Configuring Mocked Methods

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

Instantiating and using a mock object

Refactor:

- AcceptHighIncomeApplications
- ReferYoungApplications

Configure mock object method return values

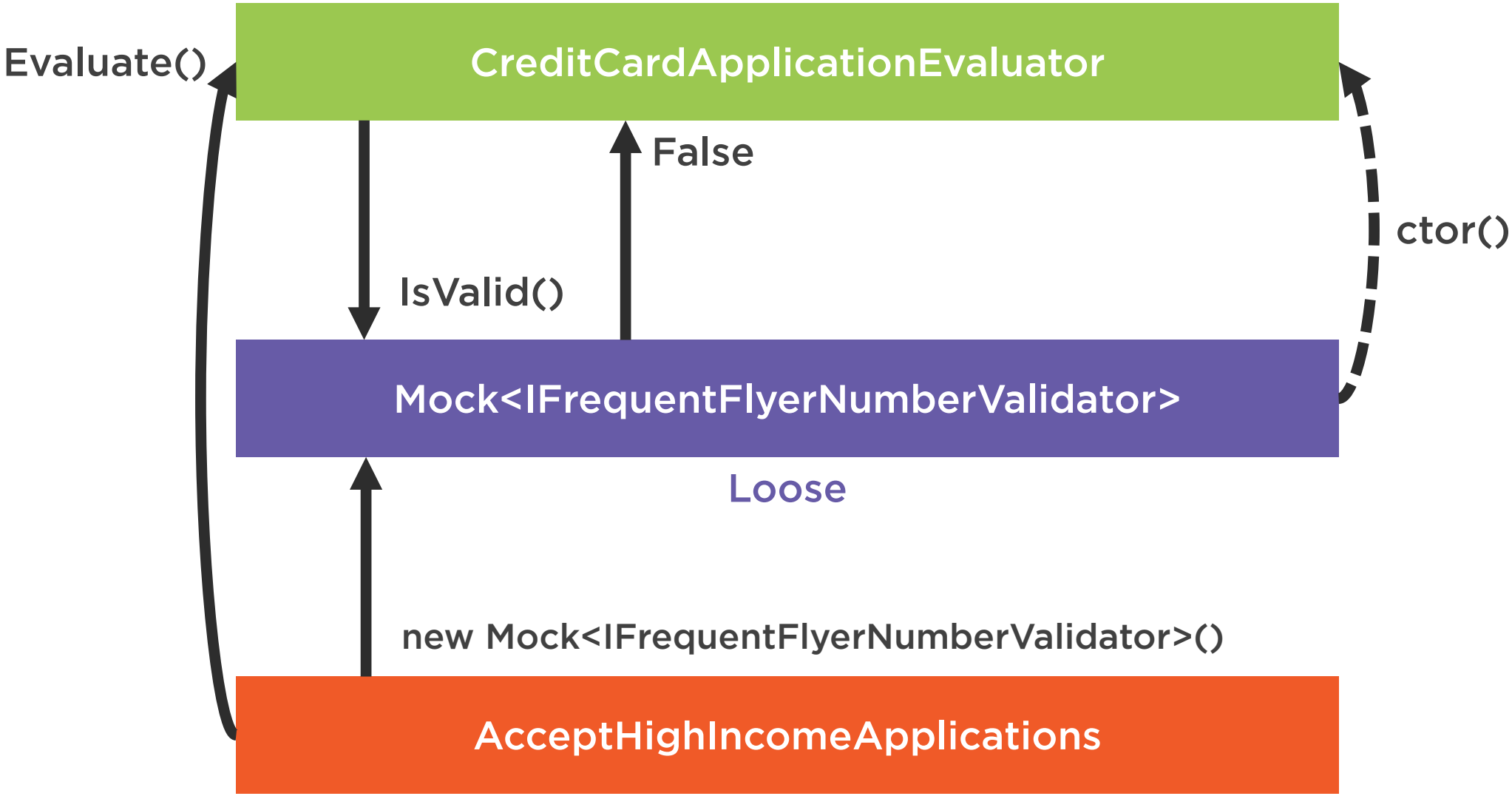
Argument matching in mocked methods

- Any values
- Predicates
- Ranges
- Regular expressions

Understanding strict and loose mocks

Methods with out/ref parameters

Understanding Strict and Loose Mocks



`MockBehavior.Strict`

- Throw an exception if a mocked method is called but has not been setup.

