

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

Recitations are running (Baldy 21)

Bring your UB card

Main course website:

www.cse.buffalo.edu/faculty/alphonse/cse115/

zyBook usage

Exercises from book count towards grade
(minimum 10% of your course grade).

Book enrollment: 523 students

Course enrollment: 626 students

~16% of class not currently getting those points

ELECTRONICS: off & away

Last time

expressions and objects
memory diagram

Today

class definitions
variables
method calls
object diagrams

Coming up

variables in depth

REVIEW

OO software systems are systems of interacting objects.

Objects have

properties:

these are things that objects know

e.g. what you had for breakfast

behaviors:

these are things objects do

e.g. being able to reply to the question “What did you have for breakfast?”

evaluating `new example1.BarnYard()`

produces a value (which we call a reference)

causes a side effect (an object is created and initialized)

we can remember a reference value by storing it in a variable

evaluating a 'new' expression

When evaluating an expression like 'new example1.BarnYard()', the operator 'new' first determines the size of the object to be created (let us say it is four bytes for the sake of this example).

Next, new must secure a contiguous block of memory four bytes large, to store the representation of the object.

Bit strings representing the object are written into the reserved memory locations. In this example we use "10101010" to indicate that some bit string was written into a given memory location; the exact bit string written depends on the specific details of the object.

The **starting address** of the block of memory holding the object's representation is the value of the 'new' expression. This address is called a '**reference**'.

107

108

109

110

111

112

113

114

115

used

10101010

10101010

10101010

10101010

available

available

available

used

evaluating `new example1.BarnYard()`

produces a value (which we call a reference)

causes a side effect (an object is created and initialized)

we can remember a reference value by storing it in a variable

Variables must be declared before use

declaration specifies encoding scheme

declaration specifies size

Declaration consists minimally of

type

name

The semicolon ';' is a
terminator.

Examples

example1.BarnYard by ;

example1.Chicken c ;

To associate a value with a variable, use an assignment statement:

SYNTAX: <variable> = <expression> ;

‘=’ is the ASSIGNMENT OPERATOR (it is not ‘equals’!)

Example

by = new example1.BarnYard();

“by is assigned the value of the expression ‘new example1.BarnYard()’ ” ...or...

“by is assigned a reference to a new example1.BarnYard() object” ...or...

“by is assigned a reference to a new BarnYard object” (example1 is implied)

MOVING ON

Developers write *class definitions*.
placed in .java files

An object is an instance of a class.

Classes are instantiated at runtime.

What is a *class definition*?

A *class definition* is a description of the properties (variables) and behaviors (methods) that instances of the class will have.

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() {  
    }  
}
```

Packages

a package is an organizational mechanism
related classes are grouped together

allows class names to be re-used in different
packages

reduces chance of naming conflicts

one analogy:

package::class

area code::phone number

a class' *fully qualified name* consists of its package name and its (unqualified) class name:

`example1.BarnYard` is a fully qualified class name

`BarnYard` is an unqualified class name

each package corresponds to a folder (directory) in the file system

packages can be nested within each other
corresponds to nested folder on disk

examples:

java.rmi.registry

javax.swing.text.html.parser

com.sun.accessibility.internal.resources.accessibility

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() {  
    }  
}
```


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** . . .

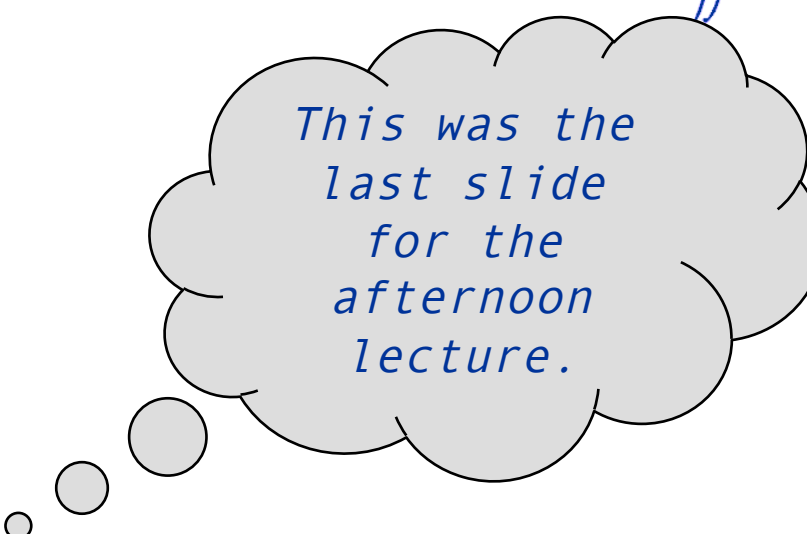
```
package lab2;
```

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

... and a **body**:

```
package lab2;
```

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



*This was the
last slide
for the
afternoon
lecture.*

object behaviors

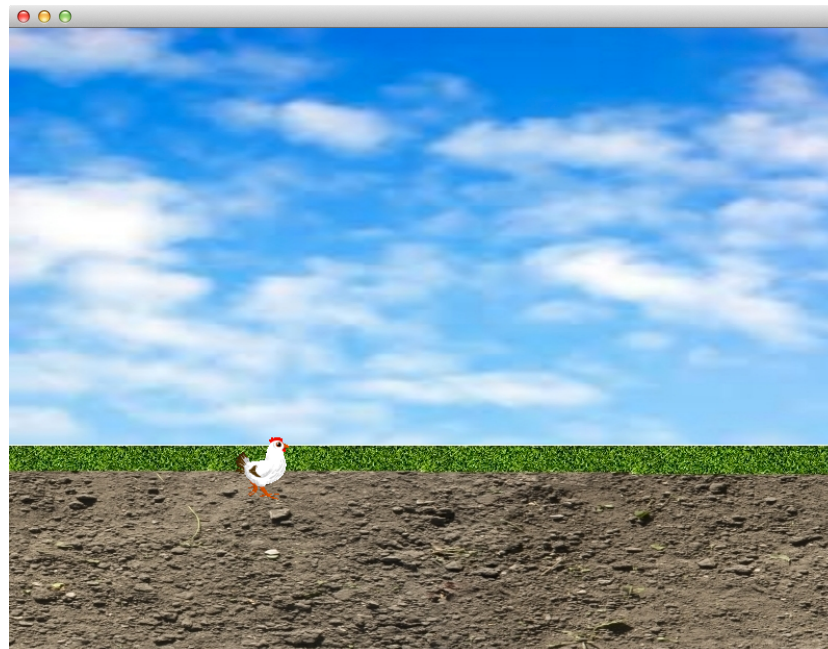
to ‘send a message’ to an object we call a method on the object

sometimes we say ‘**invoke** a method’ rather than ‘**call** a method’

To put an example1.Chicken object inside an example1.BarnYard object, call the “addChicken” method of the example1.BarnYard object with a reference to an example1.Chicken object.

A method is called using a reference to the object on which we call the method.

```
> new example1.BarnYard().addChicken(new example1.Chicken())
```



The method call from last time

```
new example1.BarnYard(). addChicken (new example1.Chicken())
```

Anatomy of a METHOD CALL

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



An expression whose value is a reference to an object.



A period '.', called 'dot' or the 'member access operator'.



The name of a method.
The method must be defined for the type of object
that the reference before the '.' refers to.



A set of parentheses with an expression inside.
This is an argument list.

Dissecting a method call

new example1.BarnYard().addChicken (new example1.Chicken())

An expression whose value is a reference to an object.

The 'member access operator'

The name of the method.

An argument list.
In this example the list
contains one expression.