Defining Function Domains as the Primary Line of Defense

Zoran Horvat

PRINCIPAL CONSULTANT AT CODING HELMET

@zoranh75 csharpmentor.com

Partial vs. Total Functions

```
public void Deposit(decimal amount)
{
}
```

Partial Function

Function which is defined on a subset of all possible argument values.

Total Function

Function which is defined for all possible argument values.

Partial vs. Total Functions

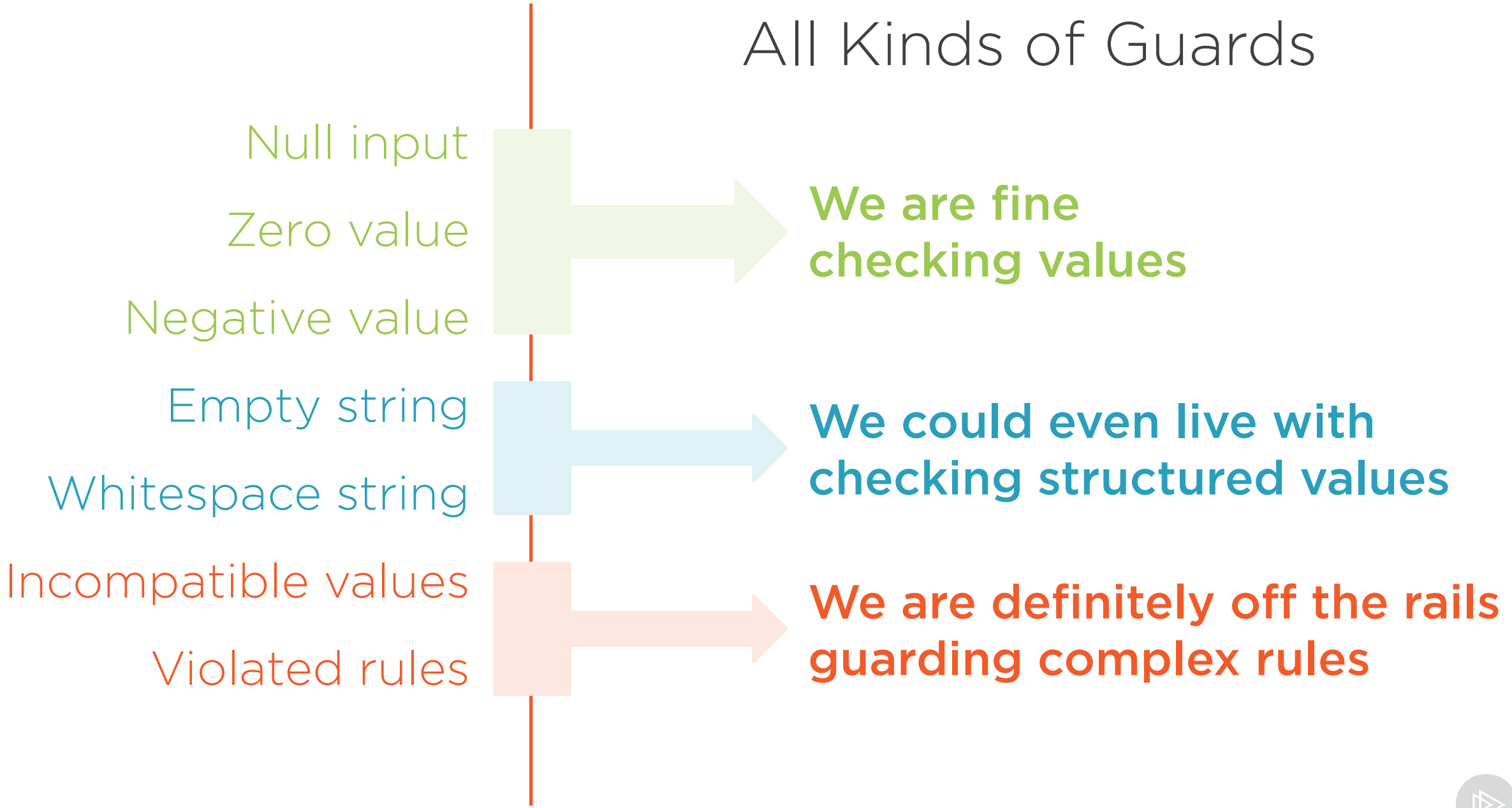
```
public void Deposit(decimal amount)
{
  if (amount <= 0)
    throw new ArgumentException(
        “Only positive amounts can be deposited.”,
        nameof(amount));
  // Keep going
}
```

Guard Clause

Programmatic implementation of the partial function.

