

# CSE115 / CSE503

## Introduction to Computer Science I

Dr. Carl Alphonc  
343 Davis Hall  
alphonc@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM\*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

*OR request appointment via e-mail*

*\*Tuesday adjustments: 11:00 AM – 1:00 PM on 10/11, 11/1 and 12/6*

# ANNOUNCEMENTS

Undergraduate TA office hours will be ramped up to meet demand.

This week:

Kira - Tuesday at 5:00

Corwyn - Wednesday at 2:00

Steven - Thursday at 1:00

See “UTA Office Hours” table here:

[www.cse.buffalo.edu/faculty/alphonc/cse115/people.php](http://www.cse.buffalo.edu/faculty/alphonc/cse115/people.php)

Undergraduate TAs are students just like you.

Please respect their time: once office hours are over they are off the clock.

In particular, it's not cool to ask them course-related questions outside of their work hours (recitations/office hours).

## (a very early) EXAM 1 NOTICE

DATE: Tuesday October 4

TIME: 8:45 PM – 9:45 PM

LOCATION: various rooms in NSC

specific room/seat assignments to come

COVERAGE:

lecture material up to and including 9/23 (this week)

lab material up to and including lab 3 (next week)

readings: all assigned up to and including 3.2

HAVE A CONFLICT?

I will ask for documentation 9/26 – 9/30

BRING: your UB card

NO ELECTRONICS: cell phone, calculator, etc.

