Starting with the First Bounded Context

Vladimir Khorikov

@vkhorikov | www.enterprisecraftsmanship.com

In This Module

- Inserting coins and notes
- Returning the money

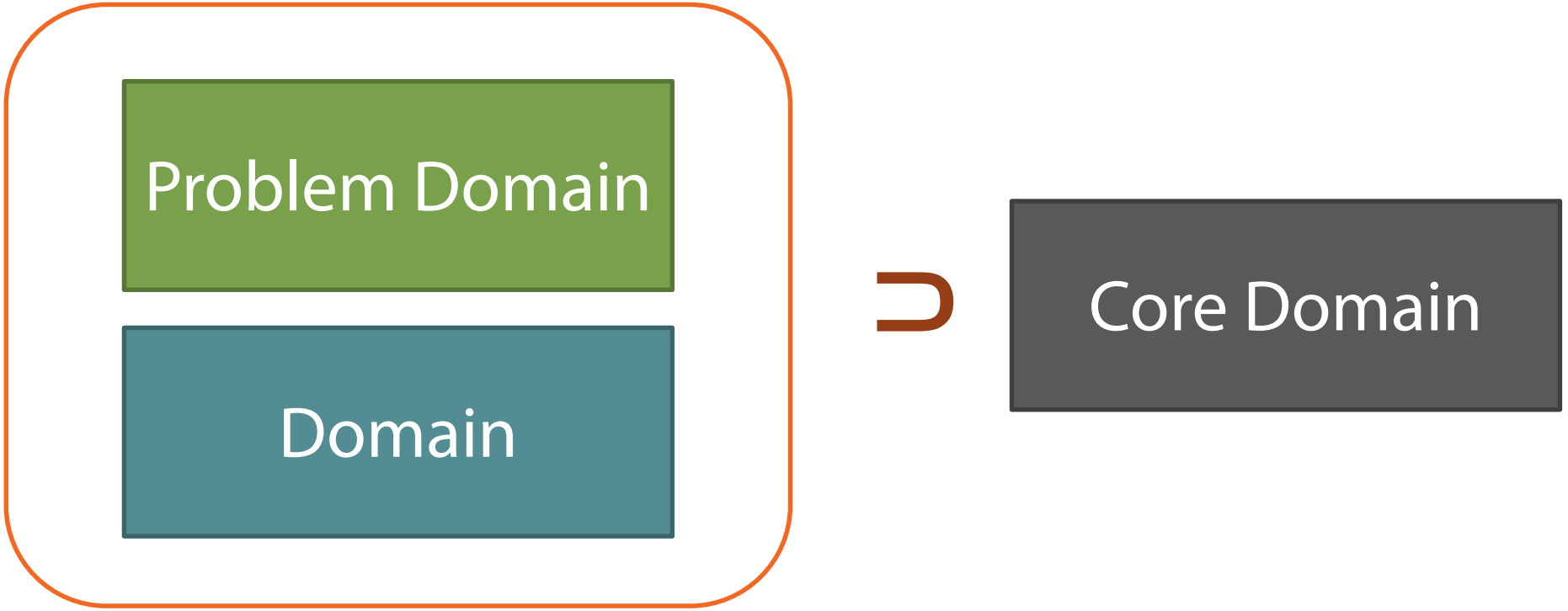
In This Module

Entities

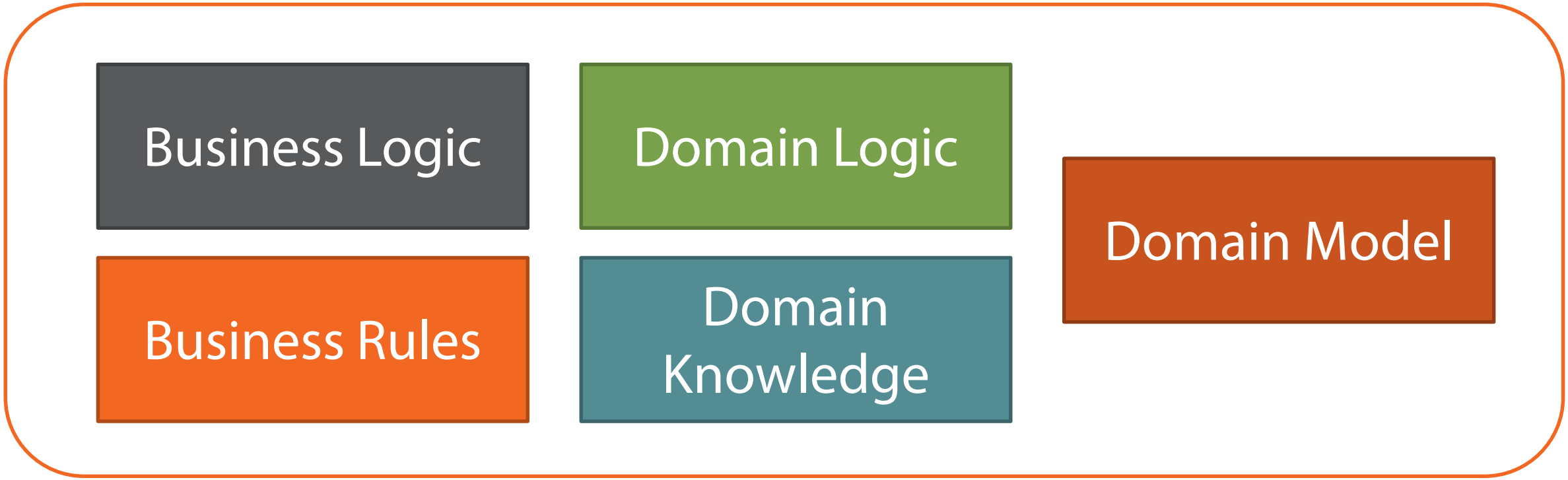
