Test-driven Development

Master Class

Day #2

What To Expect Today

- Test Doubles & Dependency Injection
- Example 5 – JG Holidays Ltd
- Putting It All Together:
  - Specification By Example & BDD
  - Test-driven Object Oriented Design
  - User Experience Design with Tests
- Example 5 – Community Video Library
- TDD & Legacy Code
- TDD Metrics
- TDD Practice Regimes

TEST DOUBLES & DEPENDENCY INJECTION

Test Doubles

Stub	Class with test-specific implementation (hard-coded responses)
Fake	A full implementation for test purposes (e.g., in-memory database)
Mock	An interface with expectations set for the test (implementation generated at runtime by mocking framework)
Dummy	A null or dummy object to be used as parameter value when test doesn’t care

Stub

```
public class TestStockPriceService implements StockPriceService {

	@Override
	public double fetchPriceForStock(String StockSymbol) {
		return 100;
	}
}
```

Mock

```
public interface StockPriceService {

        double fetchPriceForStock(String StockSymbol);
}

…

@Test
public void tradeShouldUseLatestStockPrice() {
        String stockSymbol = "BP";

        StockPriceService mockStockPriceService = createMock(StockPriceService.class);

        Trade trade = new Trade(mockStockPriceService, stockSymbol, 50);

        expect(mockStockPriceService.fetchPriceForStock(stockSymbol)).andReturn(0.01);

        replay(mockStockPriceService);

        assertEquals(0.5, trade.calculateCurrentTradePrice(), 0);

        verify(mockStockPriceService);
}
```

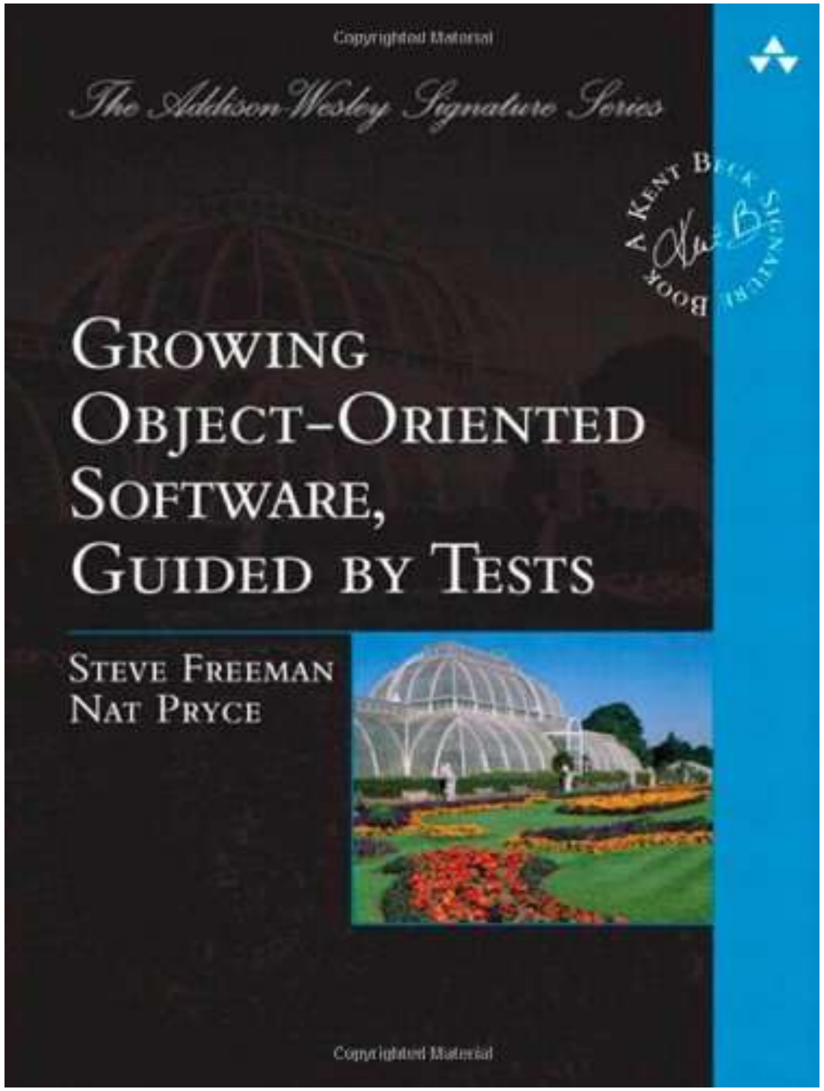
Example #5 – JG Holidays Ltd

Write some code that will tell us what villas are available on weeks where flights are also available

Use **stubs** to act as facades to external systems.

1. Villa booking system – tell it the week and year (e.g. week 26, 2011) and it will return an array of villa names that are available
2. Flight booking system – tell it the week and the year and the start/destination airports (e.g., London Heathrow (LHR) -> Paphos International (PFO)) and it will tell you if flights are available in that week

TEST-DRIVEN OBJECT ORIENTED DESIGN


Copyrighted Material
The Addison-Wesley Signature Series
A Kent Beck Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce
Copyrighted Material

Where do we start?

USERS & THEIR GOALS

Donate a DVD

As a video library **member**, I want to **donate a DVD to the library** so that other members can borrow it and I can earn points for priority services

codemanship

What’s the outcome?

TEST-DRIVEN

Given a copy of a DVD title that *isn't in the library,*

When a member donates their copy, specifying the name of the DVD title

Then that title is added to the library *and* their copy is registered against that title so that other members can borrow it,
AND an email alert is sent to members who specified an interest in matching titles,
AND the new title is added to the list of new titles for the next member newsletter
AND the member is awarded priority points

Examples: Be Specific…

<u>When</u> Joe Peters donates his copy, specifying the name of the title, that it was directed by James Cameron and released in 1989

<u>Then</u> The Abyss is added to the library and his copy is registered against that title so that other members can borrow it,
AND an email alert with the subject “New DVD title” is sent to Bill Smith and Jane Jones, who specified an interest in titles matching “the abyss”(non-case-sensitive), stating “Dear <member’s first name>, Just to let you know that another member has recently donated a copy of The Abyss (dir: James Cameron, 1989) to the library, and it is now available to borrow.”
AND The Abyss is added to the list of new titles for the next member newsletter
AND Joe Peters receives 10 priority points for making a donation

Examples: Be Specific…

<u>When</u> Joe Peters donates his copy, specifying the name of the title, that it was directed by James Cameron and released in 1989