(a.k.a. if-throw pattern)

All Kinds of Guards



```
class ExamApplicationBuilder
{
  ...
  public bool CanBuild() =>
    this.Administrator != null &&
    this.Subject != null &&
    this.Candidate != null &&
    this.Candidate.Enrolled ==
        this.Subject.TaughtDuring &&
    !this.Candidate
      .HasPassedExam(this.Subject) &&
    this.Subject.TaughtBy ==
        this.Administrator;
  ...
}
```

- **Guarding against null is fine**
  That defines the domain on which function is defined

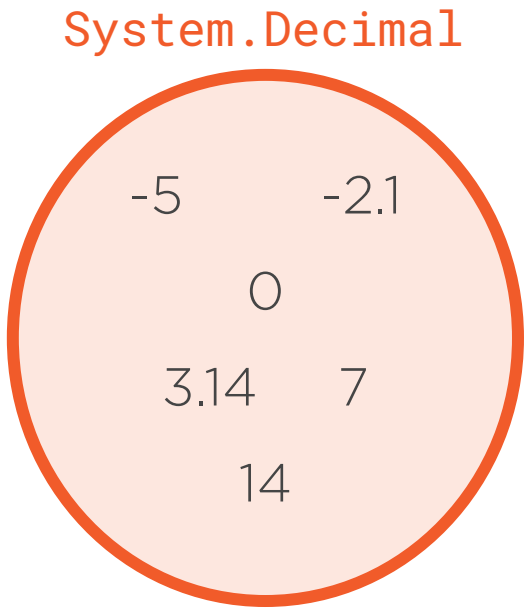
```
class ExamApplicationBuilder
{
  ...
  public bool CanBuild() =>
    this.Administrator != null &&
    this.Subject != null &&
    this.Candidate != null &&
    this.Candidate.Enrolled ==
        this.Subject.TaughtDuring &&
    !this.Candidate
      .HasPassedExam(this.Subject) &&
    this.Subject.TaughtBy ==
        this.Administrator;
  ...
}
```

- **Guarding against null is fine**
  That defines the domain on which function is defined
- **Guarding against violating dynamic rules is not fine**
  That would be controlling execution flow with exceptions

Partial Functions Revisited

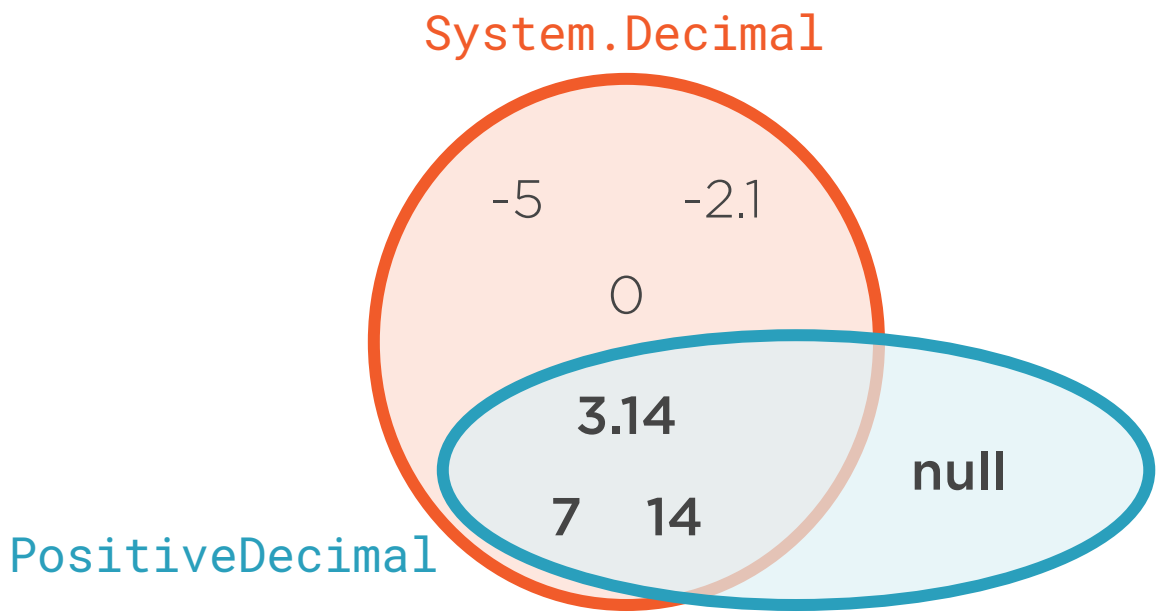
```
public void Deposit(decimal amount)
{
  if (amount <= 0)
    throw new ArgumentException(
        "Only positive amounts can be deposited.",
        nameof(amount));
  this.Balance += amount;
}
```