# ELECTRONICS: off & away

Last time

Live Eclipse demo

Today

class definitions in detail

variables revisited

variable scope & lifetime

method definitions

Coming up

class relationships

# REVIEW

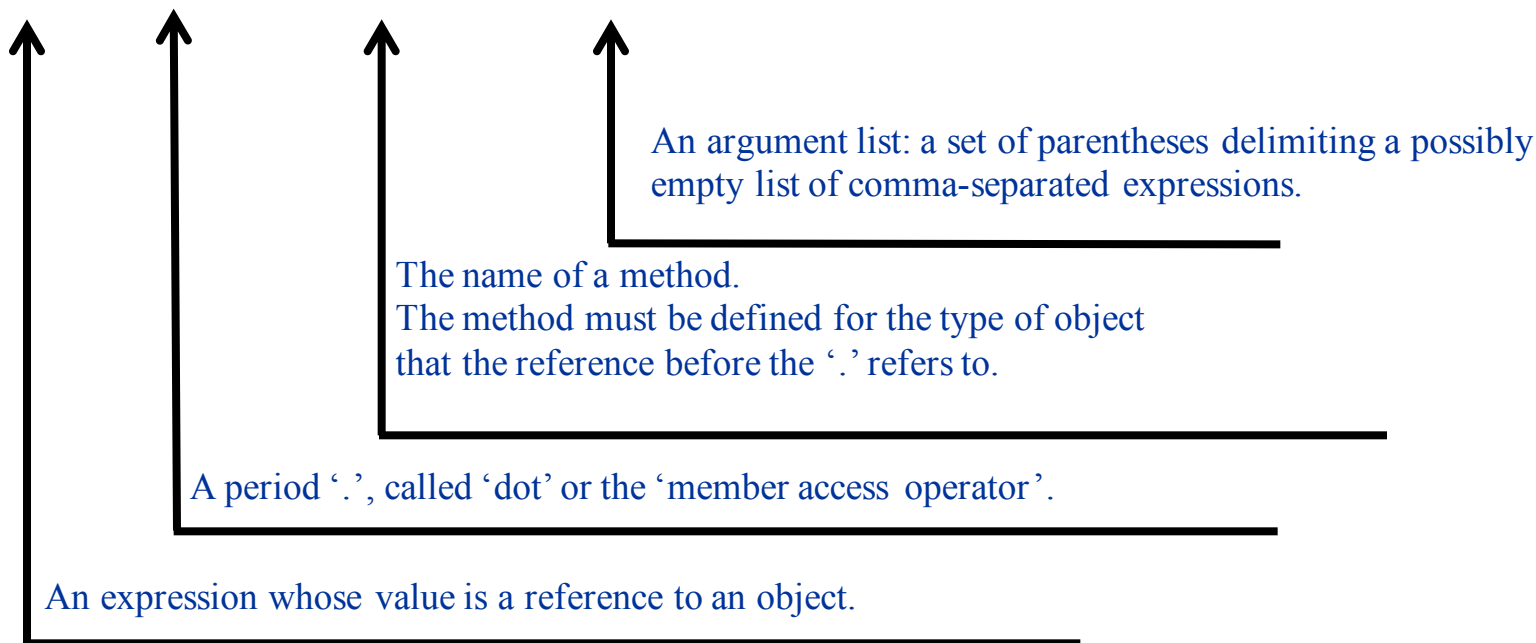


# Anatomy of a METHOD CALL

`<expr> . <method> ( )`

`<expr> . <method> ( <expr> )`

`<expr> . <method> ( <expr>, <expr>, ..., <expr> )`



# Minimal class definition

```
package lab2;

public class Farm {
    public Farm() {
    }
}
```

# Local variable declaration

```
package lab2;
```

```
public class Farm {
    public Farm() {
        example1.BarnYard by;
    }
}
```

A variable can be declared inside the body of a method. It is then called a *local variable*.

# Assignment statement

```
package lab2;
```

```
public class Farm {
```

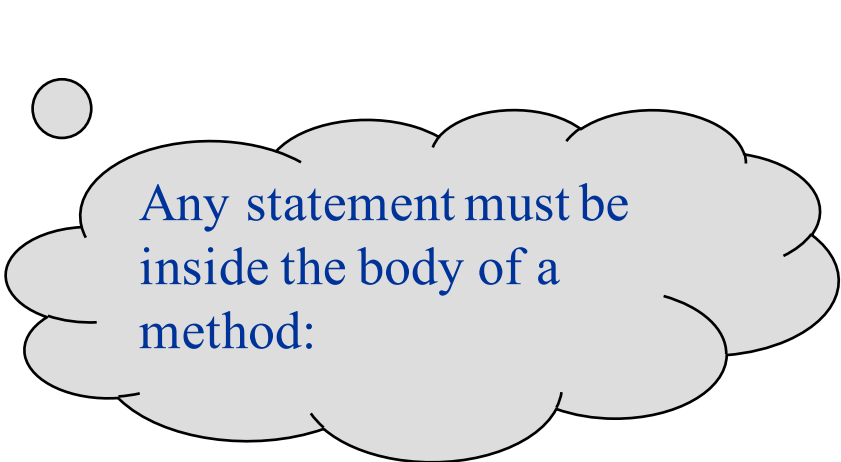
```
    public Farm() {
```

```
        example1.BarnYard by;
```

```
        by = new example1.BarnYard();
```

```
    }
```

```
}
```



Any statement must be  
inside the body of a  
method:

```
package lab2;

public class Farm {
    public Farm() {
        example1.BarnYard by;
        by = new example1.BarnYard();
        example1.Chicken c;
    }
}
```

Another assignment

```
package lab2;

public class Farm {
    public Farm() {
        example1.BarnYard by;
        by = new example1.BarnYard();
        example1.Chicken c;
        c = new example1.Chicken();
    }
}
```

```
package lab2;

public class Farm {
    public Farm() {
        example1.BarnYard by;
        by = new example1.BarnYard();
        example1.Chicken c;
        c = new example1.Chicken();
        by.addChicken(c);
    }
}
```

```
package lab2;

public class Farm {
    public Farm() {
        example1.BarnYard by;
        by = new example1.BarnYard();
        example1.Chicken c;
        c = new example1.Chicken();
        by.addChicken(c);
        c.start();
    }
}
```