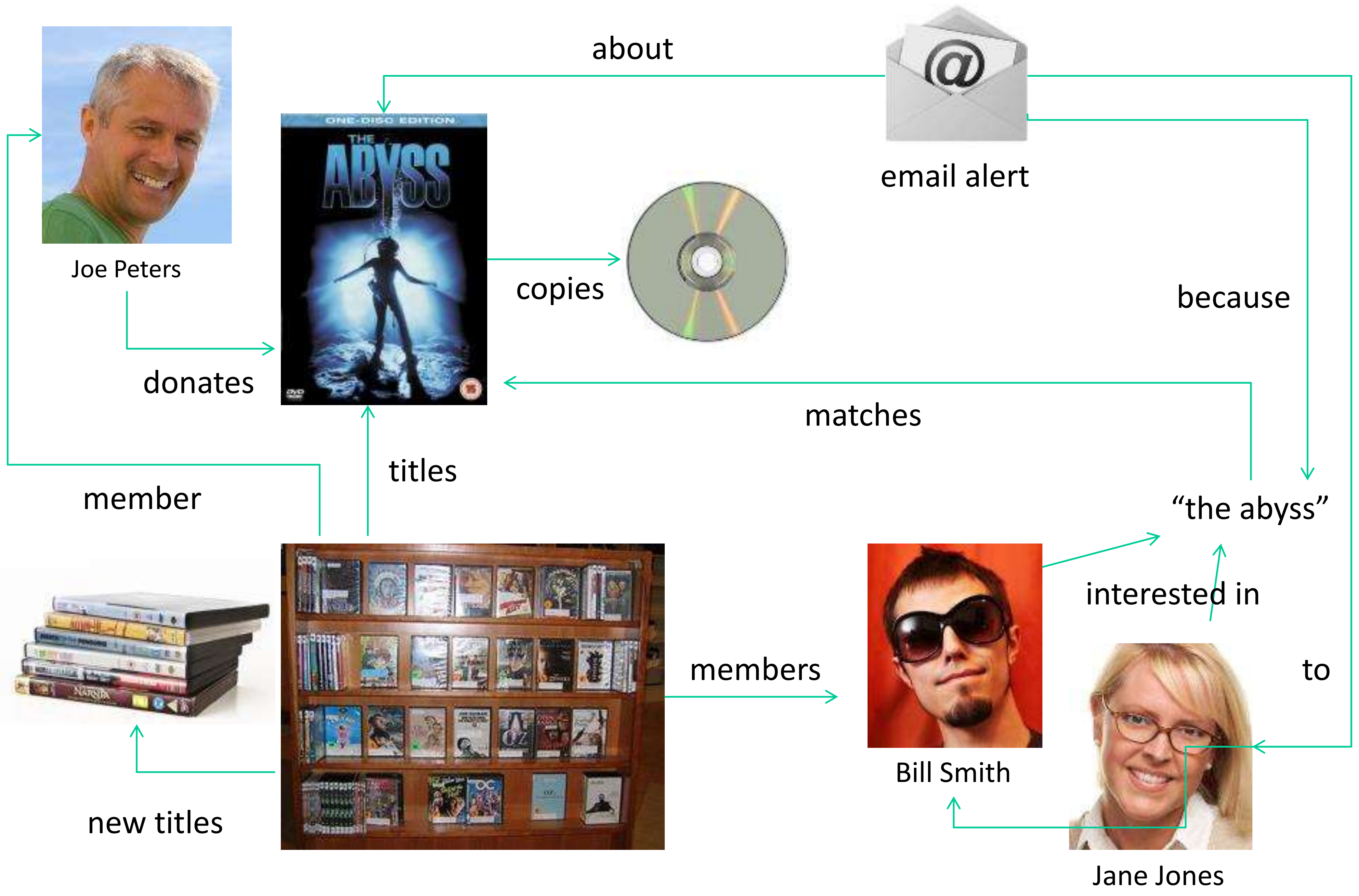
<u>Then</u> The Abyss is added to the library and his copy is registered against that title so that other members can borrow it,
AND an email alert with the subject “New DVD title” is sent to Bill Smith and Jane Jones, who specified an interest in titles matching “the abyss”(non-case-sensitive), stating “Dear <member’s first name>, Just to let you know that another member has recently donated a copy of The Abyss (dir: James Cameron, 1989) to the library, and it is now available to borrow.”
AND The Abyss is added to the list of new titles for the next member newsletter
AND Joe Peters receives 10 priority points for making a donation

codemanship

Who are the characters, and how do they know each other?

OBJECT ORIENTED

codemanship



codemanship

What's the story?

DESIGN

codemanship

Responsibilities: Passive Story

1. “The Abyss” is added to the library with title name “The Abyss”, director “James Cameron” and year 1989
2. One rental copy is registered to “The Abyss”
3. An email is sent to any members who specified that they wanted to be alerted when a title matching “the abyss” (non-case sensitive) was donated
4. “The Abyss” is added to the list of new titles
5. Joe Peters is awarded 10 priority points

Roles: Active Story

1. The **library** adds "The Abyss" to itself with title name "The Abyss", director "James Cameron" and year 1989
2. The new **title** "The Abyss" registers one rental copy to itself
3. An **email alert** sends itself to any members who specified that they wanted to be alerted when a title matching "the abyss" (non-case sensitive) was donated
4. The **library** adds "The Abyss" to the list of new titles
5. **Member** Joe Peters awards himself 10 priority points

hidden complexity

codemanship

Roles: Active Story

1. The **library** adds "The Abyss" to itself with title name "The Abyss", director "James Cameron" and year 1989
2. The new **title** "The Abyss" registers one rental copy to itself
3. An **email alert** sends itself to any members who specified that they wanted to be alerted when a title matching "the abyss" (non-case sensitive) was donated
4. The **library** adds "The Abyss" to the list of new titles
5. **Member** Joe Peters awards himself 10 priority points

hidden complexity

codemanship

Collaborations: Characters Interact

1. The **library** adds “The Abyss” to itself with title name “The Abyss”, director “James Cameron” and year 1989, then tells
2. the new **title** “The Abyss” to register one rental copy to itself, who tells
3. an **email alert** to send itself to any members who specified that they wanted to be alerted when a title matching “the abyss” (non-case sensitive) was donated
4. The **library** adds “The Abyss” to the list of new titles, then tells
5. **member** Joe Peters to award himself 10 priority points

Class-Responsibility-Collaboration (CRC) Cards

Library

• Knows about titles • Knows about new titles • Adds donated titles • Adds new titles	• Title • Member

Title

• Knows its name, director & year of release • Knows about rental copies • Registers rental copy	• Email Alert

Email Alert

• Sends email to members who specified matching title	?

