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
DATE: Tuesday October 4
TIME: 8:45 PM – 9:45 PM
LOCATION: various rooms - assignments on Friday
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
BRING: your UB card
NO ELECTRONICS: cell phone, calculator, etc.
IF YOU HAVE A CONFLICT

send me e-mail:
alphonce@buffalo.edu

use this subject line:
[CSE115] Exam 1 conflict

attach documentation of conflict
(e.g. screenshot of class schedule that has your name and the conflict)

no later than:
9:00 PM on Wednesday Sept 28
Extra office hours have been added Th/Fr this week

See PEOPLE page of course website

We are arranging for exam review sessions on the weekend – stay tuned for room/date/time details

EXAM 1 REVIEW SESSIONS:

Sat Oct 1 2016 4:00PM - 5:30PM in Davis 101
Mon Oct 3 2016 5:00PM - 6:20PM in Knox 110
ELECTRONICS: off & away
Last time
Relationships
  composition
  association

Today
Relationships (continued)
  association
  accessor/mutator methods

Coming up
Relationships (continued)
REVIEW
Clifford’s relationship to his tail
Clifford has the same tail throughout his life

Clifford’s relationship to his collar
Clifford is associated with different collars throughout his life
Dog-Tail relationship is COMPOSITION
Dog takes responsibility for creating a Tail

Dog-Collar relationship is ASSOCIATION
Dog takes NO responsibility for creating Collar
No necessary lifetime link between the two objects involved

Two implementations:

The first is very similar to composition, but differs in one crucial respect: where the target class is instantiated.

The second, which decouples lifetimes completely, is a bit more complex but also more flexible.
3 changes to source class:

1. Declaration of instance variable
2. Assignment of *existing* instance to the instance variable
3. Parameter of constructor is of same type as instance variable
public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}

Dog – Collar example in Java
MOVING ON
local variable declared in parameter list

parameter list appears in method header

due of parameter is determined at method call

the value of the argument expression is assigned to the parameter on entry to the method

multiple parameter declarations are separated by commas in the parameter list

(Actor a, Director d, Screenwriter s)
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
small

fido
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
Collar object

small

fido
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
Dog object

Collar object

small

_fido

_myCollar
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
Collar object

Dog object

_small

_fido

_c

_myCollar
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);  // (argument: a value)
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {  // (parameter: a local variable)
        _myCollar = c;
    }
}
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
public class SomeClass {
    public void someMethod() {
        Collar small;
        Dog fido;
        small = new Collar();
        fido = new Dog(small);
    }
}

public class Dog {
    private Collar _myCollar;
    public Dog(Collar c) {
        _myCollar = c;
    }
}
ASSOCIATION

(general implementation)
A method which changes the value of an instance variable.

 Allows us to grant WRITE access to the contents of a variable which itself is PRIVATE.
public class Dog {

    private Collar _collar;

    public Dog(Collar c) {
        _collar = c;
    }

    public void setCollar(Collar c) {
        _collar = c;
    }

}
public class Dog {

    private Collar _collar;

    public Dog(Collar c) {
        _collar = c;
    }

    public void setCollar(Collar c) {
        _collar = c;
    }

}
public class Dog {

    private Collar _collar;

    public Dog(Collar c) {
        _collar = c;
    }

    public void setCollar(Collar c) {
        _collar = c;
    }
}
public class Dog {
    private Collar _collar;
    private Sweater _sweater;
    private Tail _tail;
    public Dog(Collar c, Sweater s) {
        _collar = c;
        _sweater = s;
        _tail = new Tail();
    }
    public void setCollar(Collar abc) {
        _collar = abc;
    }
    public void setSweater(Sweater q) {
        _sweater = q;
    }
}

Similarities:
both set the value of an instance variable

Differences:
constructor sets value of an instance variable when the class is instantiated
mutator sets the value of an instance variable after the object already exists

constructor initializes ALL instance variables
mutator sets the value of just one instance variable
Return statement consists of the keyword “return”, followed by an expression whose type matches the given return type.

```java
public class Farm {
    private example1.BarnYard _t;
    public Farm() {
        _t = new example1.BarnYard();
    }
    public example1.BarnYard getBarnYard() {
        return _t;
    }
}
```
A void method has no return value, and **the method call is not an expression** (*)

A non-void method has a return value, and **the method call is an expression whose value is the returned value**

* Technically not quite true – void is a type, whose sole value is also called void. Some languages call the type void by the name Unit. Its only role in Java is as the return type specification of methods which do not return a value.
A method which returns the value of an instance variable

Allows us to grant READ access to the contents of a variable which itself is PRIVATE.
public class Farm {
    private example1.BarnYard _t;
    public Farm() {
        _t = new example1.BarnYard();
    }

    public example1.BarnYard getBarnYard() {
        return _t;
    }
}

Return type specification is the type of the returned value, `example1.BarnYard` in this case.

```java
class Farm {
    private example1.BarnYard _t;
    public Farm() {
        _t = new example1.BarnYard();
    }
    public example1.BarnYard getBarnYard() {
        return _t;
    }
}
```