VS

Value Objects

Problem



Solution



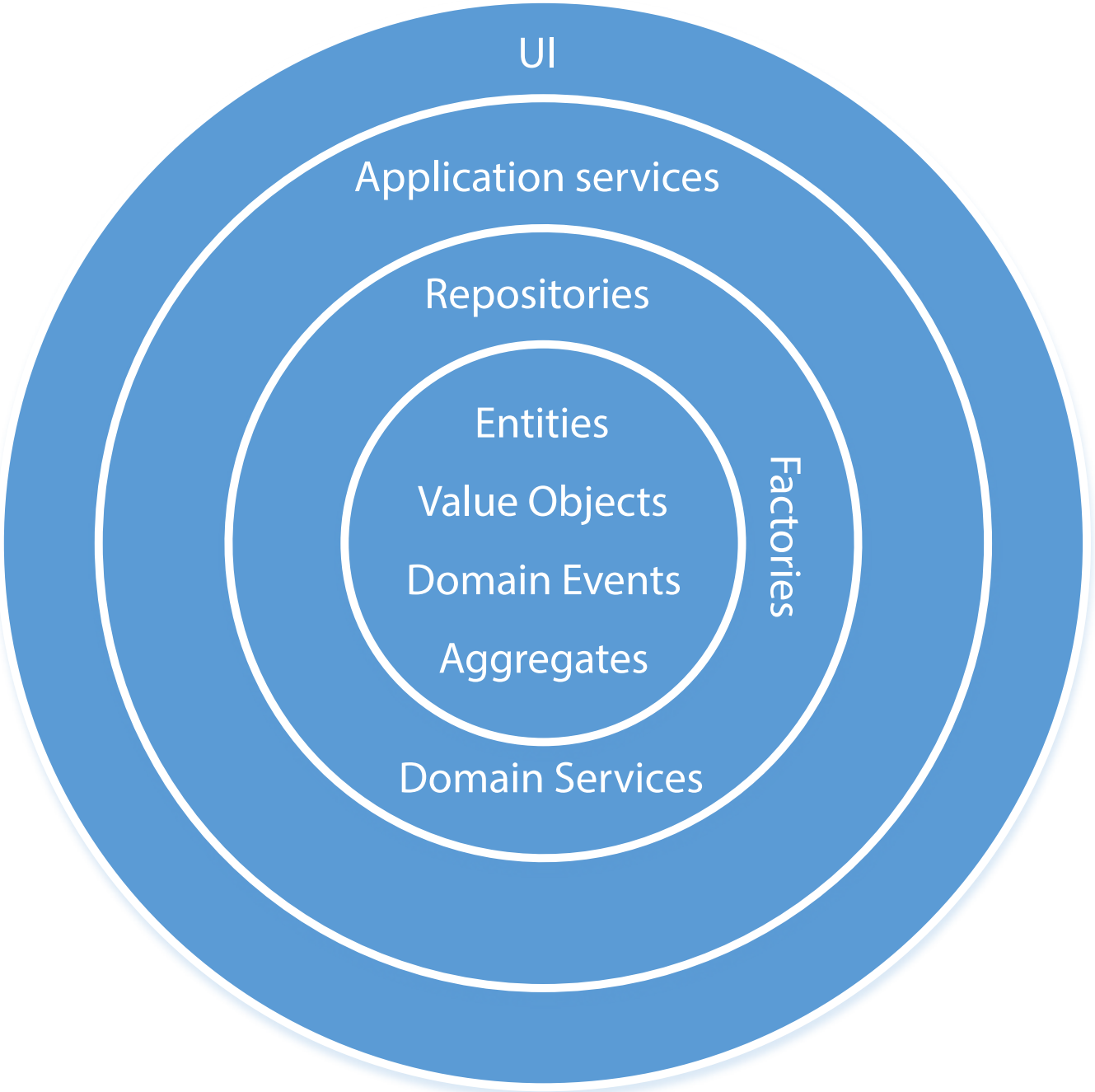
Vocabulary Used

Domain

- The problem we are working on

Domain Model

- The solution for the problem
- The artifact of the solution



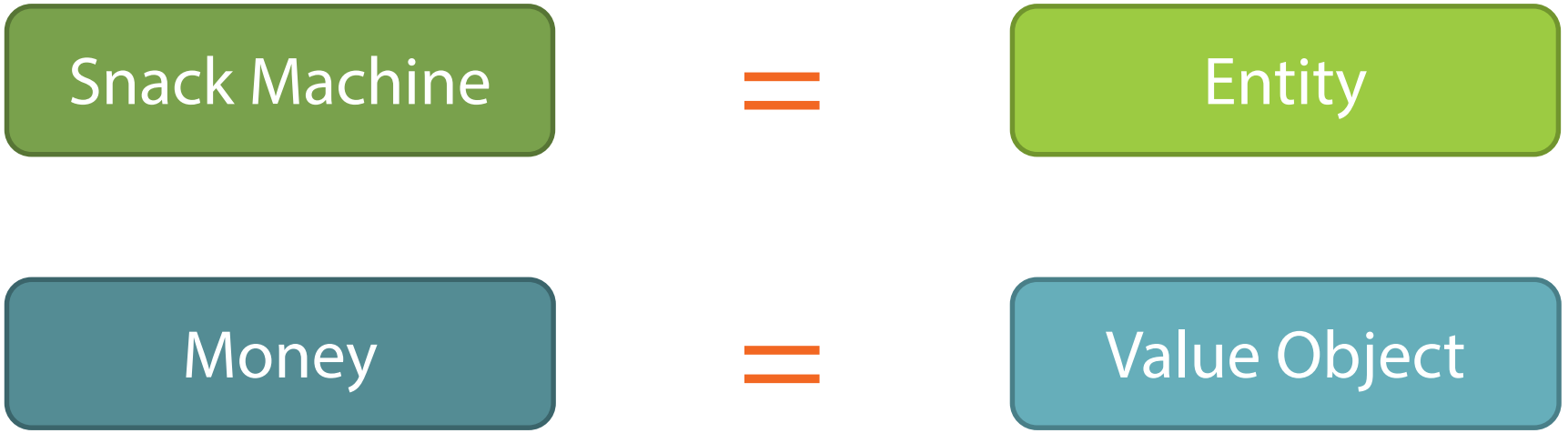
Problem Description

- Insert money into the Machine
- Return the money back
- Buy a snack

Recap: Starting with Snack Machine

- Start with the core domain
- Don't introduce several bounded contexts upfront
- Always look for hidden abstractions

