CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Thursday 12:00 PM – 2:00 PM
Friday 8:30 AM – 10:30 AM
OR request appointment via e-mail
Turn off and put away electronics:

cell phones
pagers
laptops
tablets
etc.
Last time
parameterless void methods

Today
composition

Coming up
association
interfaces
package code;

public class Farm {
    private example1.BarnYard _by;
    
    public Farm() {
        _by = new example1.BarnYard();
    }

    public void addTwoChickens() {
        example1.Chicken c;
        c = new example1.Chicken();
        c.start();
        _by.addChicken(c);
        c = new example1.Chicken();
        c.start();
        _by.addChicken(c);
    }
}

© Dr. Carl Alphonce
RELATIONSHIPS
relationships exist between objects in problem domains

want to capture those relationships in our models and express them in our code
Computing and Clifford
Clifford’s relationship to his collar
Clifford is associated with different collars throughout his life

Clifford’s relationship to his tail
Clifford has the same tail throughout his life
COMPOSITION
(whole-part)
A whole-part relationship (e.g. Dog-Tail)

Whole and part objects have same lifetime

when whole is created, it has its parts

when whole is destroyed, parts go away too
In code, this involves 3 changes to whole class:

1. Declaration of *instance* variable of part type
2. Instantiation of part class in whole class constructor
3. Assignment of new part instance to *instance* variable
Whole has responsibility for creating its parts (which is why instantiation of parts happens in constructor of whole).

Whole can communicate with parts. This is why an instance variable is declared: to establish a name for the newly created object.
public class Dog {
    private Tail _tail;
    public Dog() {
        _tail = new Tail();
    }
}
channel 1
What kind of declaration is this:
private Tail _tail;

A. method
B. class
C. local variable
D. instance variable
What kind of declaration is this:

private Tail _tail;

A. method
B. class
C. local variable
D. instance variable
What kind of declaration is this:

private Tail _tail;

A. method
B. class
C. local variable
D. instance variable  ✔
What kind of declaration is this:

private Tail _tail;

A. method
B. class
C. local variable
D. instance variable
UML = Unified Modeling Language

express design without reference to an implementation language

Examples:
Binary \( \rightarrow \) two classes are involved

**source class** has code modification

**target class** does not

composition

source: WHOLE

target: PART

in diagram:

line decoration is on source/WHOLE

show only detail that’s needed/desired
package cse115;

public class Dog {
    public Dog() {
    }
}

package cse115;

public class Tail {
    public Tail() {
    }
}
With composition

```java
package cse115;

public class Dog {
    private Tail _tail;
    public Dog() {
        _tail = new Tail();
    }
}
```

```java
package cse115;

public class Tail {
    public Tail() {
    }
}
```

Decoration (a diamond) is on source side of line for composition

Source class declares an instance variable of the target class type

Source class instantiates the target class in its constructor

In the source class’ constructor a reference to a new instance of the target class is assigned to the instance variable.

Relationship line for composition

Source class

Target class
names can consist of

- UPPER CASE LETTERS
- lower case letters
- digits (0 – 9)
- the underscore, ‘_’
- first character cannot be a digit

this is a slight simplification of the actual rules

- they are the truth
- they are not the whole truth
packages:
  no upper case letters used (this_is_an_example)

classes (and constructor names):
  first character: upper case letter
  camel case afterwards (ThisIsAnExample)

instance variables:
  first character: underscore ‘_’
  second character: lower case
  camel case afterwards (_thisIsAnExample)

local variables (and method names):
  first character: lower case
  camel case afterwards (thisIsAnExample)
channel 1
In an object diagram, what does a square denote?

A. class
B. constructor
C. object
D. label
E. package
F. reference
G. relationship
H. variable
In an object diagram, what does a square denote?

A. class  
B. constructor  
C. object  
D. label  
E. package  
F. reference  
G. relationship  
H. variable

Convince your neighbor your answer is correct.
In an object diagram, what does a square denote?

A. class
B. constructor
C. object
D. label
E. package
F. reference
G. relationship
H. variable ✅
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.