ADVANCED TOPICS

Dealing with larger systems

LARGER SYSTEMS

Most systems larger than single object

Most systems touch the outside world

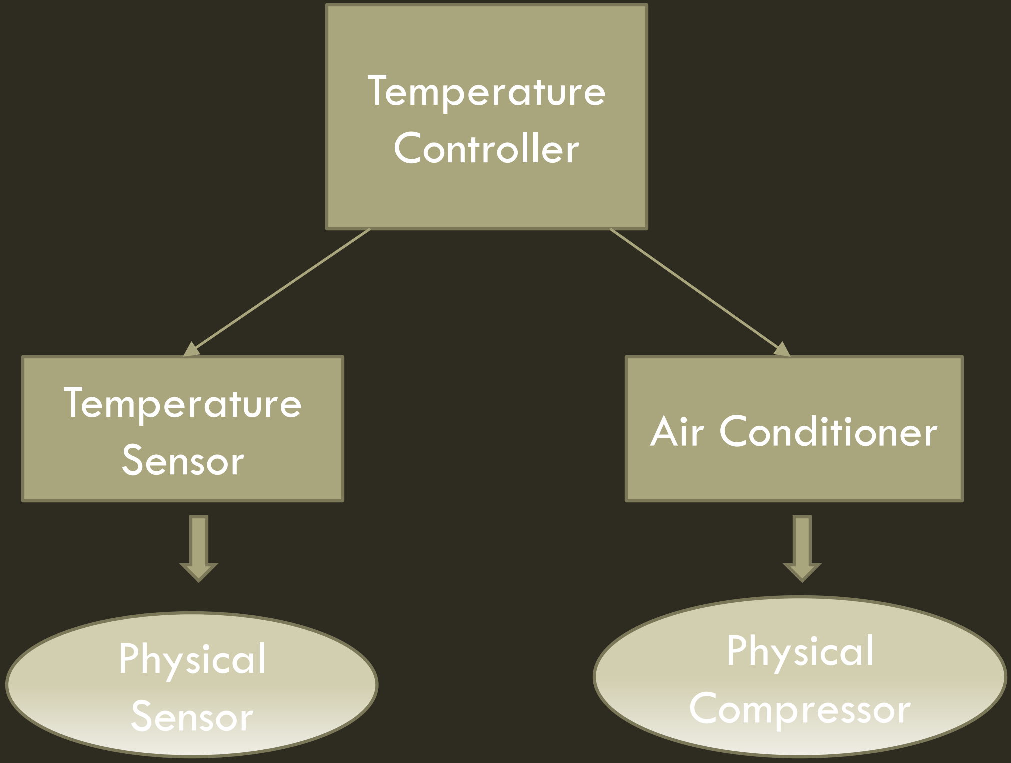
- GUIs, databases, web services

Harder to independently exercise

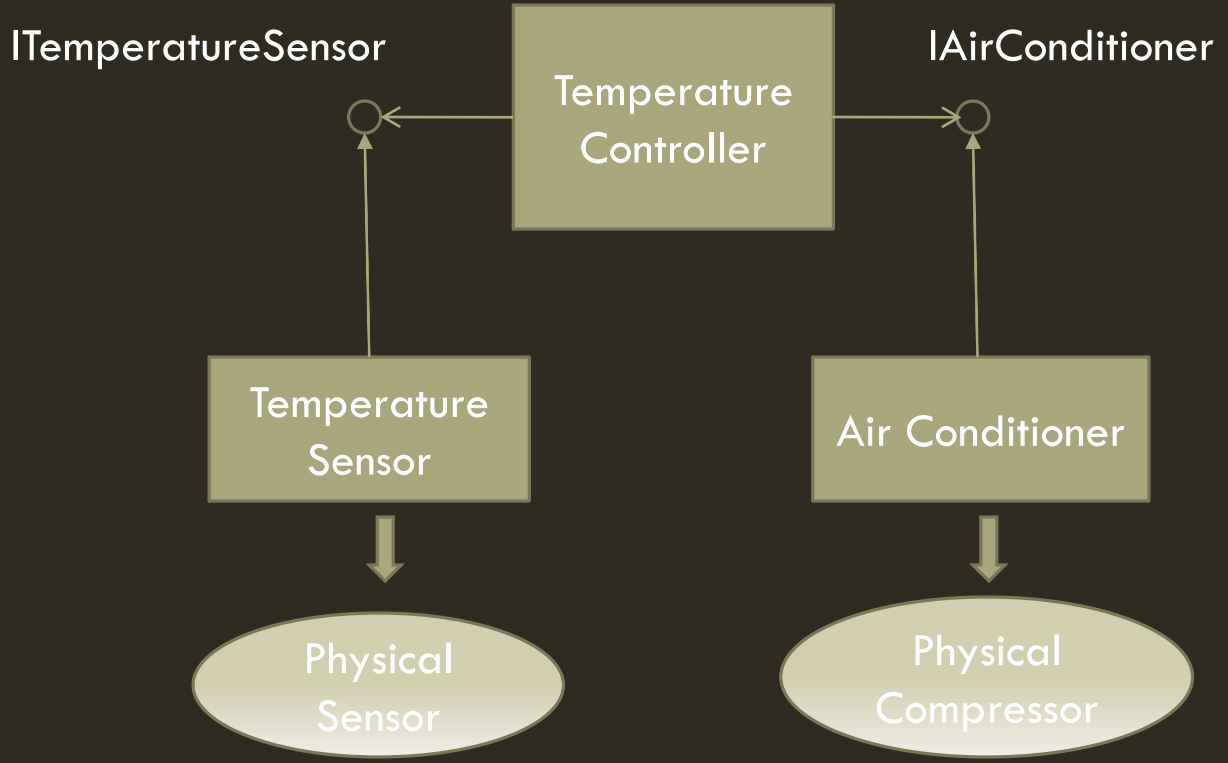
- State dependencies
- Visual testing
- Slow