Member

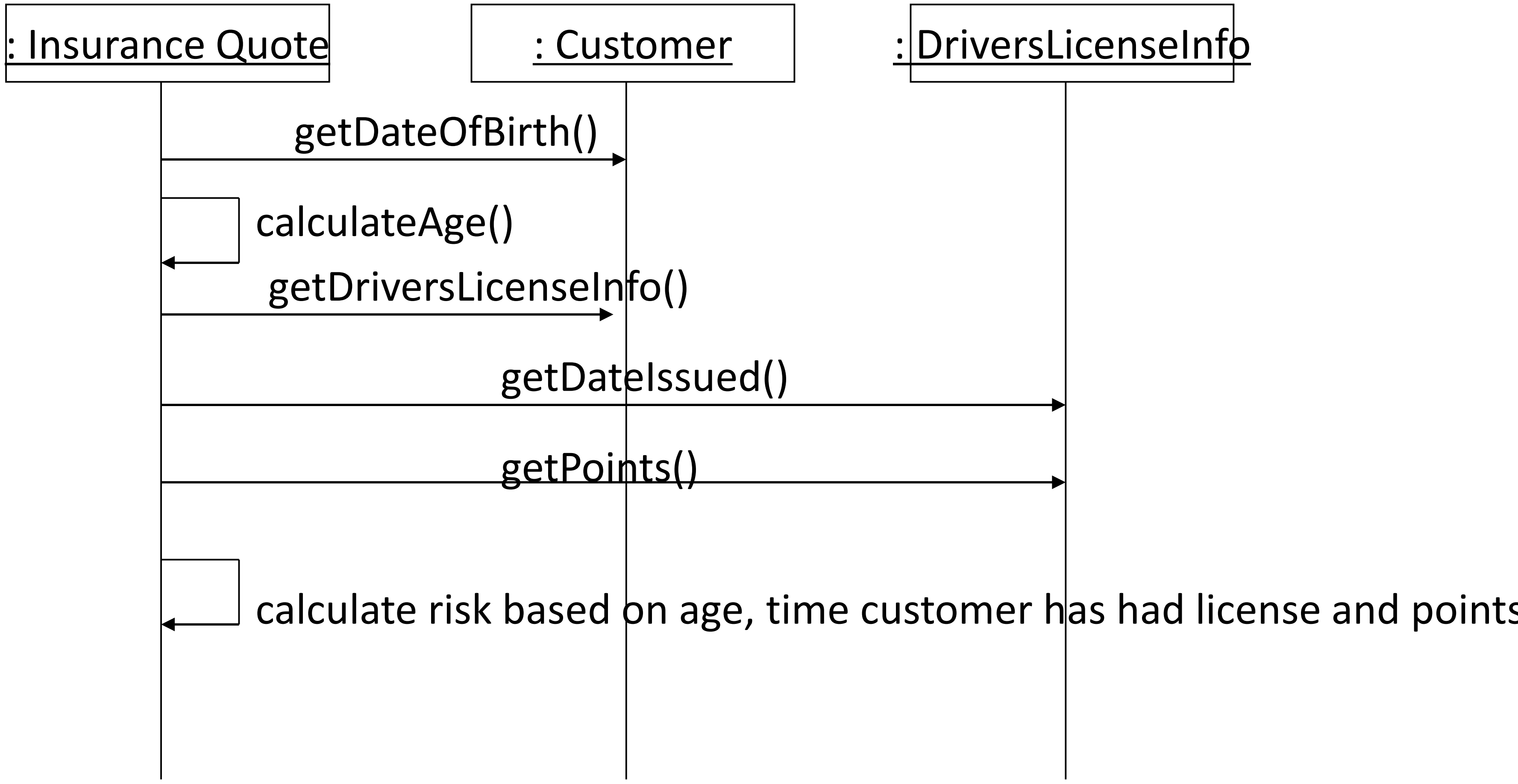
• Knows about priority points • Awards priority points	

