CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce

343 Davis Hall

alphonce@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

(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

ROADMAP

Last time

class definitions in detail (terminology review)

variable scope & lifetime

Today

method definitions

Coming up

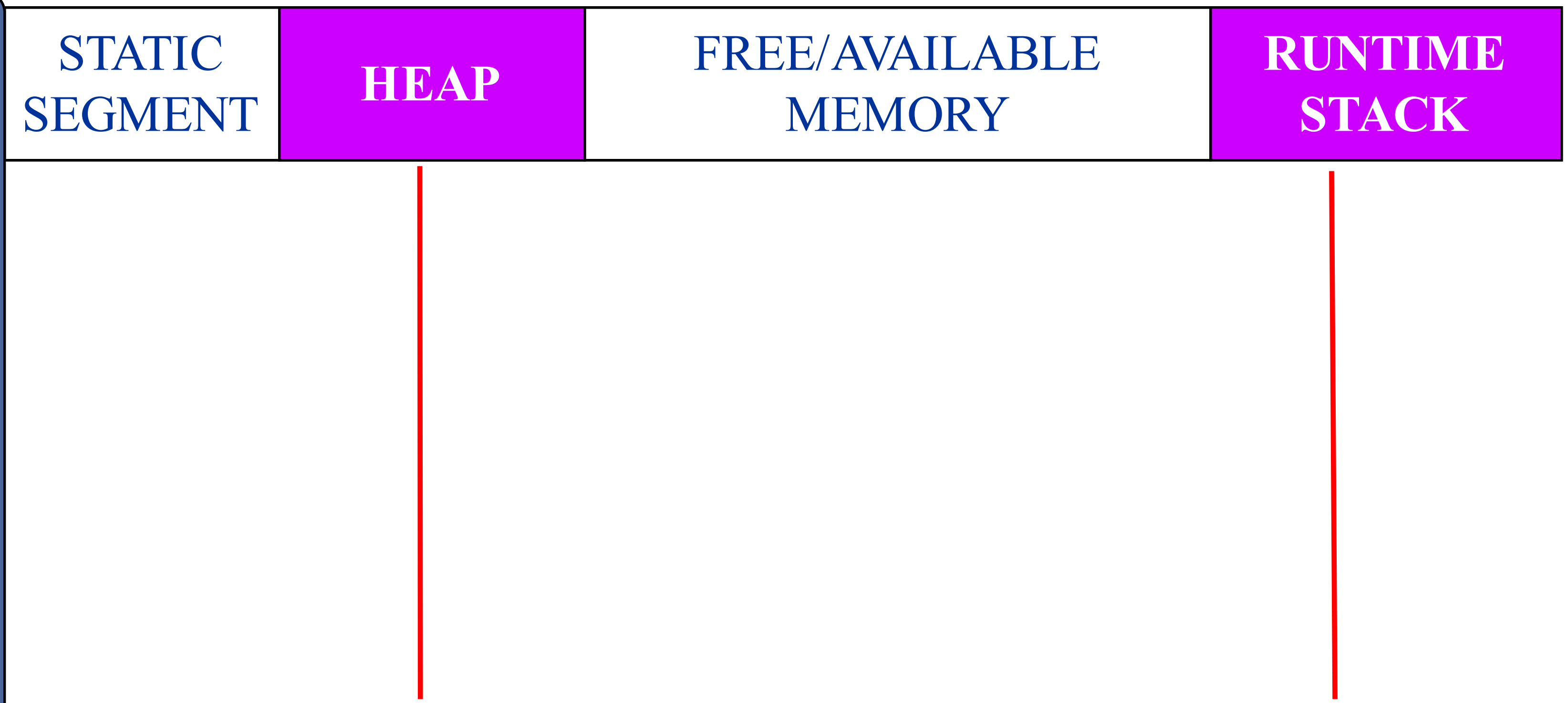
class relationships

REVIEW

Scope & Lifetime

	SCOPE	LIFETIME
LOCAL VARIABLE	From point of declaration to end of brace-delimited block containing the declaration For now think roughly: method body	From method invocation to method exit: the duration of a method call. For now, think roughly: short/fleeting
INSTANCE VARIABLE	class body	From object creation to object reclamation. For now, think roughly: long/persistent

Memory organization



All memory allocated by ‘new’ comes from the heap.

Objects are allocated space by ‘new’, and their representations (which contain their **instance variables**) therefore exist on the heap.

Local variables are stored on the runtime stack. Each method invocation (call) results in an invocation record (stack frame) being added to the top of the stack. When a method exits, its invocation record is removed from the top of the stack.

We've seen this code before

```
package demo;
public class Farm {

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

}
```

All the code in the Farm constructor executes whenever a new Farm object is created.

What if we want to be able to add moving Chickens to the Farm's BarnYard at a later point in time?

syntax

```
package demo;
public class Farm {

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

Move the Chicken creating code into its own <u>method.</u>

A constructor can be called ONLY to create a new object. It cannot be invoked on an existing object.

A method can be called only on an already existing object.

syntax

```
package demo;
public class Farm {
        private example1.BarnYard _by;
        public Farm( ) {
                _by = new example1.BarnYard( );
        }
        public void addMovingChicken( ) {
                example1.Chicken  c;
                c = new example1.Chicken( );
                _by.addChicken(c);
                c.start( );
        }
}
```

Make _by an instance variable so that it has scope thoughout entire class body.

syntax

```
package demo;
public class Farm {
        private example1.BarnYard _by;
        public Farm( ) {
                this._by = new example1.BarnYard( );
        }
        public void addMovingChicken( ) {
                example1.Chicken  c;
                c = new example1.Chicken( );
                this._by.addChicken(c);
                c.start( );
        }
}
```

Technically only the variables in a method's invocation record can be directly accessed in a method. 'this' is an implicit variable in each method which holds a reference to the object on which the method was called.

```
public void addMovingChicken ( ) {
        …declarations & statements...
}
```

‘void’ is a return type specification. It indicates that this method does not return a value when called.

syntax

```
public void addMovingChicken ( ) {
        …declarations & statements...
}
```

'addMovingChicken' is the name of the method. We get to choose that.

```
public void addMovingChicken ( ) {
        …declarations & statements...
}
```

'( )' is the parameter list of the method. In this case the parameter list is empty.

syntax

```
package demo;
public class Farm {
        private example1.BarnYard _by;
        public Farm( ) {
                _by = new example1.BarnYard( );
        }
        public void addMovingChicken( ) {
                example1.Chicken  c;
                c = new example1.Chicken( );
                _by.addChicken(c);
                c.start( );
        }
}
```

'this' is usually left implicit.

The compiler can usually figure out where to put 'this'.