Entities vs. Value Objects

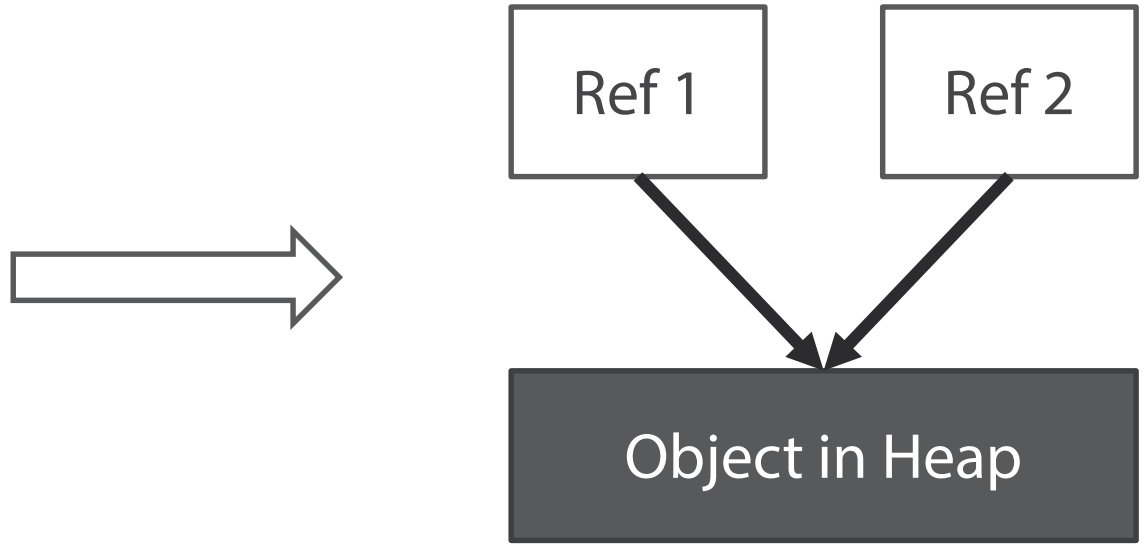


Types of Equality

Identifier equality

Reference equality

Structural equality



Types of Equality

Identifier equality

Object 1	Object 2
Id: 5	Id: 5

Reference equality

Structural equality

Types of Equality

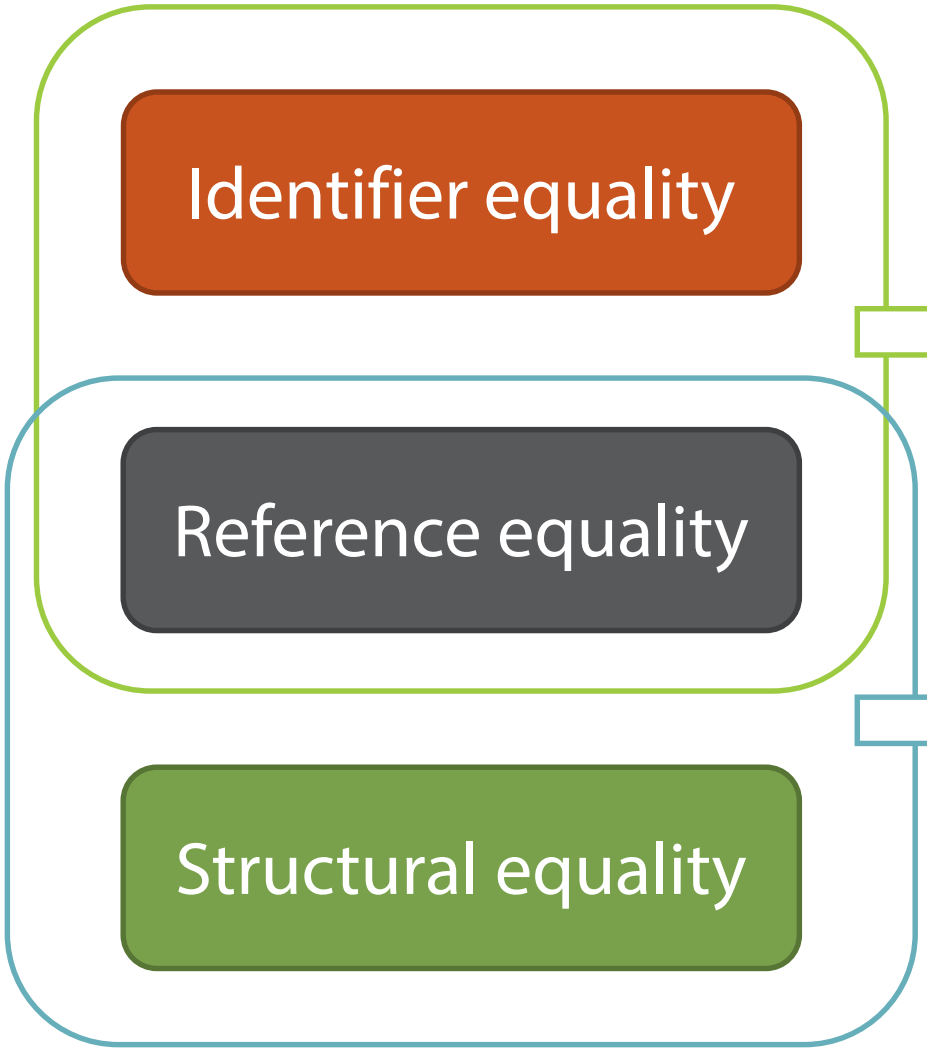
Identifier equality

Reference equality

Structural equality

Object 1	Object 2
Name: "A"	Name: "A"
Country: "B"	Country: "B"