Put the behaviour where the data is

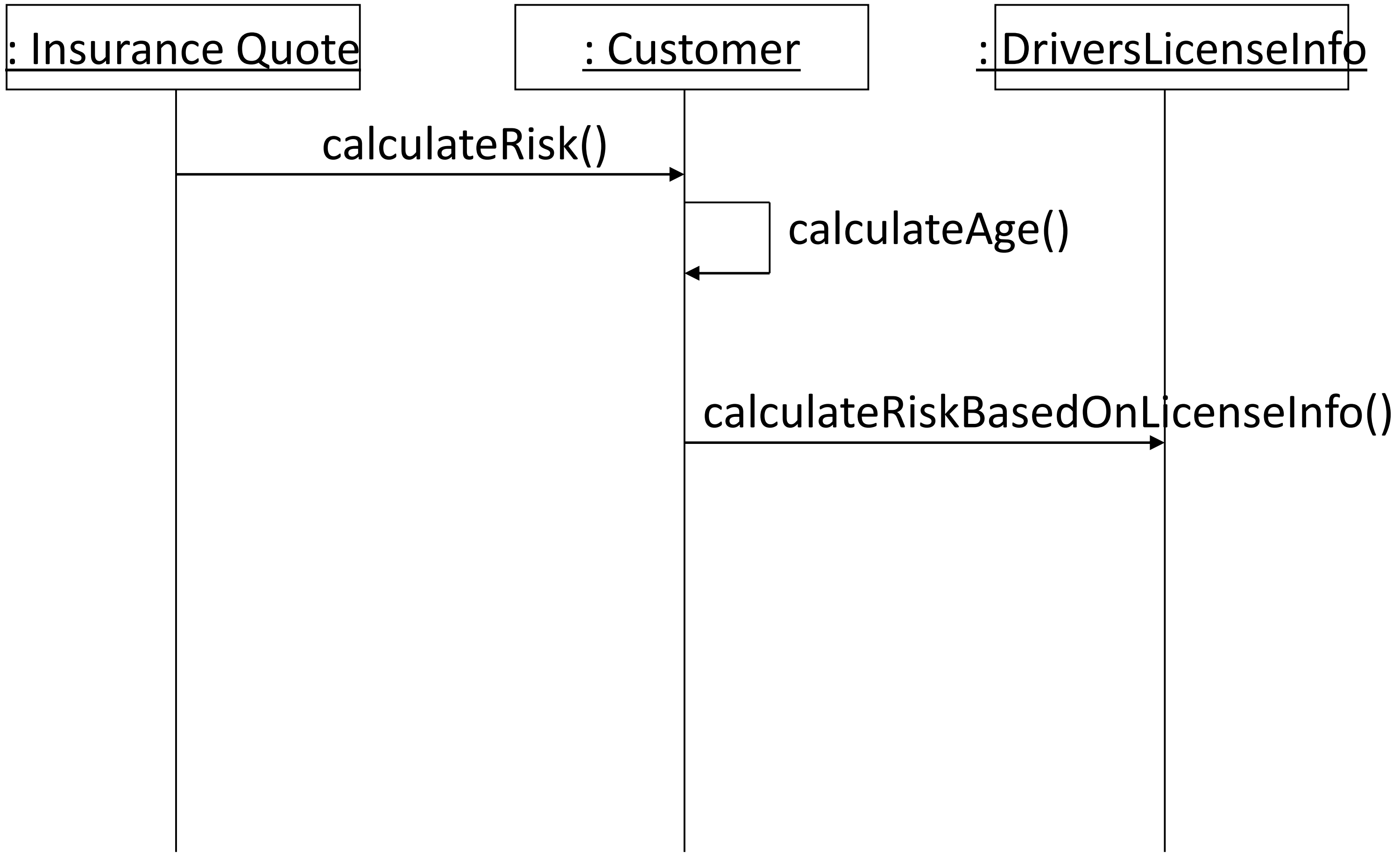
ENCAPSULATION

codemanship

Data-driven Design

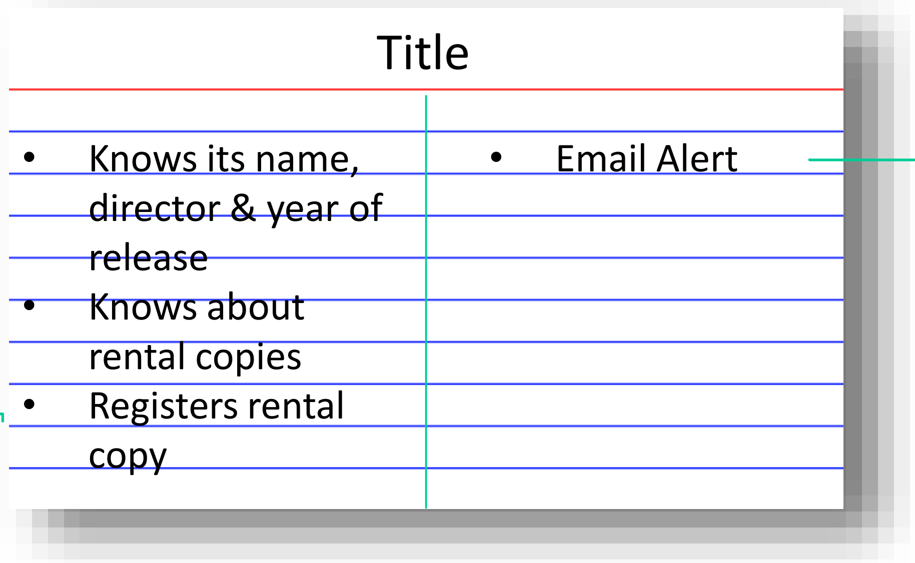


Tell, Don't Ask



Classic TDD + London School

TESTS!



Title

- Knows its name, director & year of release
- Knows about rental copies
- Registers rental copy

- Email Alert

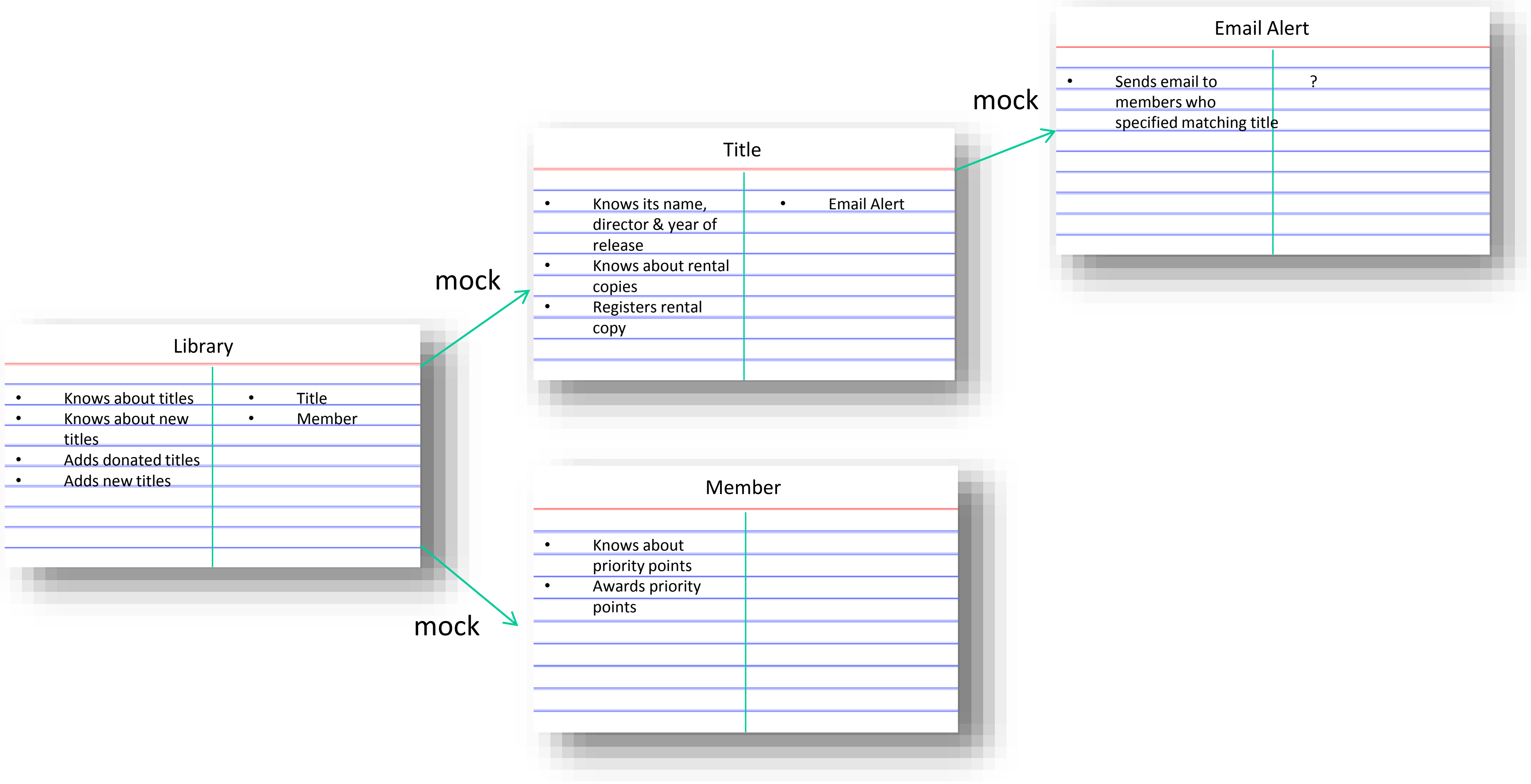
```
@Test
public void registersRentalCopy() {
     EmailAlert emailAlert = mock(EmailAlert.class);
     Title title = new Title(null, null, null, emailAlert);
     title.registerCopy();
     assertEquals(1, title.getRentalCopyCount());
}
```

```
@Test
public void tellsEmailAlertToSend() {
     EmailAlert emailAlert = mock(EmailAlert.class);
     Title title = new Title(null, null, null, emailAlert);
     title.registerCopy();
     verify(emailAlert).send(title);
}
```