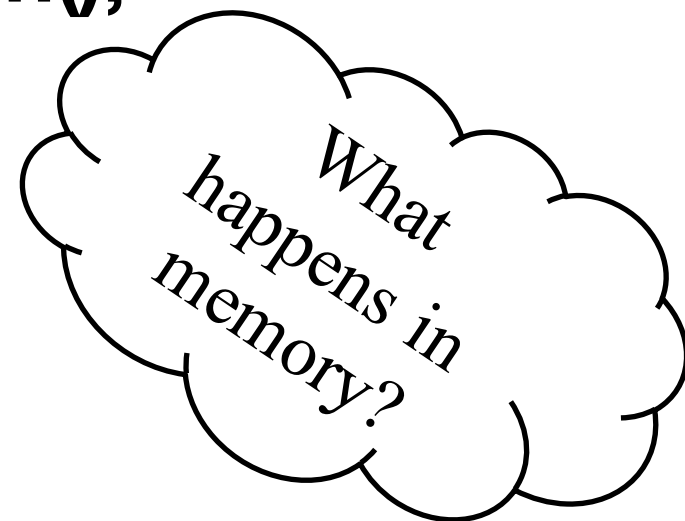
So what did it do?

Every time the class is instantiated, its constructor is executed.

Instantiating the `lab2.Farm` class creates a new `example1.BarnYard` object containing a new moving `example1.Chicken` object.

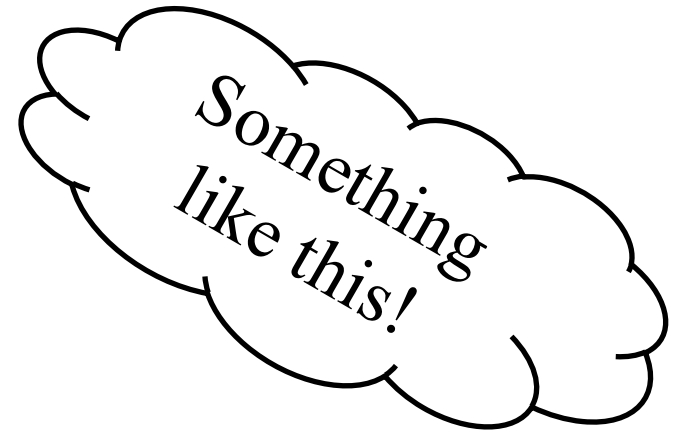
# MOVING ON

```
example1.BarnYard by;  
example1.Chicken c;  
by = new example1.BarnYard();  
c = new example1.Chicken();
```



```

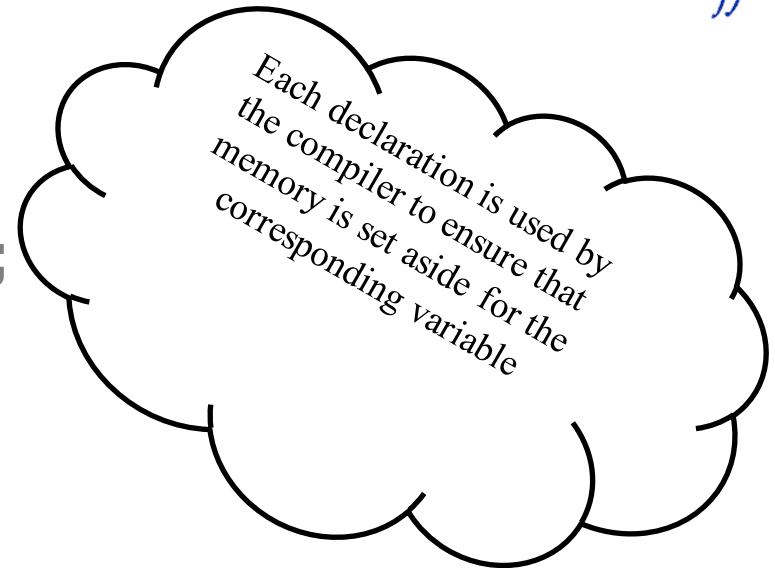
example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```



12203	<b>used</b>	497362	<b>available</b>
12204	<b>available</b>	497363	<b>available</b>
12205	<b>available</b>	497364	<b>available</b>
12206	<b>available</b>	497365	<b>available</b>
12207	<b>available</b>	497366	<b>available</b>
12208	<b>available</b>	497367	<b>available</b>
12209	<b>available</b>	497368	<b>available</b>
12210	<b>available</b>	497369	<b>available</b>
12211	<b>available</b>	497370	<b>used</b>

