Mocking Method Calls

Jason Roberts
.NET MVP

@robertsjason dontcodetired.com

Overview

Instantiating and using a mock object

Refactor:

- AcceptHighIncomeApplications
- ReferYoungApplications

Configure mock object method return values

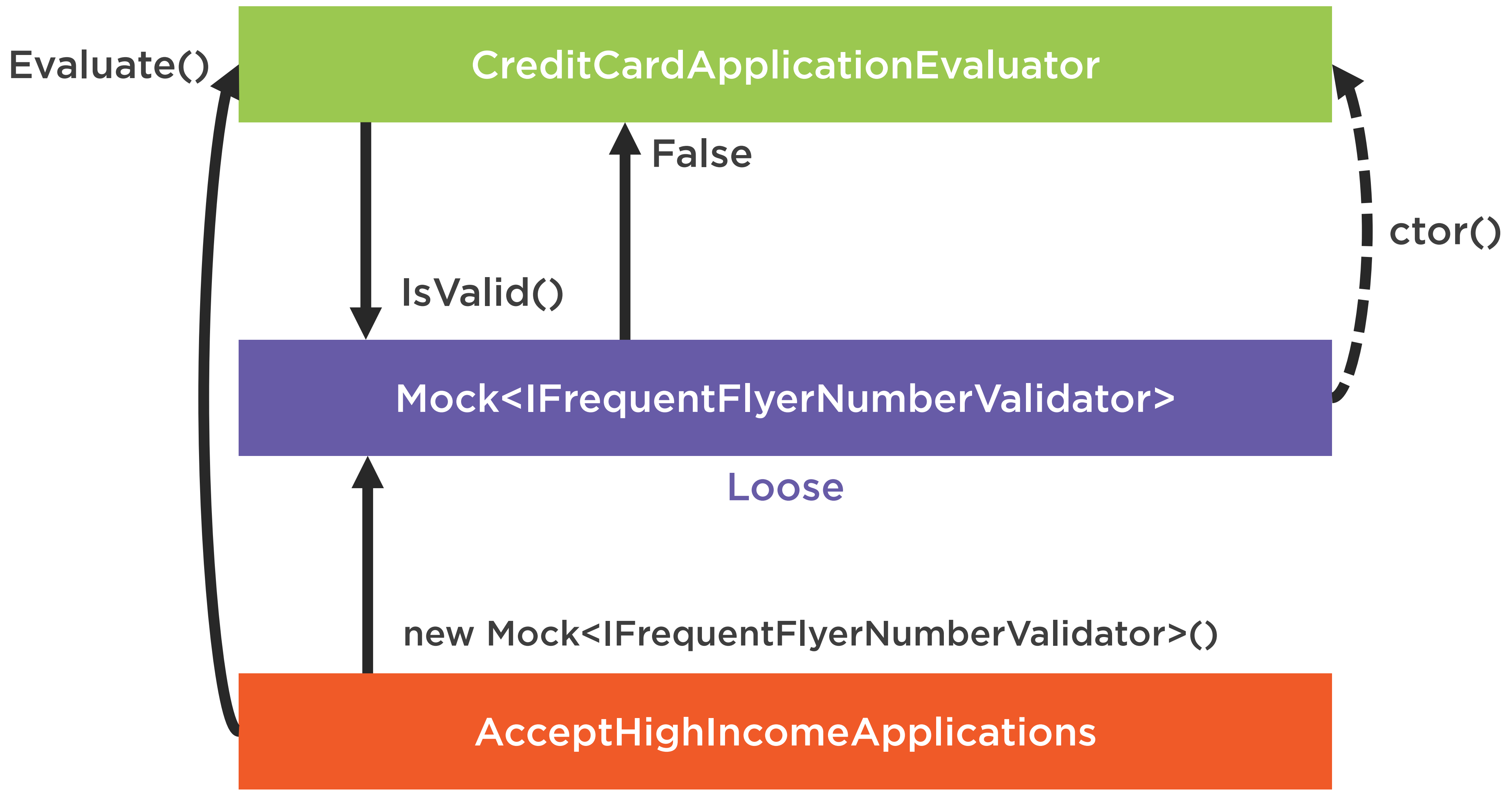
Argument matching in mocked methods

- Any values
- Predicates
- Ranges
- Regular expressions

Understanding strict and loose mocks

Mocking methods with out parameters

Understanding Strict and Loose Mocks


Evaluate()
CreditCardApplicationEvaluator
False
ctor()
IsValid()
Mock<IFrequentFlyerNumberValidator>
Loose
new Mock<IFrequentFlyerNumberValidator>()
AcceptHighIncomeApplications