How long has this
negative value been around?



Partial Functions Revisited

```
public void Deposit(PositiveDecimal amount)
{
  if (amount <= 0)
    throw new ArgumentException(
        “Only positive amounts can be deposited.”,
        nameof(amount));
  this.Balance += amount;
}
```

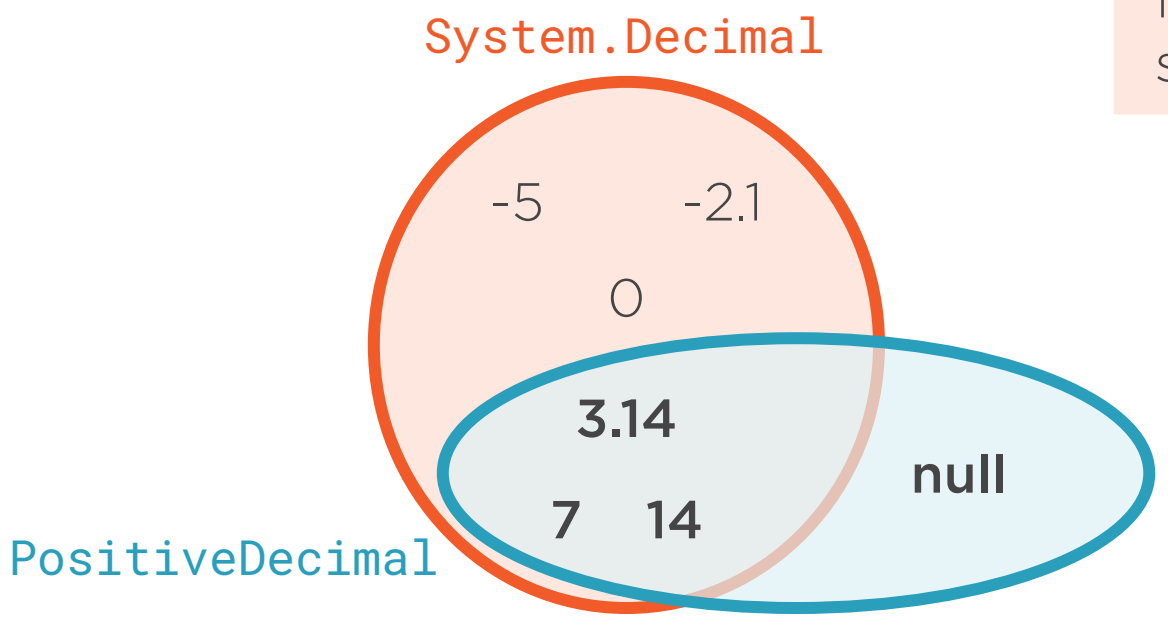


Partial Functions Revisited

```
public void Deposit(PositiveDecimal amount)
{
  if (amount == null)
    throw new ArgumentNullException(nameof(amount));
  this.Balance += amount;
}
```