```

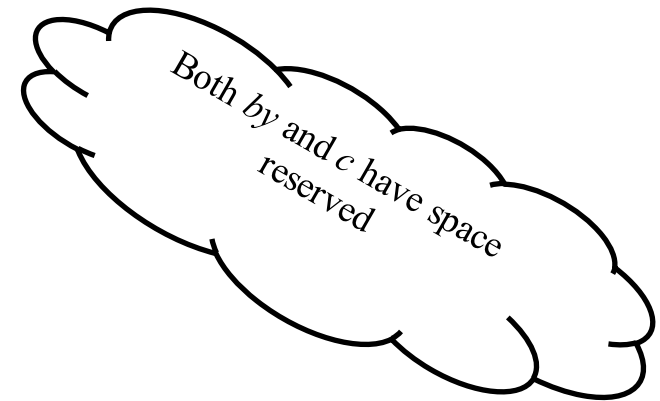
example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```



12203	<b>used</b>	497362	<b>available</b>
12204	<b>available</b>	497363	<b>available</b>
12205	<b>available</b>	497364	<b>available</b>
12206	<i>reserved for variable by</i>	497365	<b>available</b>
12207	<b>available</b>	497366	<b>available</b>
12208	<b>available</b>	497367	<b>available</b>
12209	<b>available</b>	497368	<b>available</b>
12210	<b>available</b>	497369	<b>available</b>
12211	<b>available</b>	497370	<b>used</b>

```

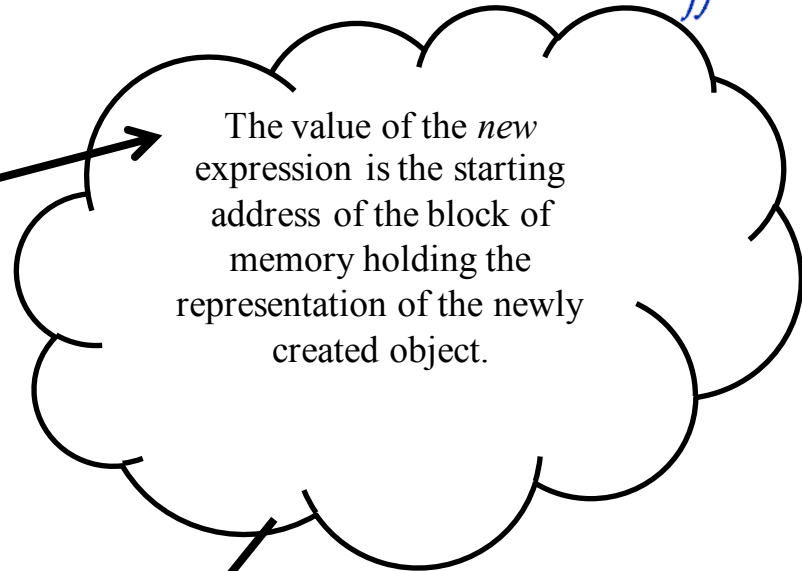
example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```



12203	<b>used</b>	497362	<b>available</b>
12204	<b>available</b>	497363	<b>available</b>
12205	<b>available</b>	497364	<b>available</b>
12206	<i>reserved for variable by</i>	497365	<b>available</b>
12207	<i>reserved for variable c</i>	497366	<b>available</b>
12208	<b>available</b>	497367	<b>available</b>
12209	<b>available</b>	497368	<b>available</b>
12210	<b>available</b>	497369	<b>available</b>
12211	<b>available</b>	497370	<b>used</b>