Isolation creates independence

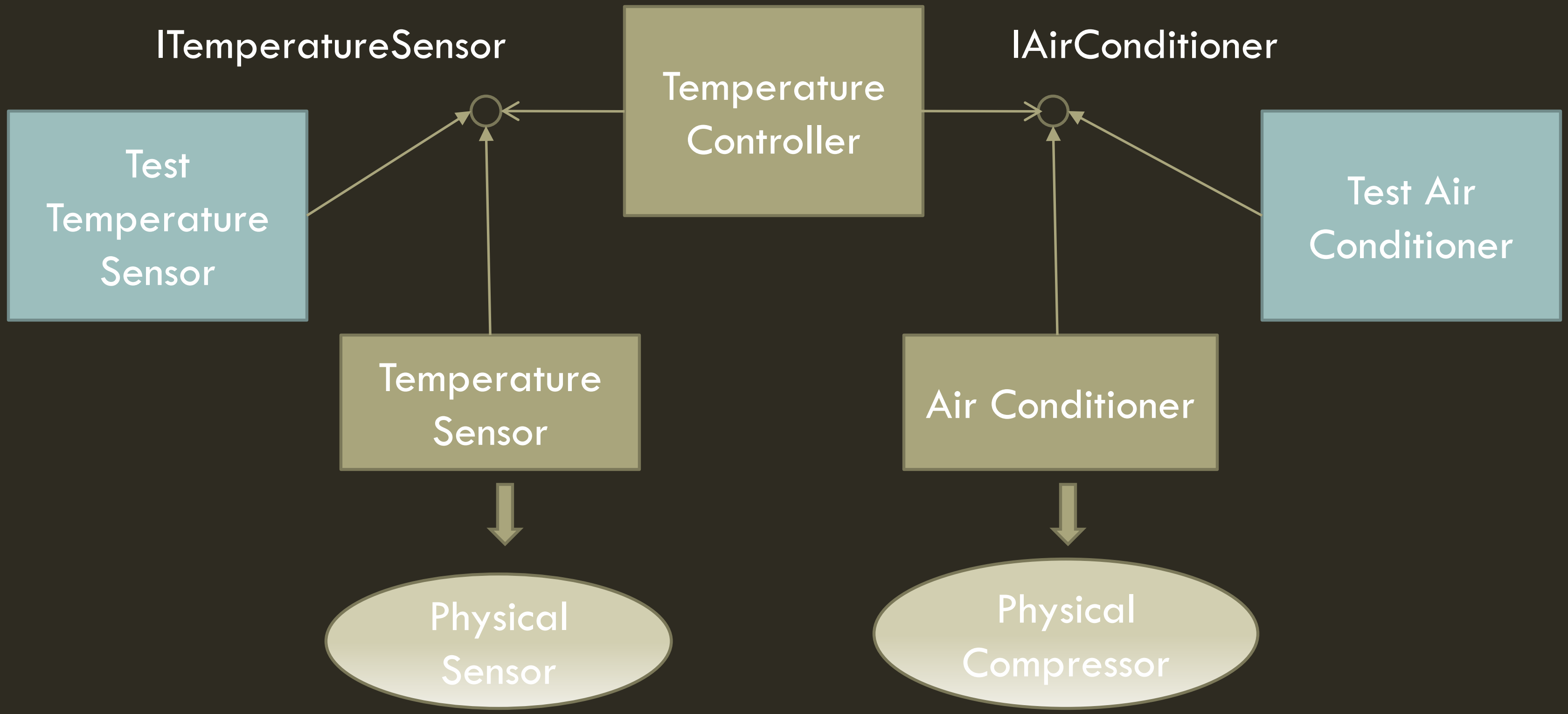
ISOLATION PROMOTES GOOD DESIGN



ISOLATION PROMOTES GOOD DESIGN



ISOLATION PROMOTES TESTABILITY



ISOLATION PROMOTES TESTABILITY

Test Doubles

- Take advantage of isolation
  - State
  - Behavior
  - Speed
- Take the place of real objects
  - Look like
  - Act like

STUBS, FAKES, AND MOCKS

Stubs

- Hand coded for single purpose

Fakes

- Slimmed down versions of real classes

Mocks

- Larger testing frameworks that take over your test for you

STUBS

Hand-coded Test Doubles

- Return value
- Record value

```
public interface IAirConditioner
{
    void Cool();
}

public class StubAirConditioner :
IAirConditioner
{
    public bool WasTurnedOn;

    public void Cool()
    {
        WasTurnedOn = true;
    }
}
```

```
public interface IStubTemperatureSensor
{
    int CurrentTemperature { get; }
}

public class StubTemperatureSensor : IStubTemperatureSensor
{
    public StubTemperatureSensor(int temperatureToReport)
    {
        CurrentTemperature = temperatureToReport;
    }

    public int CurrentTemperature { get; private set; }
}
```

MOCK FRAMEWORKS

Useful for testing interactions

- Define expectations of object interactions
- Exercise system
- Confirm expectations were fulfilled