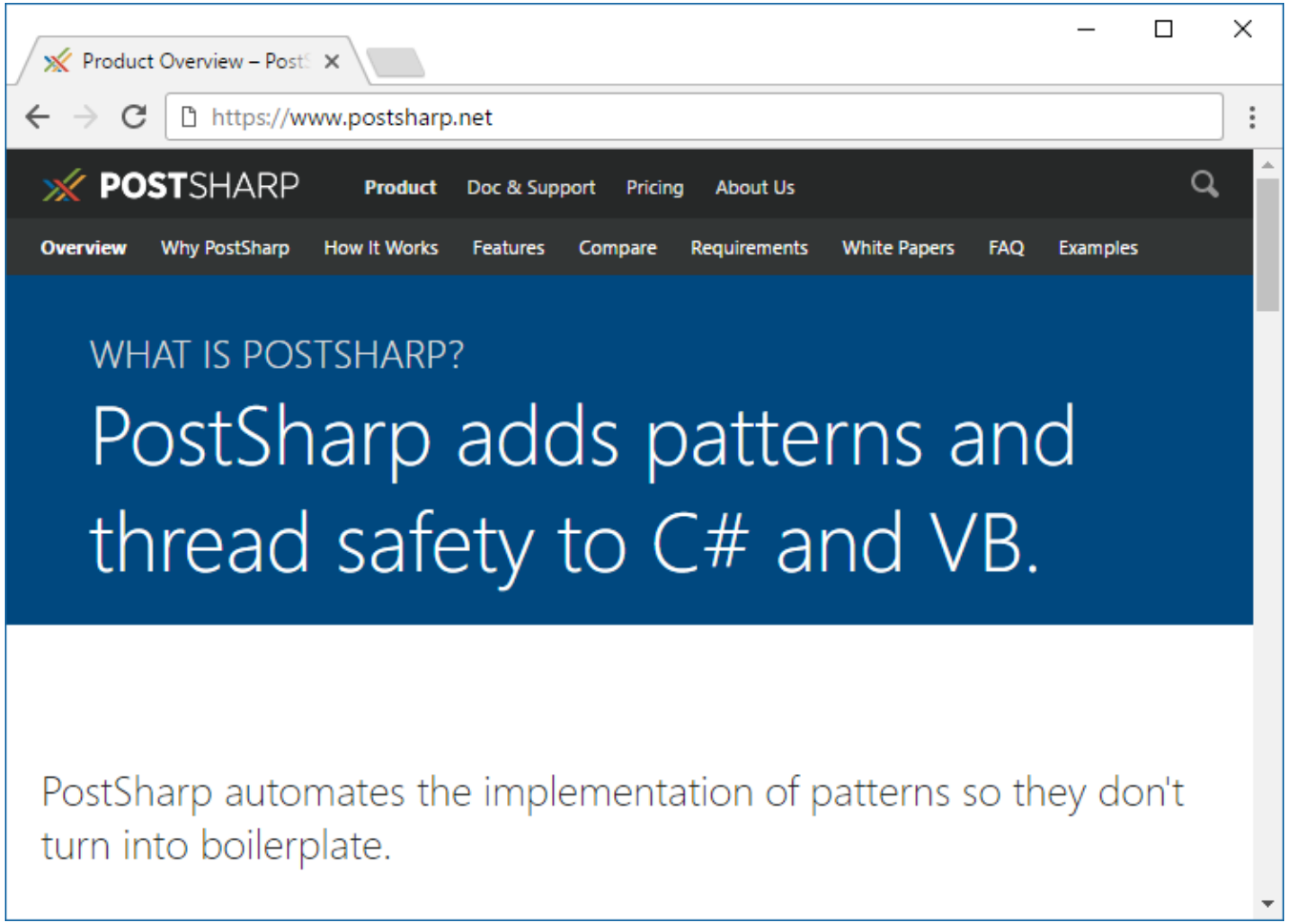
Null remains the only villain

Until we get the ability to forbid null syntactically



Guarding with (Paid) Tools

https://www.postsharp.net



Input Validation Attributes

NotNull
NotEmpty
Required
CreditCard
EmailAddress
Phone
RegularExpression
StringLength
EnumDataType
GreaterThan
LessThan
Positive
StrictlyPositive
Range

Guarding with (Paid) Tools

```
class Account
{
  private NonNegativeAmount Balance { get; set; }

  public void Deposit([NotNull] PositiveDecimal amount) =>
    this.Balance += amount;
}
```

Advice:

