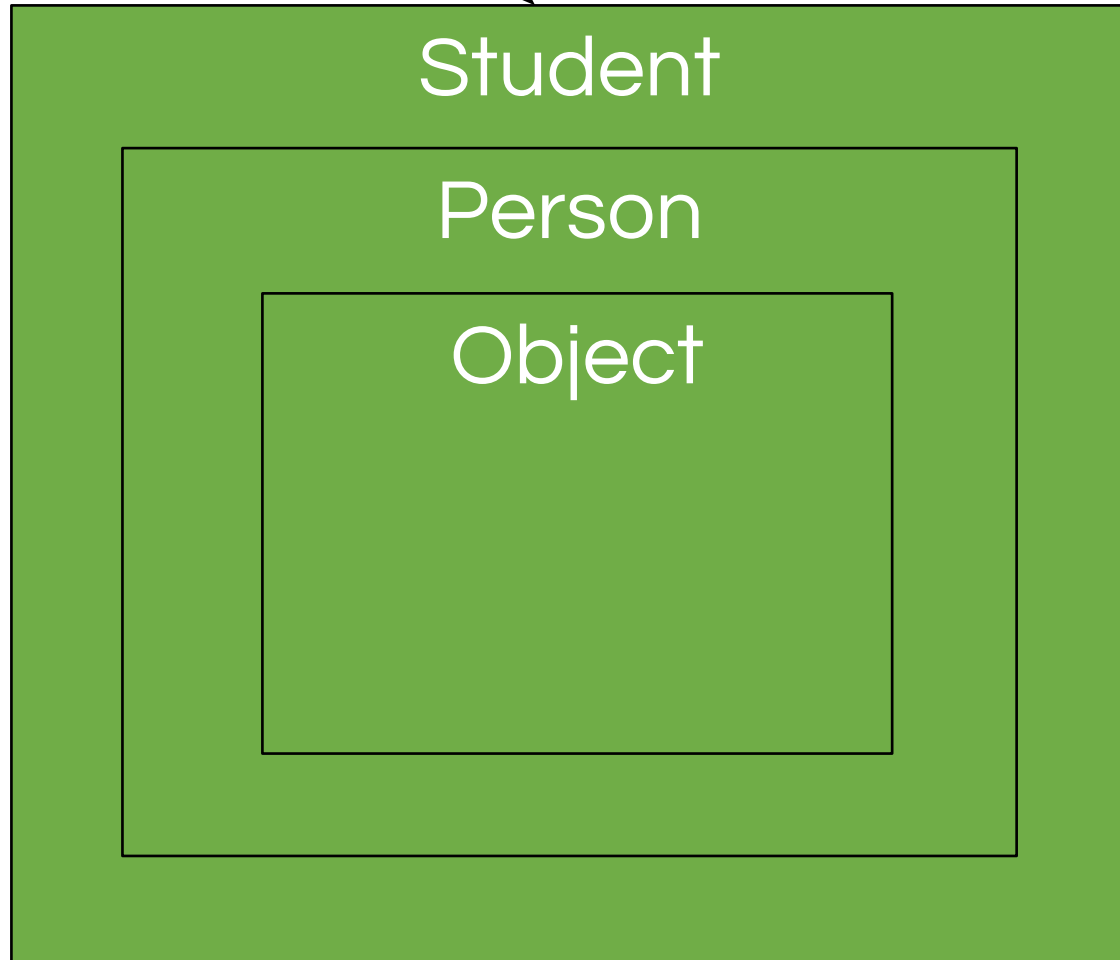


```
public class Person extends Object
{
    private String name;
}
```

## Rule #1

No superclass? Compiler  
inserts:  
`extends Object`

```
Student s = new Student();
```



But where did we  
call `Person()` and  
later `Object()`



```
public class Person
{
    private String name;
}
```



```
public class Person extends Object
{
    private String name;

    public Person() {

    }
}
```

## Rule #2

No constructor? Java gives you one for you.

