

Embedding Calling Protocols into the Builder



Zoran Horvat

OWNER AT CODING HELMET CONSULTANCY


@zoranh75 www.codinghelmet.com



Throw...

```
public class PersonBuilder
{
    ...
    public void Add(IContactInfo contact)
    {
        if (this.Contacts.Contains(contact))
            throw new ArgumentException();

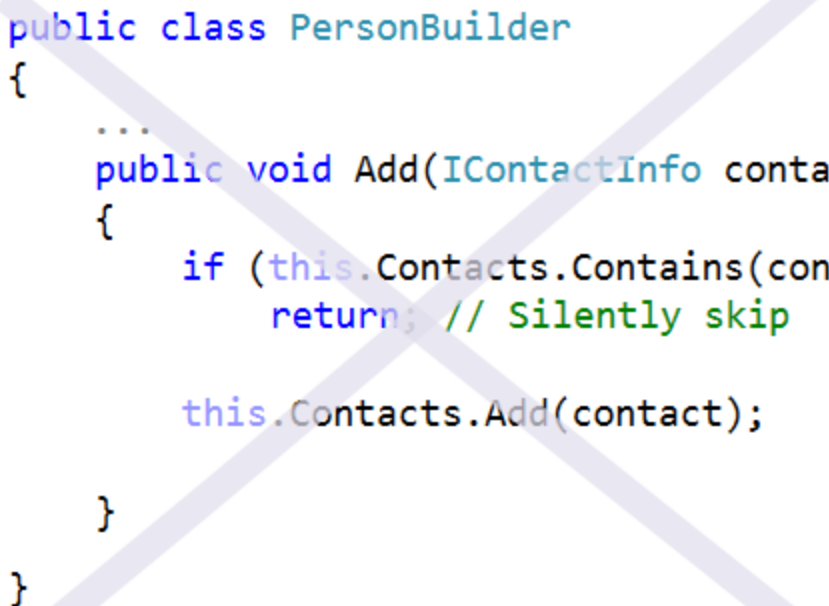
        this.Contacts.Add(contact);
    }
}
```



... or ignore

```
public class PersonBuilder
{
    ...
    public void Add(IContactInfo contact)
    {
        if (this.Contacts.Contains(contact))
            return; // Silently skip

        this.Contacts.Add(contact);
    }
}
```

