Object-Oriented Design

Goals of Object-Oriented Design

Robustness
- Complex programs should operation correctly
- Should deal with improper inputs and conditions

Adaptability
- Software grows over a long lifetime
- May run on different generations and makes of hardware

Reusability
- Building from reusable pieces avoids "reinventing the wheel"

Design Principles

Abstraction
- Intuitive, high-level interface promotes understandable and correct implementations

Encapsulation
- Interface hides implementation details
- Allows designer more freedom and user does not need to worry about low-level details

Modularity
- Organized functional units may be connected together to build more complex software

Polymorphism

Ability of variable to take on many forms
- Class variable may contain exact class or any descendent
- Interface variable may contain any class implementing the interface

Allows for greater modularity

Using Inheritance

Specialization
- Handle differences in behavior between parent and child for the same task
- Override some parent methods
  - —Refinement: call parent method and then do something extra
  - —Replacement: just do something different

Extension
- Add to the functionality of parent by adding new data and behaviors

(Real examples often do some of both)

"Is a" and "Has a" Relationships

"Is a"
- One object is a specialized example of another
- Example: museum is a building
- Often implemented by inheritance

"Has a"
- One object is a component of another
- Example: building has a door
- Often implemented by one object having another as a field

Adapter Pattern

Implements "Is a" relationship without inheritance

- **One class has another as a field**
- **"Forward" all methods of the field to the larger class**
- **Useful for multiple inheritance and for implementing interfaces**

Adapter Pattern Example

```
interface Driveable { drive() };
class Vehicle implements Driveable {
   drive(){...complicated code...};
}
class Automobile implements Driveable {
   Vehicle v;
   drive() { ... v.drive() ... };
}
```


In-Class Exercise

Groups of 3 or 4 people

Specify some useful fields and methods for:

- **Human, man, woman, parent, child**

Organize using classes and interfaces

Then I'll ask some of you to share with the class