codemanship

Gluing It All Together From The

OUTSIDE-IN



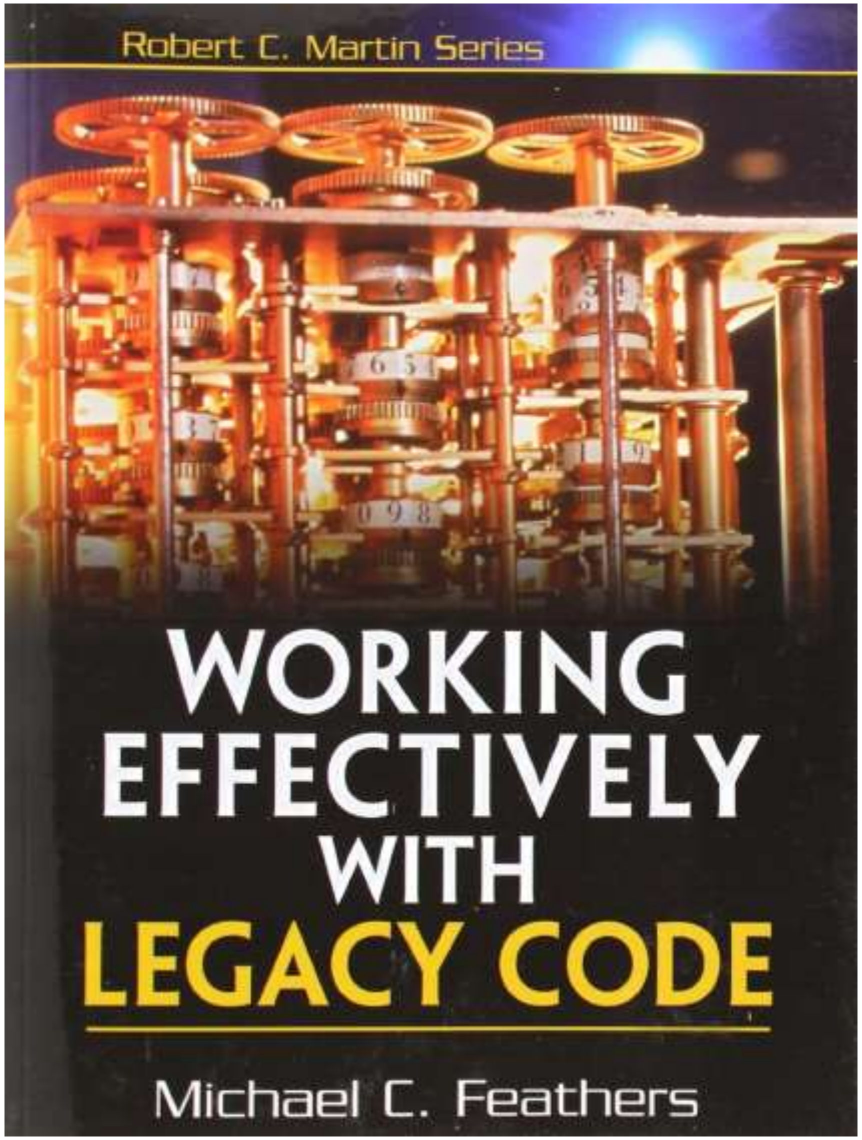
codemanship

And so to business

VIDEO LIBRARY USER STORIES

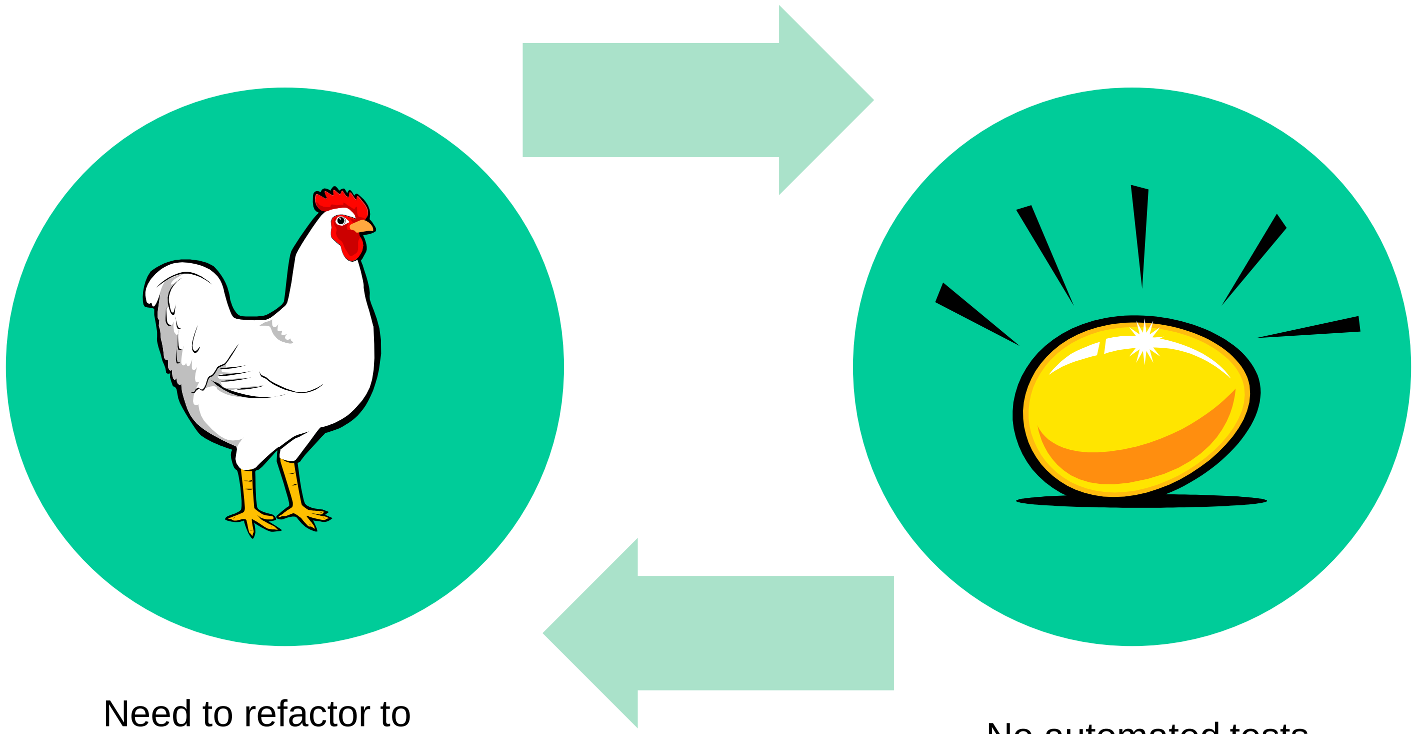
www.codemanship.com/files/tddmasterclass.zip