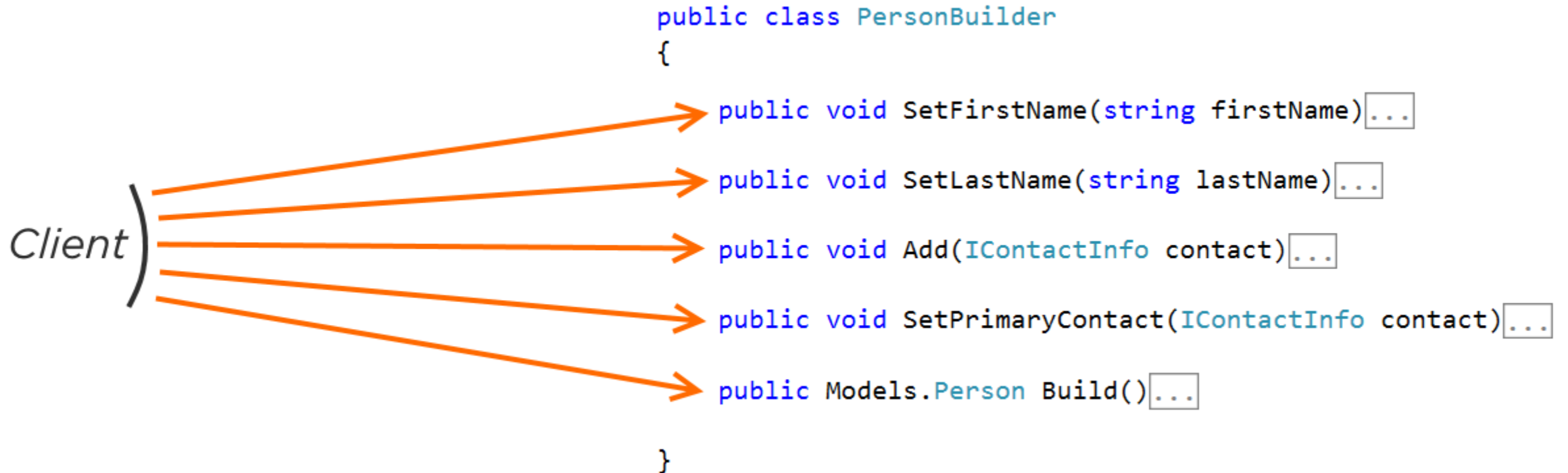


5x {

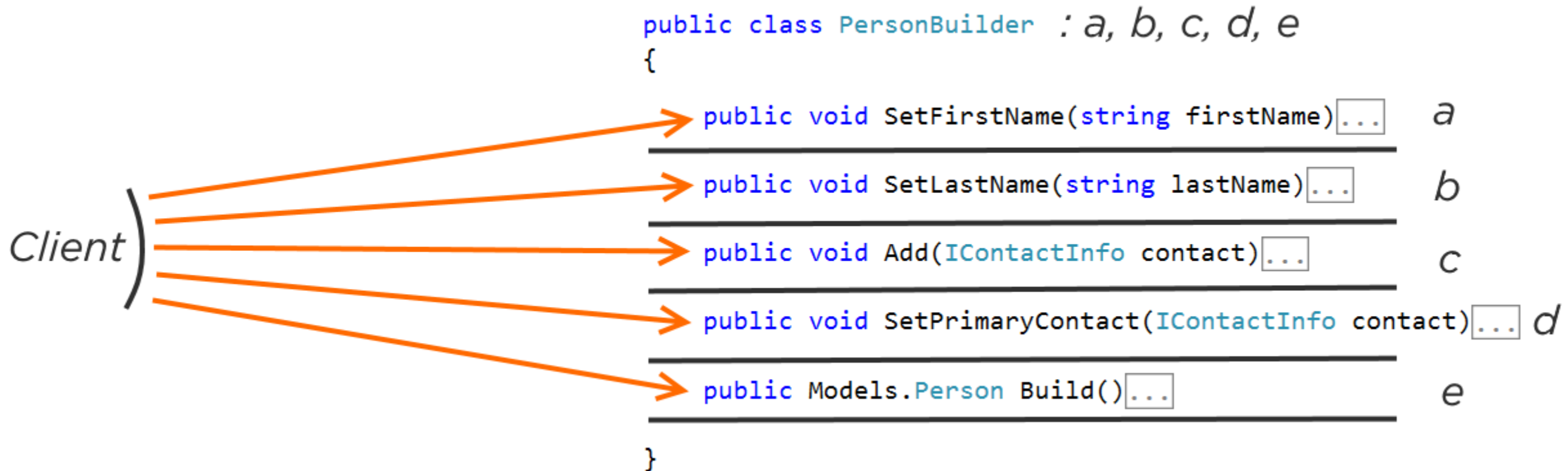
```
builder.Add(one);
builder.Add(two);
builder.Add(three);
builder.Add(four);
builder.Add(four);
```