`MockBehavior.Strict`

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

`MockBehavior.Loose`

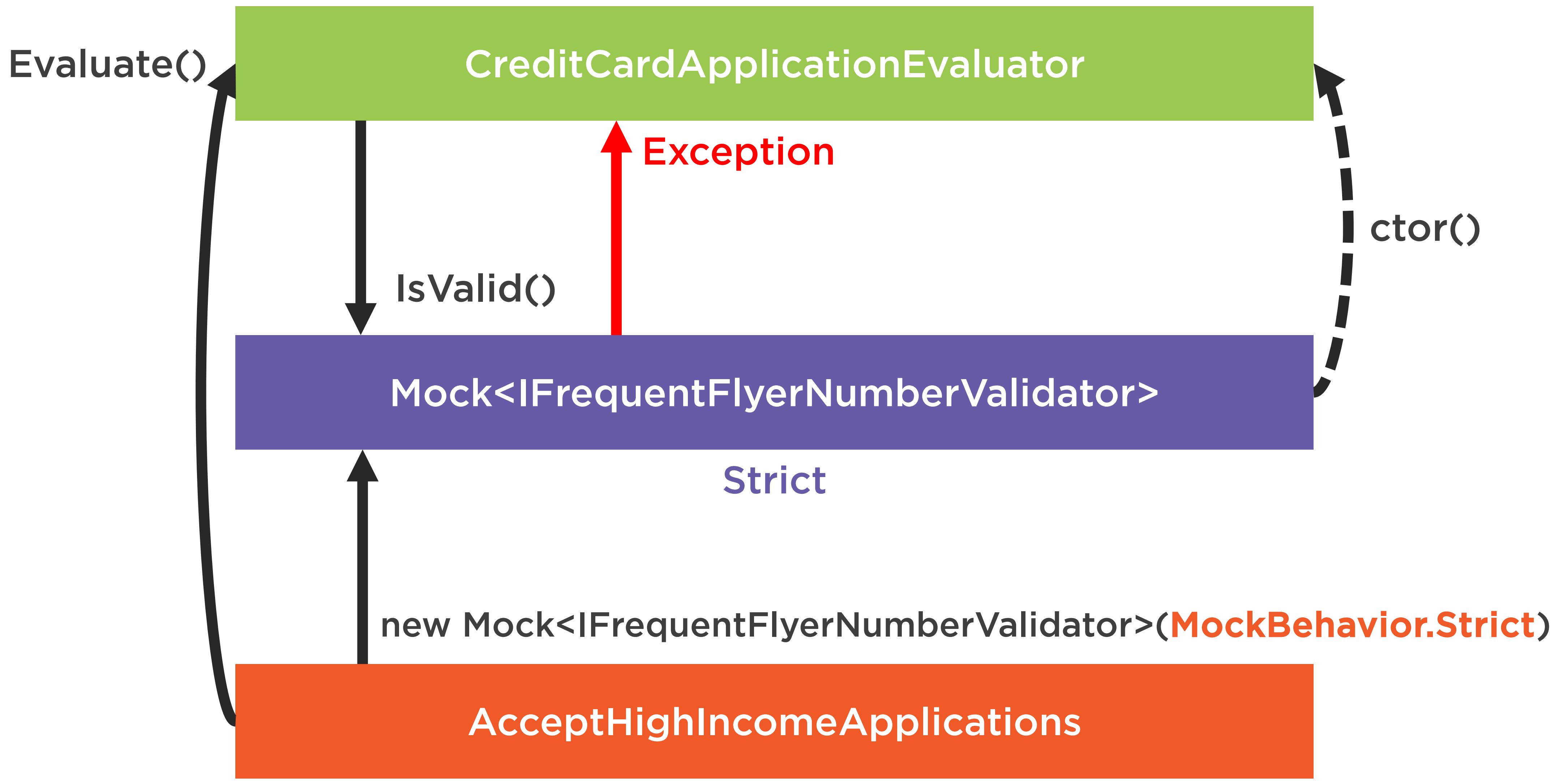
◂ Never throw exceptions, even if a mocked method is called but has not been setup.

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

`MockBehavior.Default`

◂ Default behavior if none specified (MockBehavior.Loose)

Understanding Strict and Loose Mocks


Evaluate()
CreditCardApplicationEvaluator
Exception
ctor()
IsValid()
Mock<IFrequentFlyerNumberValidator>
Strict
new Mock<IFrequentFlyerNumberValidator>(MockBehavior.Strict)
AcceptHighIncomeApplications

Comparing Strict and Loose Mocks

Loose	Strict
Less lines of setup code	More setup code
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.

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 mocked methods

- It.IsAny
- It.IsInRange

MockBehavior.Strict

Mocking methods with out parameters

Next:

Mocking Properties