TDD & LEGACY CODE



codemanship

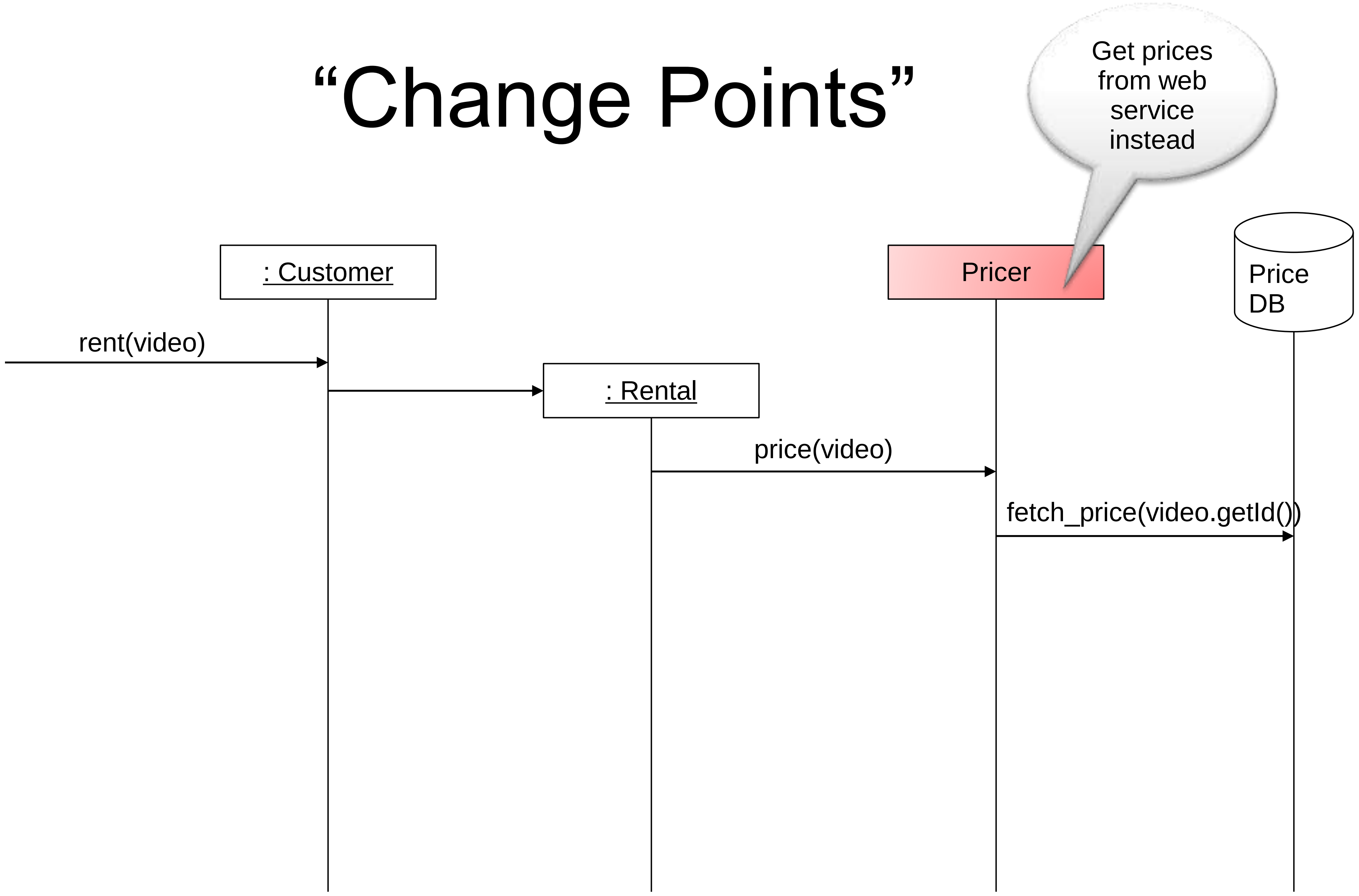
Catch 22



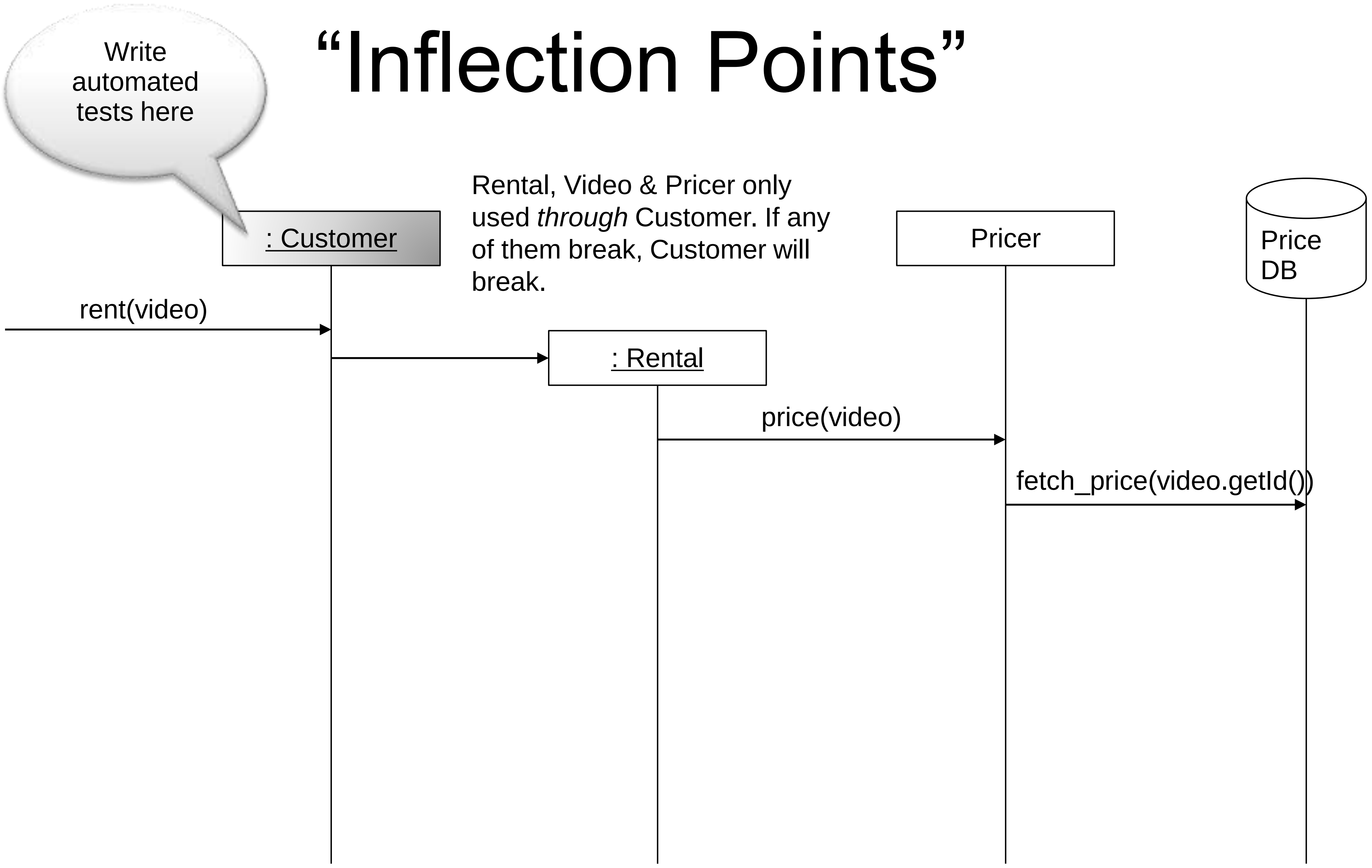
Need to refactor to make code unit testable

No automated tests to support refactoring

"Change Points"


Get prices from web service instead
: Customer
Pricer
Price DB
rent(video)
: Rental
price(video)
fetch_price(video.getId())

“Inflection Points”


Write automated tests here
: Customer
Rental, Video & Pricer only used *through* Customer. If any of them break, Customer will break.
Pricer
Price DB
rent(video)
: Rental
price(video)
fetch_price(video.getId())

Initial Testing Options

- Automated system/integration tests
- Test manually
- ~~Don’t test~~

BREAKING DEPENDENCIES

```
public class CustomerServices
    {
        public string FetchCustomerXmlById(int customerId)
        {
            Customer customer = DataRepository.GetCustomer(customerId);
            string xml = "<Error>Customer not found</Error>";
            if(customer != null)
            {
                xml = "<Customer id='" + customerId + "'>" + customer.Name + "</Customer>";
            }
            return xml;
        }

        public string FindCustomersXmlByName(string nameToMatch)
        {
            IList<Customer> matches = DataRepository.FindCustomers(nameToMatch);
            string xml = "<Matches count='" + matches.Count + "'>";
            foreach(Customer customer in matches){
                xml += "<Customer id='" + customer.Id + "'>" + customer.Name + "</Customer>";
            }
            xml += "</Matches>";
            return xml;

        }

    }
```

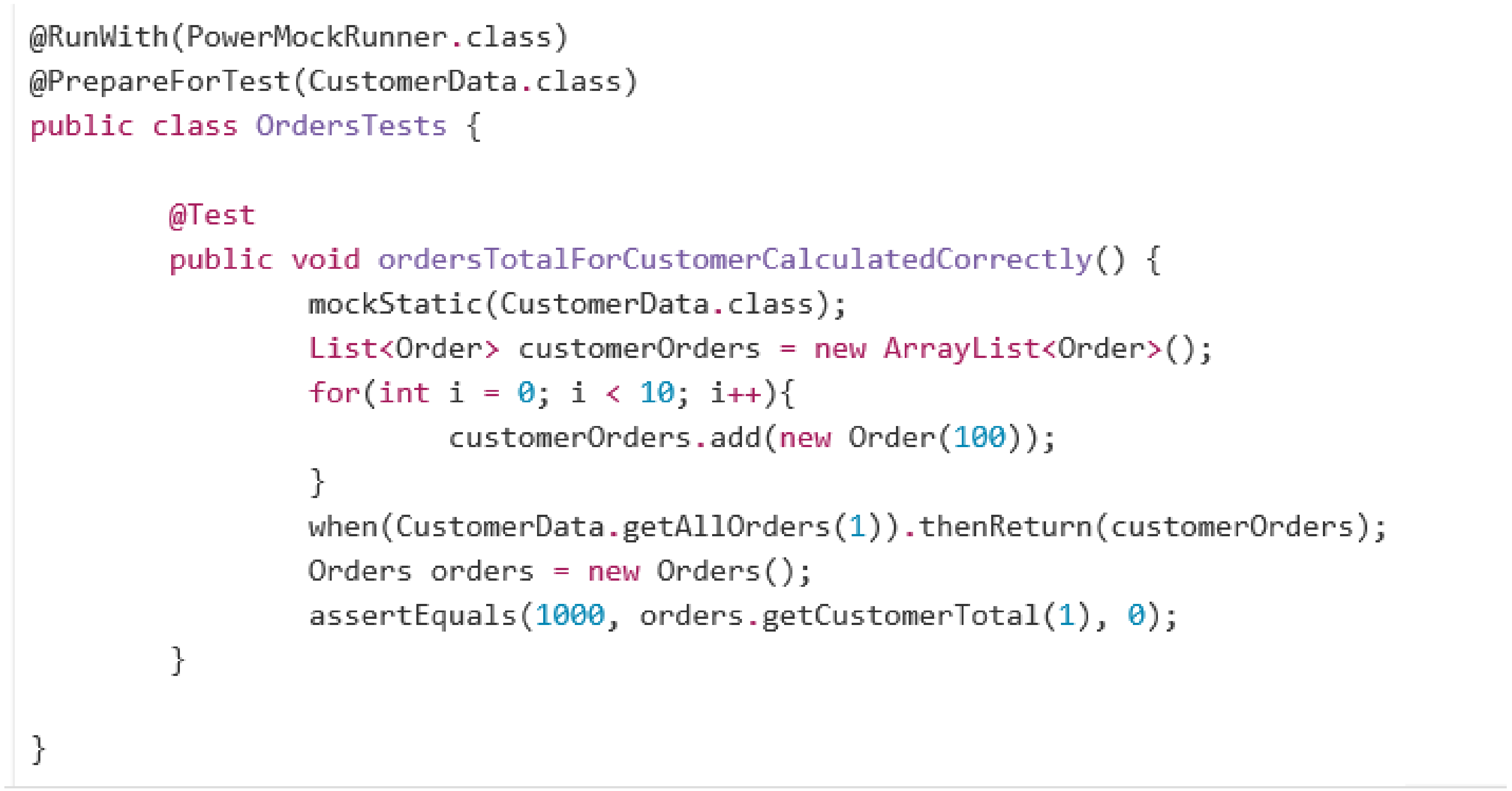
```
public class CustomerServices
    {
        public string FetchCustomerXmlById(int customerId)
        {
            Customer customer = new DataRepository().GetCustomer(customerId);
            string xml = "<Error>Customer not found</Error>";