Types of Equality



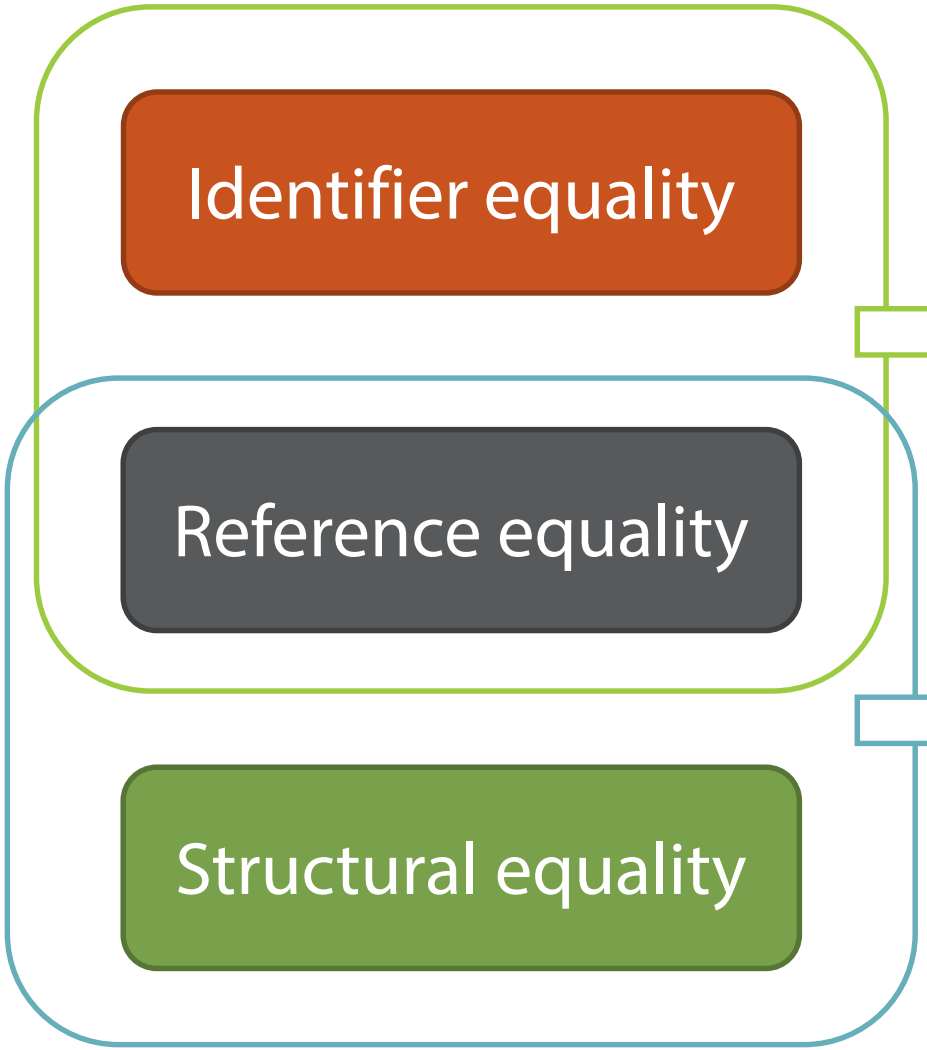
Entities

- Have inherent Identity

Value Objects

- Don't have an Id field
- Can be treated interchangeably

Types of Equality



Entities

- Have inherent Identity

Value Objects

- Don't have an Id field
- Can be treated interchangeably

Entities vs. Value Objects

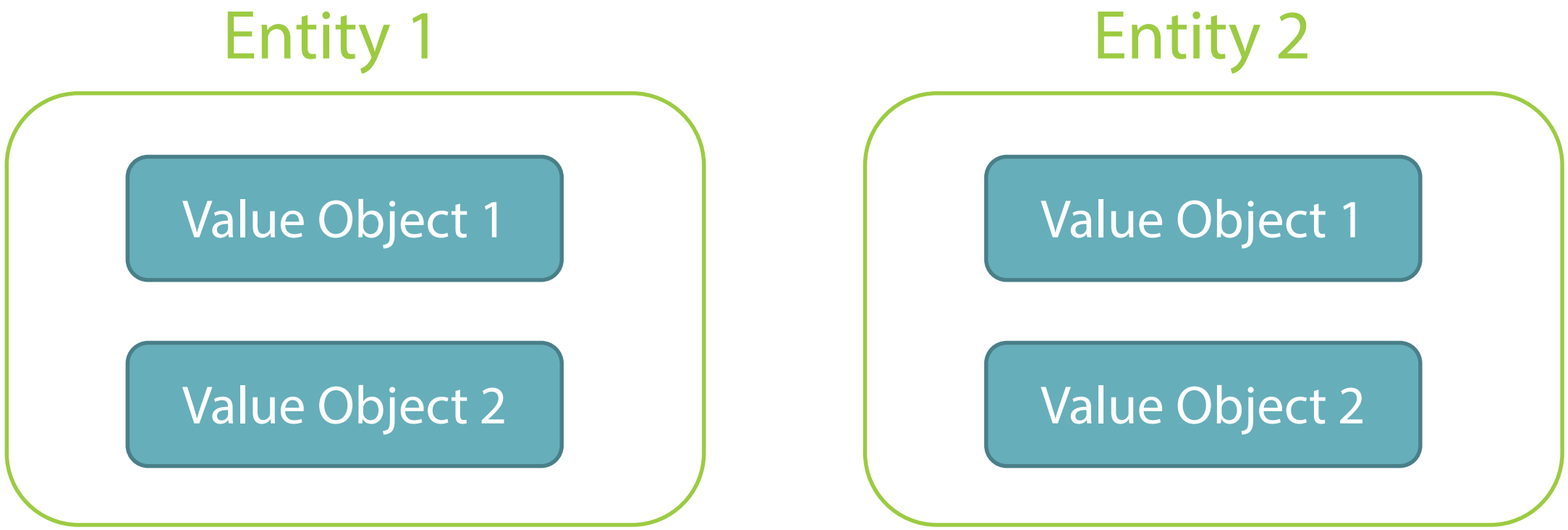
Entities

- Have inherent Identity
- Mutable

Value Objects

- Don't have an Id field
- Can be treated interchangeably
- Immutable

Lifespan



How to Recognize a Value Object

Entities

Value Objects

How to Recognize a Value Object

Structural equality

Integers = Value Objects

```
public void Method()
{
    int value1 = 5;
    int value2 = 5;
}
```

```
public void Method()
{
    Money value1 = new Money(5);
    Money value2 = new Money(5);
}
```

Prefer value objects to entities

- Value objects are light-weight
- Put most of business logic to value objects
- Entities act as wrappers

Entity Base Class

Interface:

```
public interface IEntity
{
    int Id { get; }
}
```

Code duplication

Doesn't show proper relations between entities

Entity Base Class

```
public interface IEntity
{
}

public class Entity1 : IEntity
{
}

public class Entity2 : IEntity
{
}
```

"Can-do" relationship

```
public abstract class Entity
{
}

public class Entity1 : Entity
{
}

public class Entity2 : Entity
{
}
```

"Is-a" relationship

The use of the Equals method:

```
public void EqualsUsageExample(List<Entity> entities, Entity entity)
{
    bool contains = entities.Contains(entity);
}
```

The use of the equality operator:

```
public void EqualityOperatorUsageExample(Entity entity1, Entity entity2)
{
    bool equal = entity1 == entity2;
}
```