*Duplicate call
Now what?*

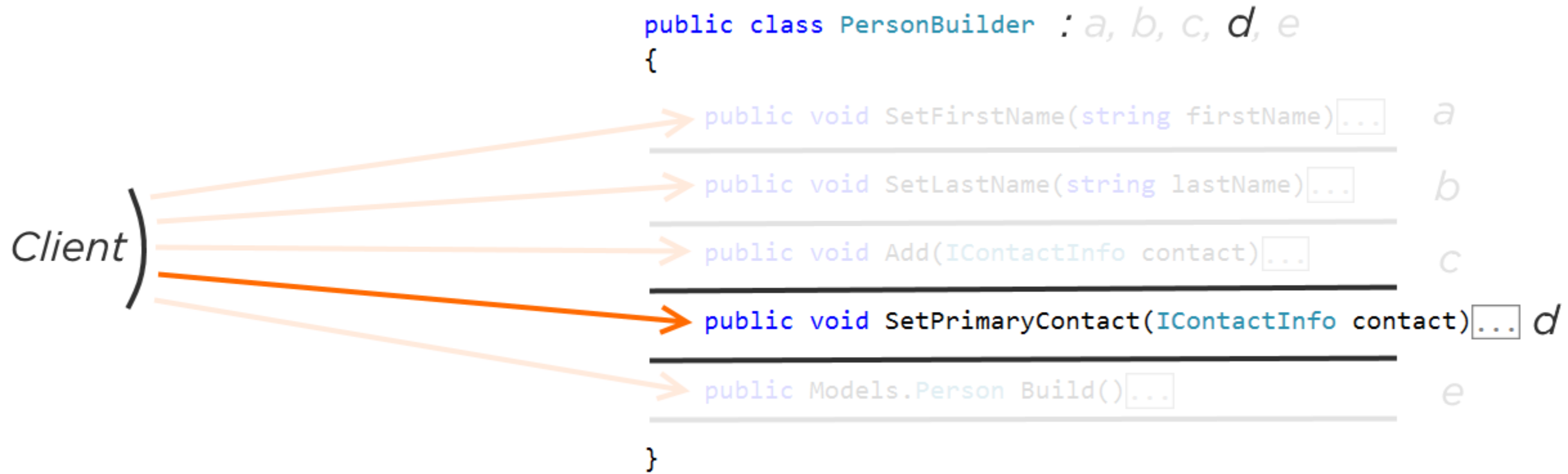
Interface Segregation Principle



Interface Segregation Principle



Interface Segregation Principle

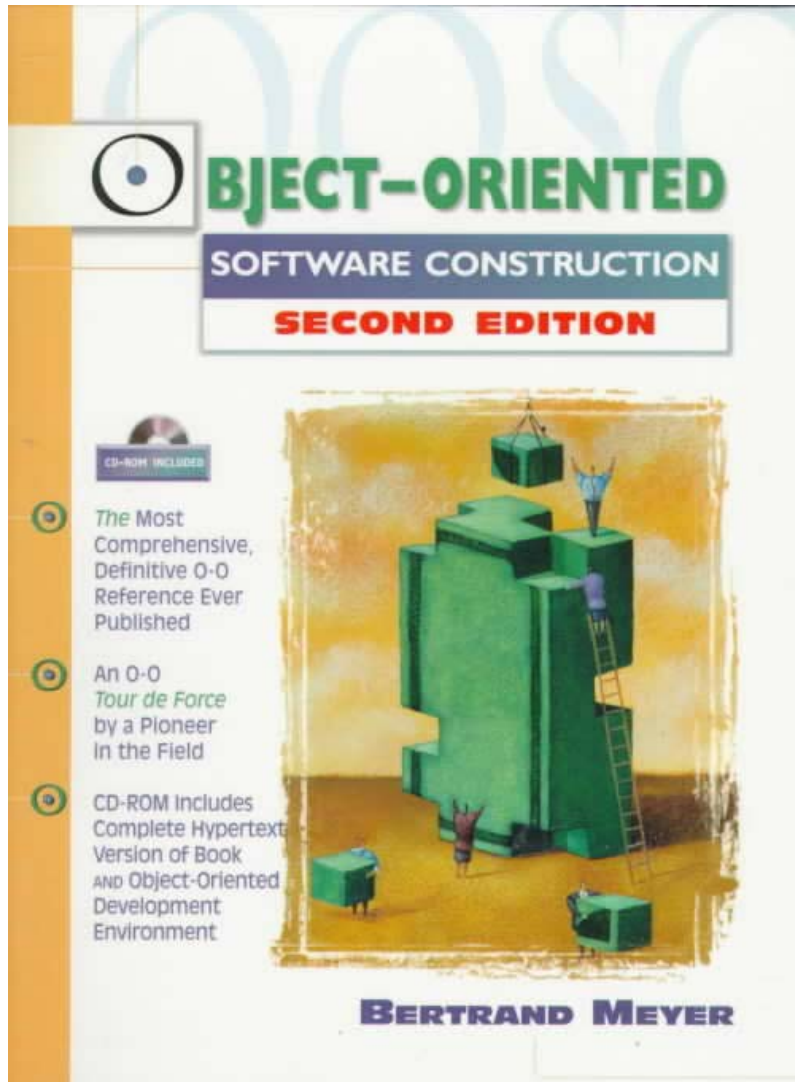


Design by Contract (DbC)

Bertrand Meyer,
Object-Oriented Software Construction

DbC invented during 1986-88

Software design patterns first mention 1987



Design by Contract in .NET Folklore

Method preconditions – if-then-throw

```
public void Add(IContactInfo contact)
{
    if (contact == null)
        throw new ArgumentException();

    if (this.Contacts.Contains(contact))
        throw new ArgumentException();

    this.Contacts.Add(contact);
}
```

```
public void SetPrimaryContact(IContactInfo contact)
{
    if (contact == null)
        throw new ArgumentNullException();

    if (!this.Contacts.Contains(contact))
        throw new ArgumentException();

    this.PrimaryContact = contact;
}
```



Bertrand Meyer on Preconditions

5 of this course:

and Liskov Substitution Principles



Related Pluralsight Courses

Provable Code

by Michael Perry

Tools and patterns for using mathematics to write more reliable and readable software

[▶ Resume Course](#)

Table of contents

Description

Transcript

Exercise files

Discussion

Learning Check

Recommended

▶ Predicate Calculus

✓


🔖

35m 27s

▼

Expand all

Course author

**Michael Perry**

Mathematician and software developer, Michael L Perry applies formal proof to creating reliable software. He has developed a method starting from the works of the greats (Meyer, Rumbaugh, Knuth), a...

Course info

Level	Intermediate
Rating	★★★★★ (80)
My rating	★★★★★
Duration	5h 51m
Released	13 Jun 2012



Related Pluralsight Courses

Code Contracts

by John Sonmez

This course provides an introduction to Code Contracts in the Microsoft .NET Framework.