STUB VERSUS MOCK COMPARISON – STUB

```
[Fact]
public void TurnedOnIfCurrentTemperatureIsMore ()
{
    const int temperatureToReport = 75;
    const int desiredTemperature = 72;

    StubTemperatureSensor sensor = new StubTemperatureSensor
       (temperatureToReport);
    StubAirConditioner airConditioner =
            new StubAirConditioner     ();
    TemperatureController controller = new
            TemperatureController(sensor, airConditioner);

    controller.AdjustTemperature(desiredTemperature);

    Assert.True(airConditioner.WasTurnedOn);
}
```

STUB VERSUS MOCK COMPARISON - MOCK

```
[Fact]
public void AirConditionerTurnedOnIfCurrentTemperatureIsMoreThanDesired_Moq()
{
    const int temperatureToReport = 75;
    const int desiredTemperature = 72;

    var mockSensor = new Mock<ITemperatureSensor>();
    var mockAirConditioner = new Mock<IAirConditioner>();
    TemperatureController controller = new TemperatureController                    (mockSensor.Object, 
mockAirConditioner.Object);

    mockSensor.SetupGet(sensor => sensor.CurrentTemperature).Returns
         (temperatureToReport).Verifiable();
    mockAirConditioner.Setup(ac => ac.Cool()).Verifiable();

    controller.AdjustTemperature(desiredTemperature);

    mockSensor.VerifyAll();
    mockAirConditioner.VerifyAll();
}
```