## Rule #3

1<sup>st</sup> Line must be:

```
this (argsopt )
```

or

```
super ( argsopt )
```

Otherwise, Java inserts:

```
"super() ;"
```

### Rule #3

1<sup>st</sup> Line must be:

`this (argsopt )`

or

`super ( argsopt )`

Otherwise, Java inserts:

`"super() ;"`

**Same class constructor  
call**



### Rule #3

1<sup>st</sup> Line must be:

`this (argsopt )`

or

`super ( argsopt )`

**Same class constructor  
call**



**Super class constructor  
call**



Otherwise, Java inserts:

`"super() ;"`

```
public class Person
{
    private String name;
}
```



```
public class Person extends Object
{
    private String name;

    public Person() {
        super();
    }
}
```

```
public class Student extends Person
{
}
```



```
public class Student extends Person
{

}

```

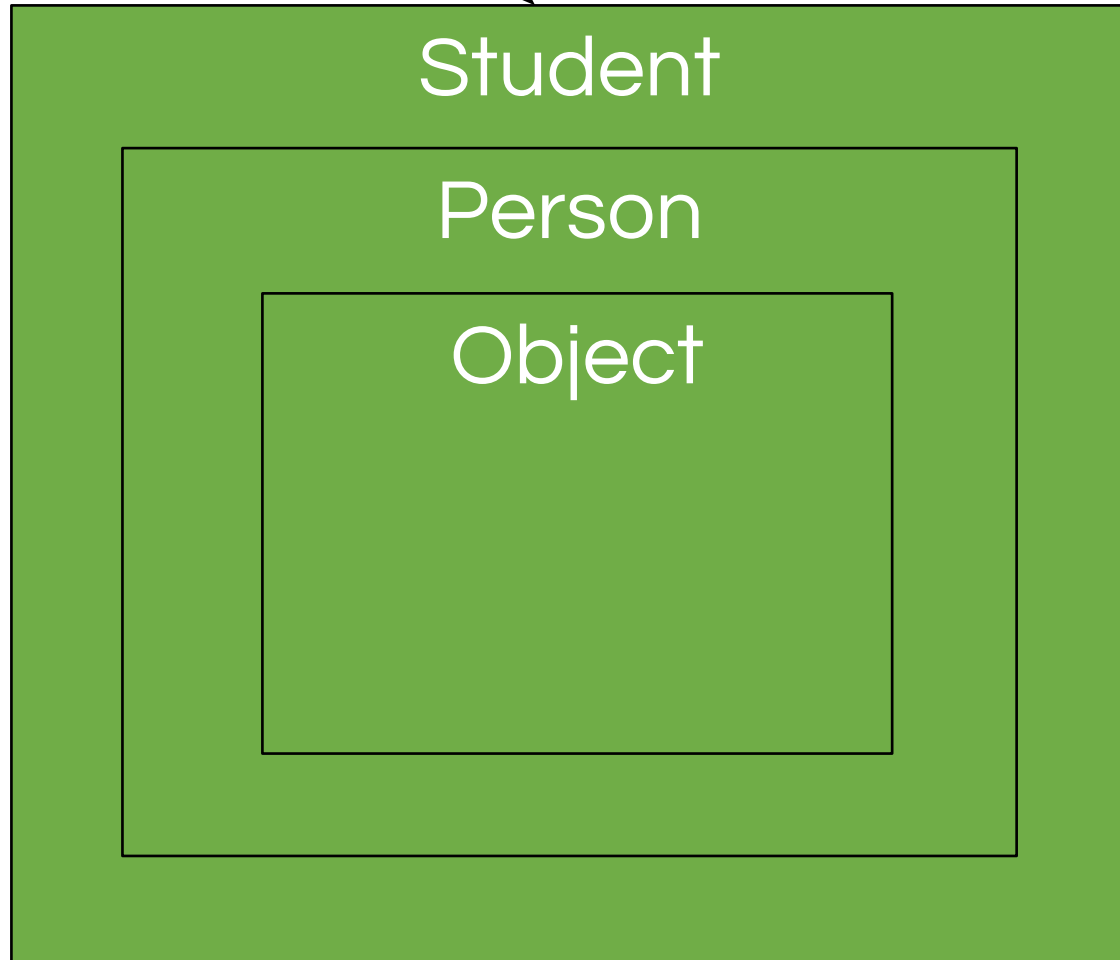


```
public class Student extends Person
{
}
```



```
public class Student extends Person
{
    public Student()
    {
        super();
    }
}
```

```
Student s = new Student();
```



**The compiler  
makes this  
happen!**



```
public class Student extends Person
{
}
```



```
public class Student extends Person
{
    public Student()
    {
        super();
    }
}
```

**But how do we  
initialize name ?**