            if(customer != null)
            {
                xml = "<Customer id='" + customerId + "'>" + customer.Name + "</Customer>";
            }
            return xml;
        }

        public string FindCustomersXmlByName(string nameToMatch)
        {
            IList<Customer> matches = new DataRepository().FindCustomers(nameToMatch);
            string xml = "<Matches count='" + matches.Count + "'>";
            foreach(Customer customer in matches){
                xml += "<Customer id='" + customer.Id + "'>" + customer.Name + "</Customer>";
            }
            xml += "</Matches>";
            return xml;

        }
}
```

```csharp
public class CustomerServices
    {
        private readonly IDataRepository dataRepository;

        public CustomerServices(IDataRepository repository)
        {
            dataRepository = repository;
        }

        public string FetchCustomerXmlById(int customerId)
        {
            Customer customer = dataRepository.GetCustomer(customerId);
            string xml = "<Error>Customer not found</Error>";
            if(customer != null)
            {
                xml = "<Customer id='" + customerId + "'>" + customer.Name + "</Customer>";
            }
            return xml;
        }

        public string FindCustomersXmlByName(string nameToMatch)
        {
            IList<Customer> matches = dataRepository.FindCustomers(nameToMatch);
            string xml = "<Matches count='" + matches.Count + "'>";
            foreach(Customer customer in matches){
                xml += "<Customer id='" + customer.Id + "'>" + customer.Name + "</Customer>";
            }
            xml += "</Matches>";
            return xml;

        }

    }
```

```csharp
public interface IDataRepository
{
    Customer GetCustomer(int customerId);
    IList<Customer> FindCustomers(string nameToMatch);
}
```