```

example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```



12203	<b>used</b>	497362	<b>available</b>
12204	<b>available</b>	497363	<b>available</b>
12205	<b>available</b>	497364	<b>available</b>
12206	<i>reserved for variable by</i>	497365	<b>available</b>
12207	<i>reserved for variable c</i>	<b>497366</b>	<i>example1.BarnYard object</i>
12208	<b>available</b>	497367	<i>example1.BarnYard object</i>
12209	<b>available</b>	497368	<i>example1.BarnYard object</i>
12210	<b>available</b>	497369	<i>example1.BarnYard object</i>
12211	<b>available</b>	497370	<b>used</b>

```

example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```

The assignment stores that value in the space set aside for the variable *by*

12203	<b>used</b>	497362	<b>available</b>
12204	<b>available</b>	497363	<b>available</b>
12205	<b>available</b>	497364	<b>available</b>
12206	<b>497366</b>	497365	<b>available</b>
12207	<i>reserved for variable c</i>	<b>497366</b>	<i>example1.BarnYard object</i>
12208	<b>available</b>	497367	<i>example1.BarnYard object</i>
12209	<b>available</b>	497368	<i>example1.BarnYard object</i>
12210	<b>available</b>	497369	<i>example1.BarnYard object</i>
12211	<b>available</b>	497370	<b>used</b>



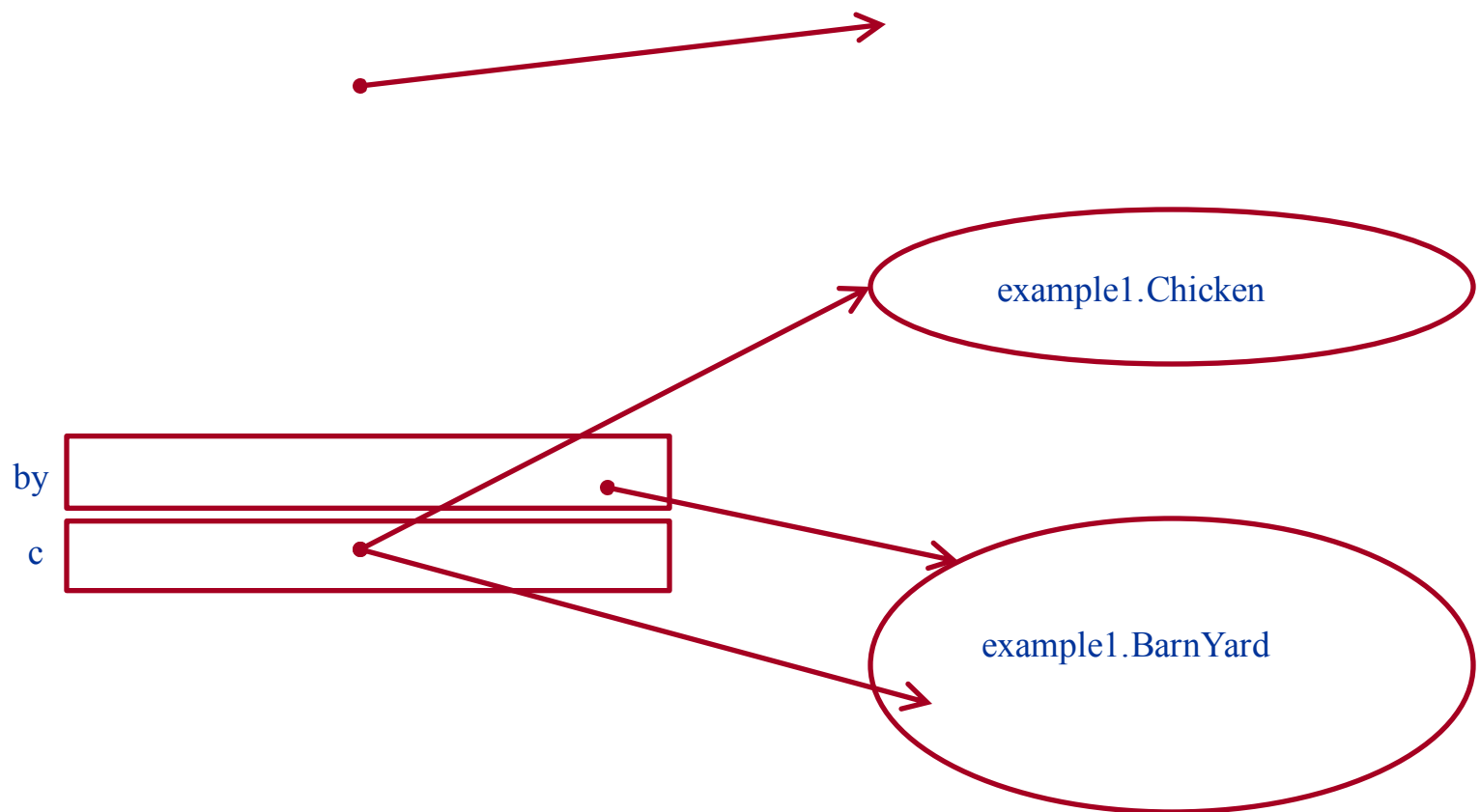
```