`MockBehavior.Loose`

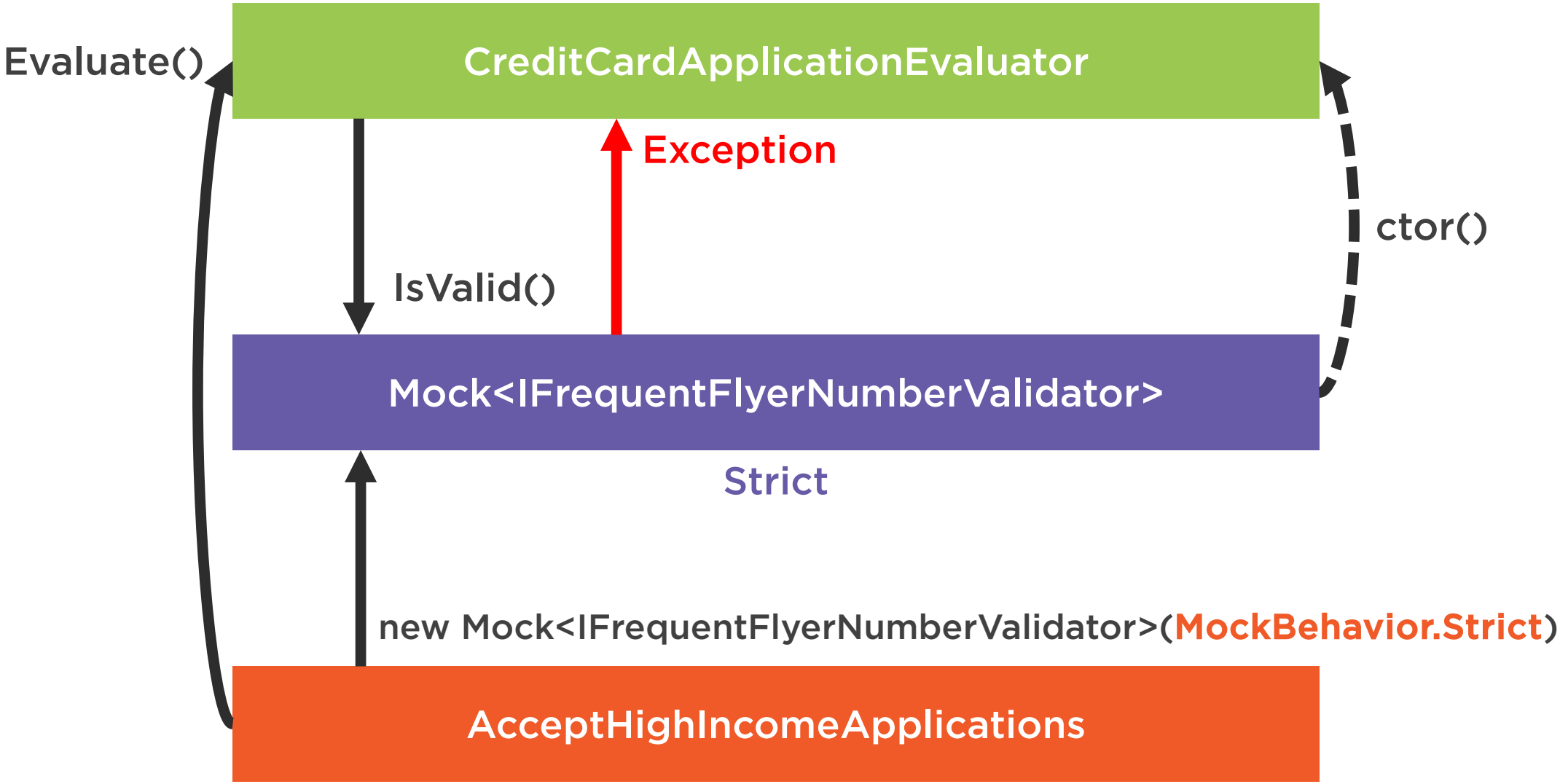
- Never throw exceptions, even if a mocked method is called but has not been setup.

  Returns default values for value types, null for reference types, empty array/enumerable.

`MockBehavior.Default`

- Default behavior if none specified (MockBehavior.Loose)

Creating a Strict Mock



Comparing Strict and Loose Mocks

Loose	Strict
Fewer lines of setup code	More lines of setup code
Setup only what's relevant	May have to setup irrelevant things
Default values	Have to setup each called method
Less brittle tests	More brittle tests
Existing tests continue to work	Existing tests may break

Use strict mocks only when absolutely necessary, prefer loose mocks at all other times.

Matching ref Arguments with Moq

```
public interface IGateway
{
    int Execute(ref Person person);
}
```

```
public class Processor
{
    private readonly IGateway _gateway;

    public Processor(IGateway gateway)
    {
        _gateway = gateway;
    }

    public bool Process(Person person)
    {

        int returnCode = _gateway.Execute(ref person);

        return returnCode == 0;
    }
}
```

No Explicit ref Setup

```
var person = new Person();
var mockGateway = new Mock<IGateway>();
var sut = new Processor(mockGateway.Object);
sut.Process(person); // IGateway.Execute() returns 0
```

Match Specific ref Object Instance

```
var person1 = new Person();
var person2 = new Person();
var mockGateway = new Mock<IGateway>();
mockGateway.Setup(x => x.Execute(ref person1)).Returns(-1);
var sut = new Processor(mockGateway.Object);
sut.Process(person1); // IGateway.Execute() returns -1
sut.Process(person2); // IGateway.Execute() returns 0
```

Match Any Assignment Compatible Type

```
var person1 = new Person();
var person2 = new Person();

var mockGateway = new Mock<IGateway>();

mockGateway.Setup(x => x.Execute(ref It.Ref<Person>.IsAny))
           .Returns(-1);

var sut = new Processor(mockGateway.Object);

sut.Process(person1); // IGateway.Execute() returns -1

sut.Process(person2); // IGateway.Execute() returns -1

public class Boss : Person { }

sut.Process(new Boss()); // IGateway.Execute() returns -1
```

Summary

Mock<IFrequentFlyerNumberValidator>()

Fixed existing tests

mockValidator.Object

Configured mock object method return values

mockValidator.Setup(...).Returns(true)

Specific value: x => x.IsValid("x")

Argument matching in mock methods, e.g.

- It.IsAny
- It.IsInRange

MockBehavior.Strict

Mocking out and ref parameters

Up Next:
Configuring Mock Object Properties