



## **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"

> Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen



# Design Principles

## Abstraction

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

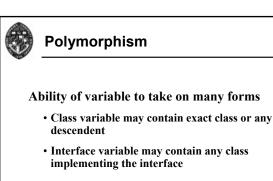
## Encapsulation

- 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

Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen



Allows for greater modularity

Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen



## 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)

Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Coher

## "Is a" and "Has a" Relationships

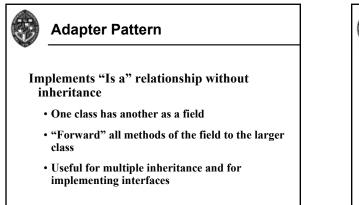
## "Is a"

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

## "Has a"

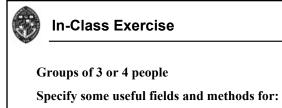
- · One object is a component of another
- Example: building has a door
- Often implemented by one object having another as a field

Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Coher



Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen

Adapter Pattern Example
<pre>interface Driveable { drive() };</pre>
class Vehicle implements Driveable {
<pre>drive(){complicated code};</pre>
}
class Automobile implements Driveable {
Vehicle v;
<pre>drive() { v.drive() };</pre>
}
Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen



• Human, man, woman, parent, child

Organize using classes and interfaces

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

Johns Hopkins Department of Computer Science Course 600.226: Data Structures, Professor: Jonathan Cohen