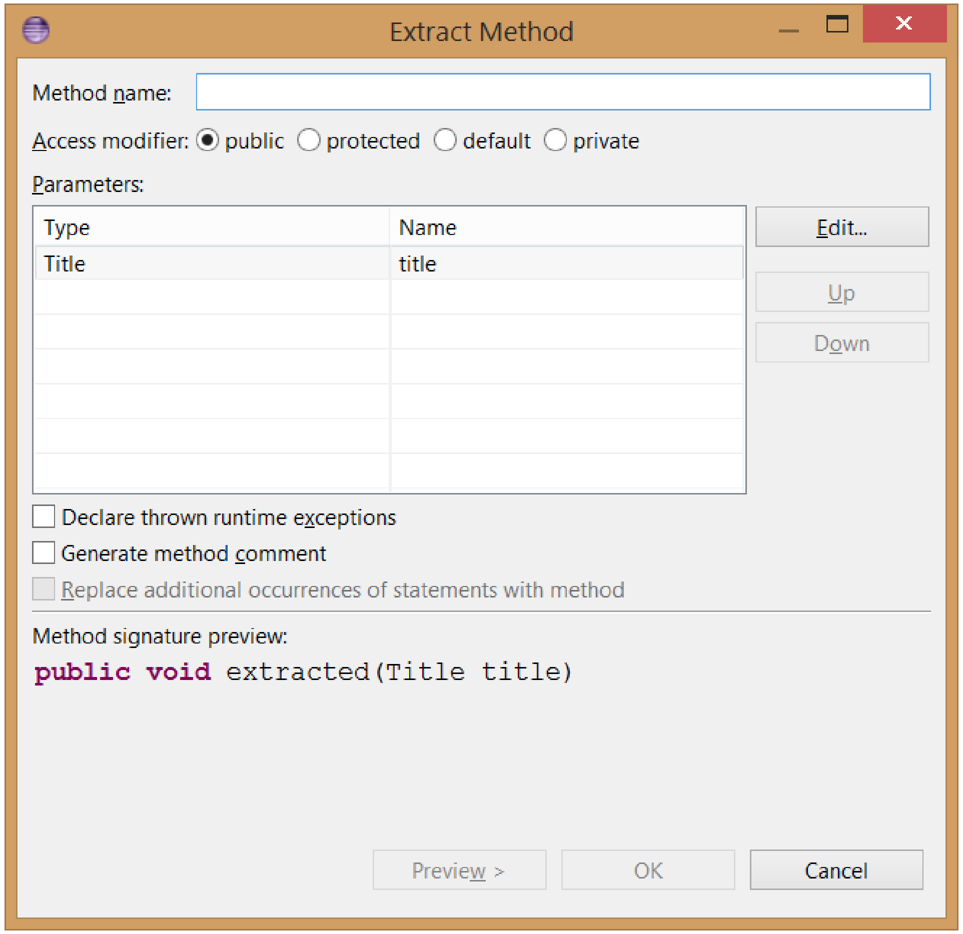
LEGACY CODE GOTCHAS

Baking In A Bad Design



```
@RunWith(PowerMockRunner.class)
@PrepareForTest(CustomerData.class)
public class OrdersTests {

        @Test
        public void ordersTotalForCustomerCalculatedCorrectly() {
                mockStatic(CustomerData.class);
                List<Order> customerOrders = new ArrayList<Order>();
                for(int i = 0; i < 10; i++){
                        customerOrders.add(new Order(100));
                }
                when(CustomerData.getAllOrders(1)).thenReturn(customerOrders);
                Orders orders = new Orders();
                assertEquals(1000, orders.getCustomerTotal(1), 0);
        }

}
```

Automated Refactoring Tools Negate Need For Testing



Method name:

Access modifier: [✓ public] protected default private

Parameters:

Type	Name
Title	title

☐ Declare thrown runtime exceptions

☐ Generate method comment

☐ Replace additional occurrences of statements with method

Method signature preview:

```
public void extracted(Title title)
```

codemanship

How Do We Know If We're Doing It Right?

TDD METRICS

Getting Into The Habit…

TDD PRACTICE REGIMES