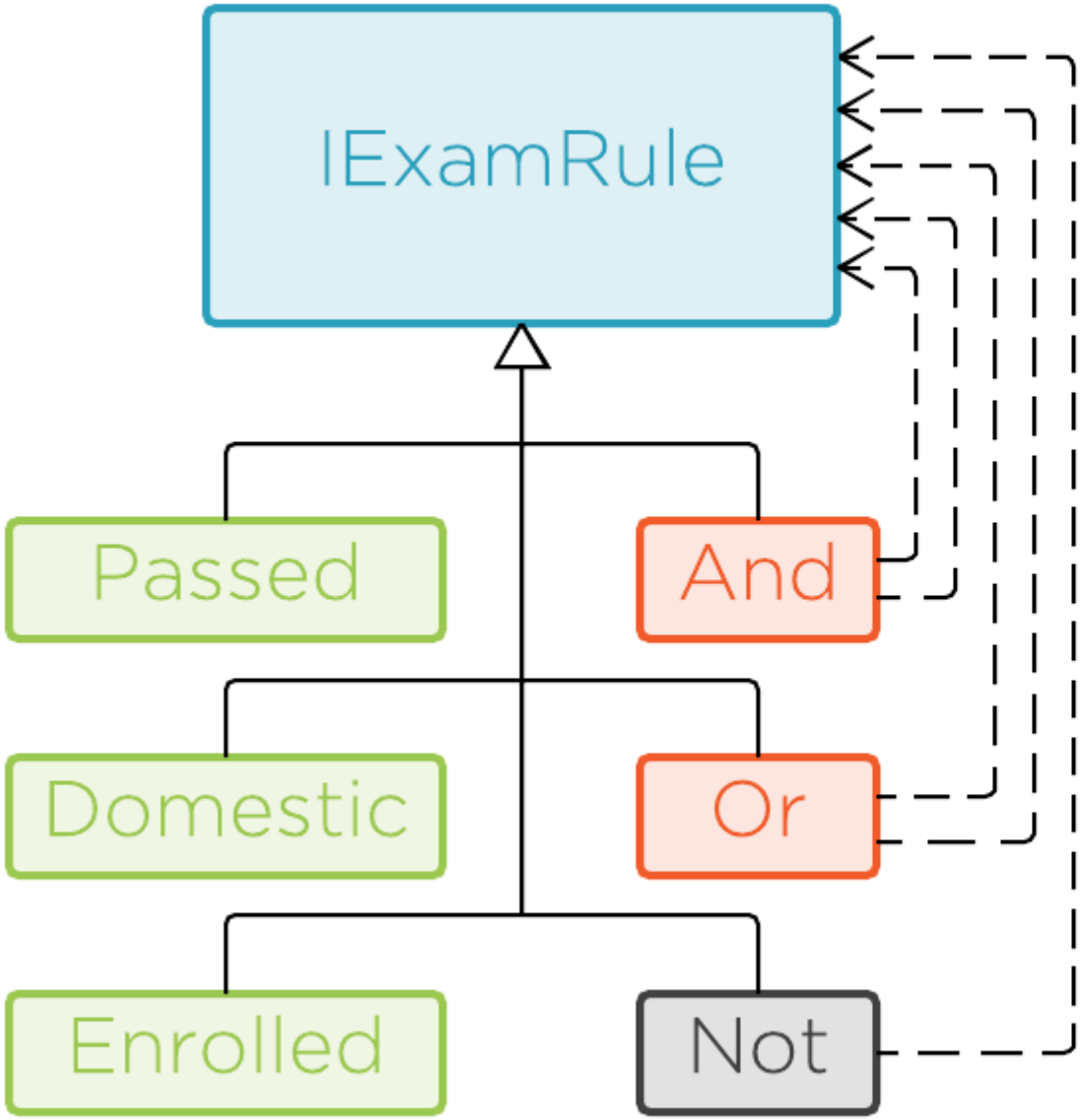
If you don't use null in code,
then null checks will never fail

