ECE 3574: Applied Software Design

Composition

Today we will learn how to build up complex concepts and models from simple parts.

- Composition models “has-a”
- Examples
- Composition and the Law of Demeter
- Composition and Qt
- Exercise

Composition is a major way of modeling has-a relationships

A composite type has member variables that correspond to its components.

```
class Foo
{
  ComponentType component;
}
```

Classic Example: People, Employees, and Customers

A Person has-a

- name
- age
- address

An Employee is-a Person and has-a

- id
- role
- salary

Classic Example: People, Employees, and Customers

A Person has-a

- name (first/last?)
- age (possibly unknown?)
- address (format?)

An Employee is-a Person and has-a

- id (unique?)
- role (static or dynamic?)
- salary (currency?)

Is a customer always a person?

Prefer Composition to Inheritance

Inheritance is overused and leads to tight coupling.

Composition

- gives the most flexibility with least coupling
- shorter compile times, a member can be a pointer, thus only declared
- less error prone, no private/protected/public

Use inheritance only when you need to implement is-a relationships that require polymorphism.

Sometimes has-a is just as good as is-a

Consider the Employee is-a Person.

A Person could also have-a Job.

Ontology

Ontology is the name used for defining objects and their relationships.

Composition and Inheritance in C++ gives us the primary means to model the world.

Each problem domains have their own ontologies

Composition is very useful in GUI design

For example, a window has-a

- menu
- controls
- view

A menu has-a . . .

A control has-a . . .

A view has-a . . .

Exercise

See website

Next Actions and Reminders

- Read about Qt Event System