▶ Resume Course

Table of contents


- Description
- Transcript
- Exercise files
- Discussion

Learning Check Recommended

Expand all

▶ Code Contracts Overview ✓ 14m 4s ▼

Course author

 **John Sonmez**

John Sonmez is the founder of Simple Programmer (<http://simpleprogrammer.com>), where he tirelessly pursues his vision of transforming complex issues into simple solutions

Course info

Level	Intermediate
Rating	★★★★★ (194)
My rating	★★★★★
Duration	1h 51m
Released	17 Jul 2012



In This Module...

**Quick demo of the
Code Contracts
library**

**Remove
data validation
from the
PersonBuilder**

**Implement
data validation
in separate classes
associated
with interfaces**



A Word of Warning

Code Contracts library has never been widely adopted

You can download Code Contracts from the Visual Studio Gallery

Keep the difficulties on mind

Lack of proper support

Slight chance of discontinuation

Code Contracts is now a community project

Could improve both on support and longevity of the project



Command-Query Separation (CQS)

to change its state

to retrieve its state

either be a command or a query



Summary



Enhanced Builder implementation

- Added State pattern to the Builder
- Better handling of mandatory operations
- Better handling of method coupling

Temporal coupling (coupling in time)

- Well handled by the State pattern

The problem of a calling protocol

- Added Interface Segregation Principle to the Builder



Summary



Where are we?

- Builder produces objects of a concrete class
- We had concrete Person and Machine
- We had abstract User as a base
- Builder cannot cope with this diversity

In the next module

- Factory Method pattern
- We will wrap multiple Builders and make them return abstract type



Next module -

*Breathing Life Back into
Factory Methods with Lambdas*