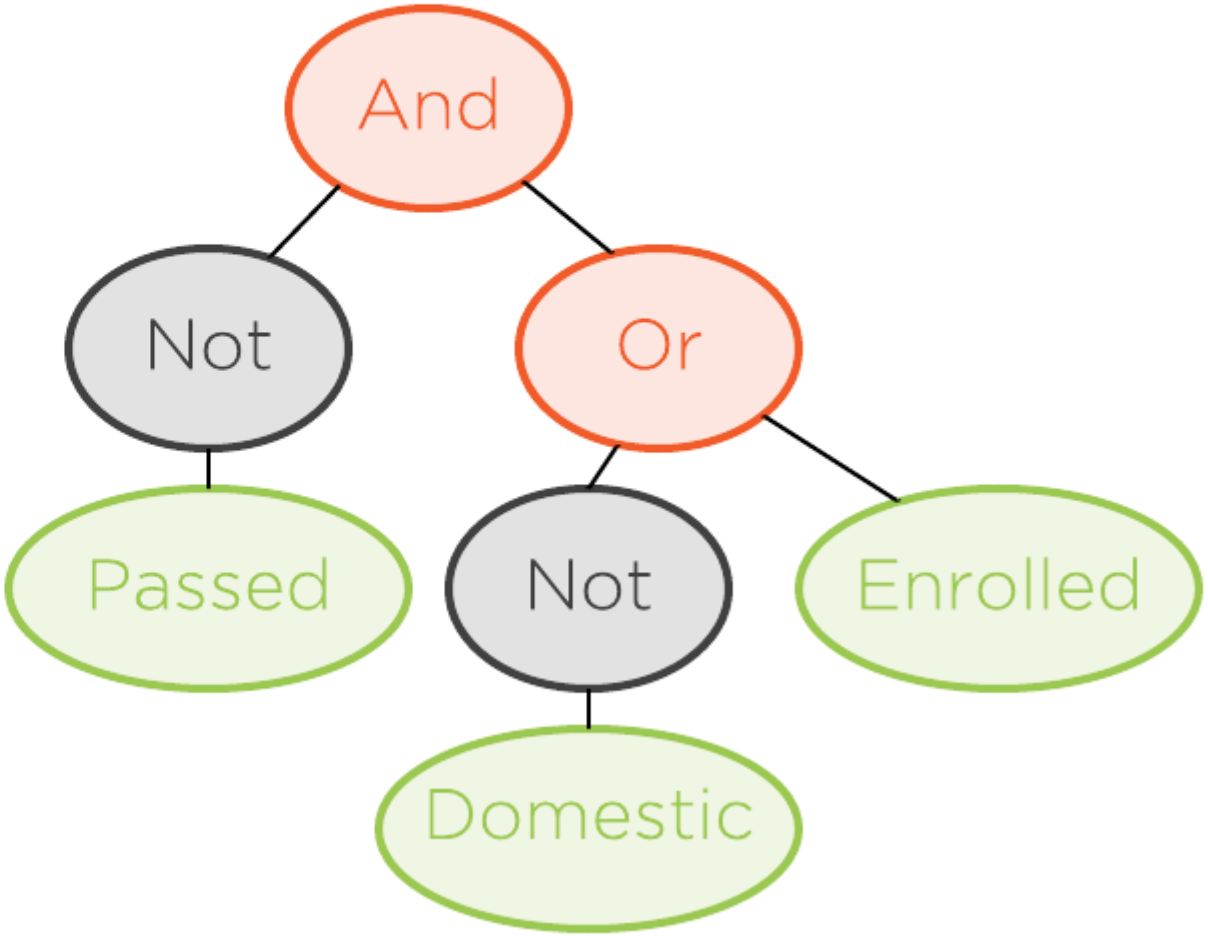
Do not pass null as argument
Do not return null from a method
Do not set variables to null

Null check remains the last guard

PostSharp offers attributes, leaving implementation clean

Expressions and Expression Builders





Separating View and Domain Models

Querying data

Bulk load
flat data

Reconstructing
domain objects

Load lots of
related objects

Summary

Programmatic vs. Mathematical Functions

- Both can be partial

Function fails if called outside its domain

- Typically throws an exception

Summary

Techniques to make functions total

- Define custom argument types with specific domain
- Apply Callback pattern and execute only when possible
- Filter objects that belong to domain

Later during this course

- Option<T> functional type
- Either<TLeft, TRight> functional type

Null checks as the last barrier

- Possibility to turn them into an aspect

Summary

Persisting complex model

- Every model, complex or not, has to be persisted at some point

Separation of models

- Domain layer defines separate Domain and View models
- Persistence layer defines its own Persistence model
- More details follow...

Next module

Encapsulation and Domain Rules