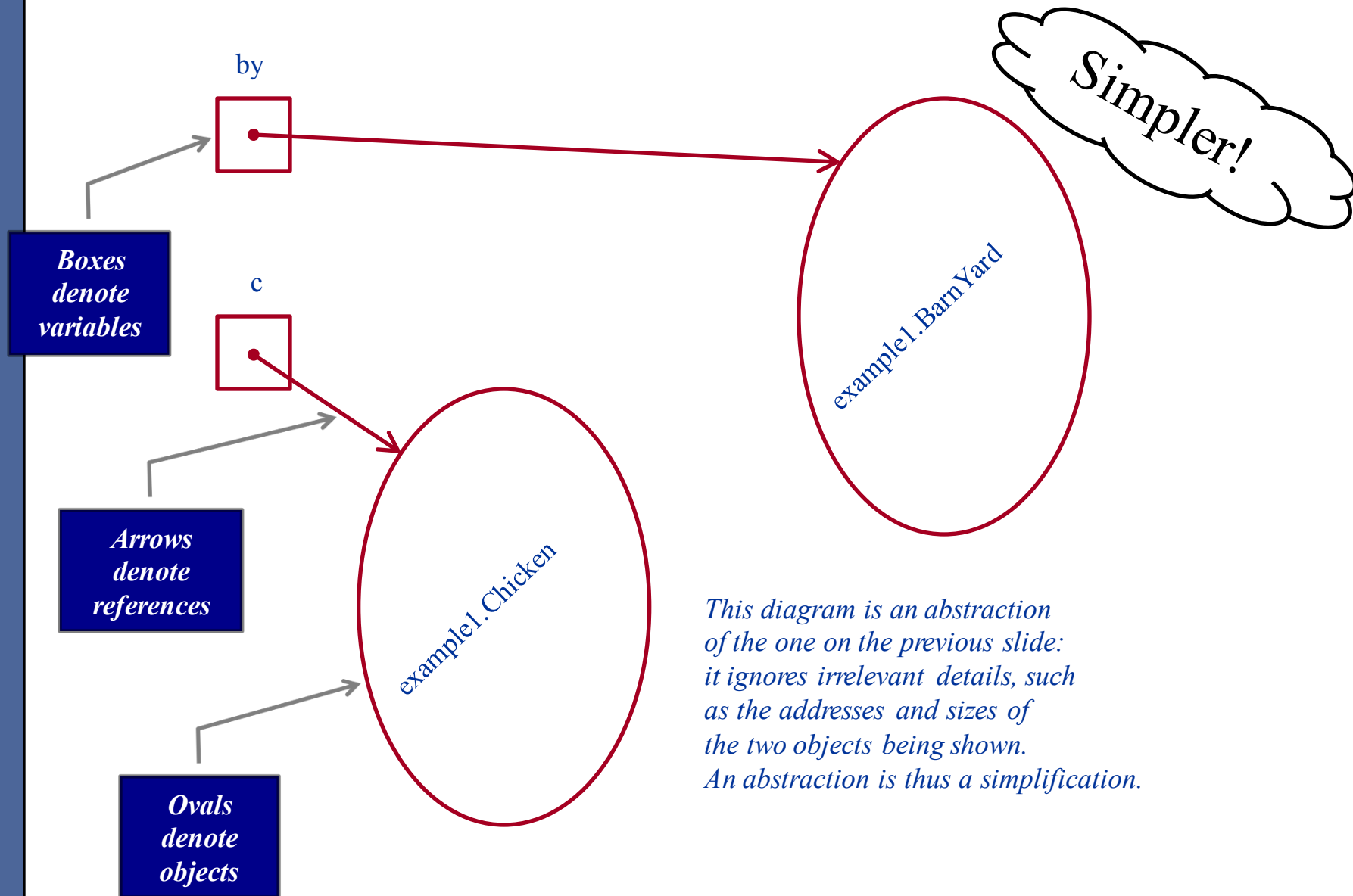
example1.BarnYard by;
example1.Chicken c;
by = new example1.BarnYard();
c = new example1.Chicken();
  
```

The same applies to the  
 creation of a new  
*example1.Chicken* object, and  
 the assignment of its  
 reference to the variable *c*

12203	<b>used</b>	<b>497362</b>	<i>example1.Chicken object</i>
12204	<b>available</b>	497363	<i>example1.Chicken object</i>
12205	<b>available</b>	497364	<b>available</b>
12206	<b>497366</b>	497365	<b>available</b>
12207	<b>497362</b>	497366	<i>example1.BarnYard object</i>
12208	<b>available</b>	497367	<i>example1.BarnYard object</i>
12209	<b>available</b>	497368	<i>example1.BarnYard object</i>
12210	<b>available</b>	497369	<i>example1.BarnYard object</i>
12211	<b>available</b>	497370	<b>used</b>

12203	<b>used</b>	497362	<i>example1.Chicken object</i>
12204	<b>available</b>	497363	<i>example1.Chicken object</i>
12205	<b>available</b>	497364	<b>available</b>
12206 <sub>by</sub>	<b>497366</b>	497365	<b>available</b>
12207 <sub>c</sub>	<b>497362</b>	497366	<i>example1.BarnYard object</i>
12208	<b>available</b>	497367	<i>example1.BarnYard object</i>
12209	<b>available</b>	497368	<i>example1.BarnYard object</i>
12210	<b>available</b>	497369	<i>example1.BarnYard object</i>
12211	<b>available</b>	497370	<b>used</b>





The afternoon lecture stopped here.

The morning lecture went over (quickly) the remaining slides.

We will review the remaining slides in both lectures on Wednesday before diving into variable scope and variable lifetime.

Class  
definition  
details

# Our first class definition!

Here's a minimal class definition. We will label and discuss each part of it in detail next class. For now we identify the major parts:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
        }  
}
```

Package declaration is shown in green:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```



`package` is a reserved word:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

`lab2` is the name of the package – you choose this

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

A semicolon ; marks the end of the declaration:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The class definition is shown in green:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The class definition consists of a **header**...

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

# Syntax

... and a **body**:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The class header consists of an **access control modifier** . . .

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

# Syntax

... the reserved word **class** ...

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```



# Syntax

... and a **class name**:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
        }  
}
```

The class body begins with an opening brace ‘{’ . . .

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
        }  
}
```

... and ends with the matching closing brace ‘}’:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
        }  
}
```

In this example, the body consists of a single **constructor definition**:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The constructor definitions consists of a **header** . . .

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

# Syntax

... and a **body**:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The constructor header consists of an **access control modifier**...

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

... the **constructor name** (which is the same as the class name) ...

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```



# Syntax

... and a **parameter list**:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

The constructor body begins with an opening brace ‘{’ . . .

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
    }  
}
```

... and ends with the matching closing brace ‘}’:

```
package lab2;
```

```
public class Farm {  
    public Farm() {  
        }  
}
```

# COMMENTARY

The morning lecture stopped here.