Recap: Entity Base Class

Entity base class

- Id property
- Equality members

Value Object Base Class

Value Object base class

- Don’t have an Id property
- Can’t place equality members to the base class

Recap: Value Object Base Class

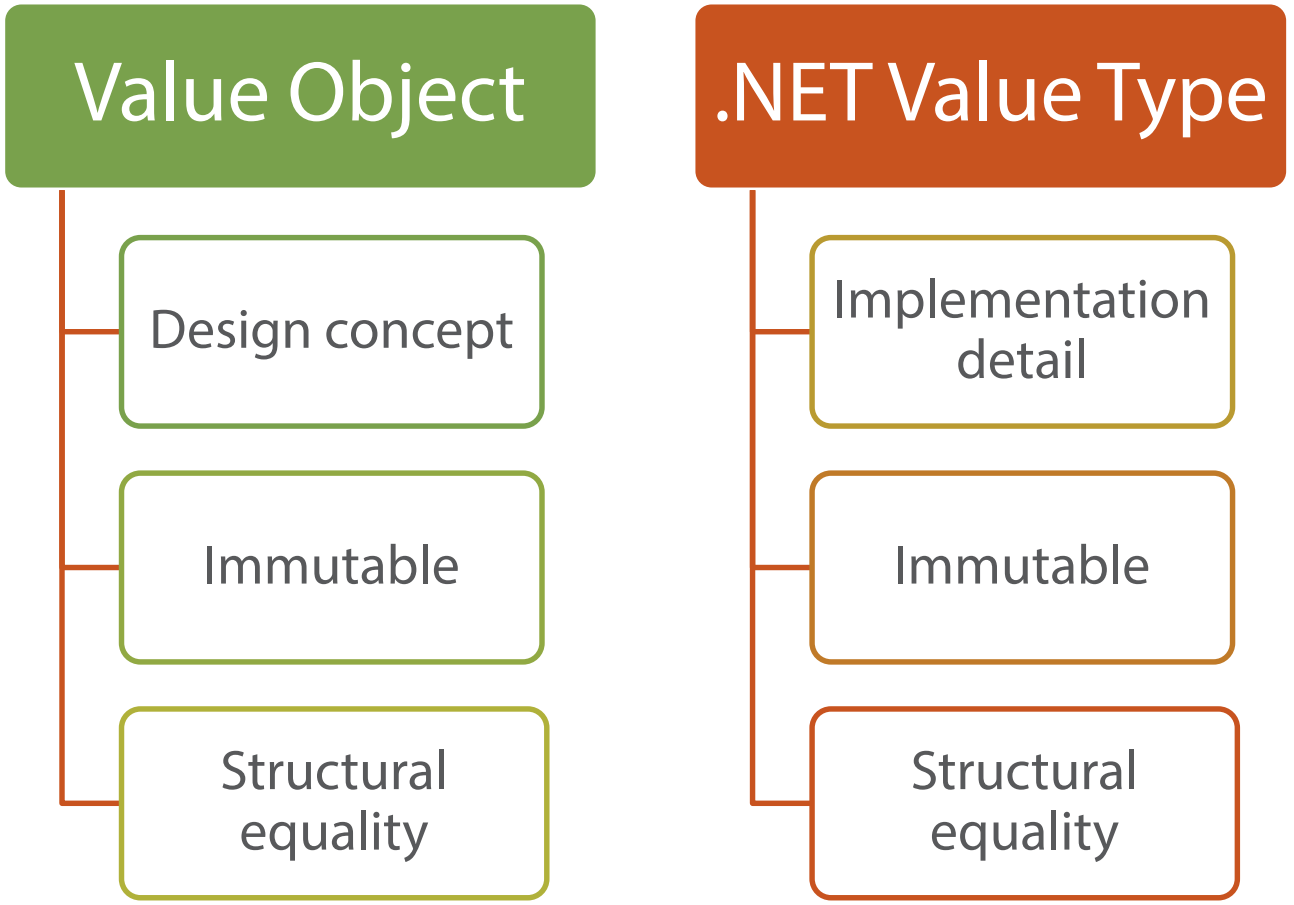
Entity base class

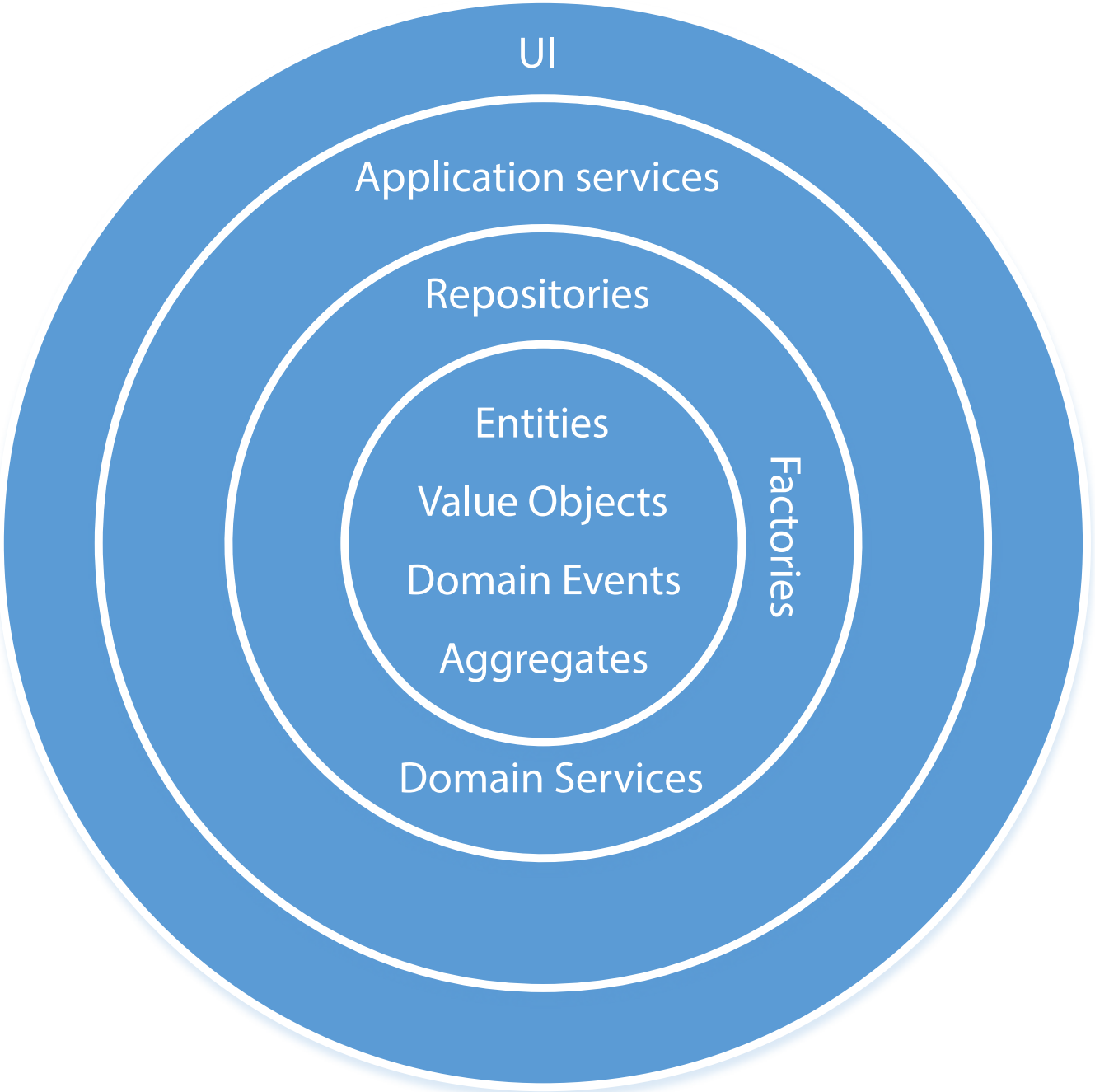
- Reference equality
- Identifier equality
- Should have an identity
- Single place for equality members

Value Object base class

- Reference equality
- Structural equality
- Don't have an identity
- No single place for equality members

Value Objects vs. .NET Value Types





When to Write Unit Tests

Test-First

- Know how the code should look like

Code-First

- Experimenting with code

Details at
http://bit.ly/1XF0J6H

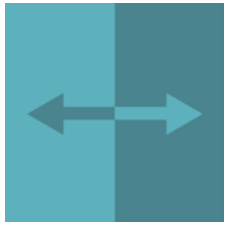
Recap: Money and Snack Machine

```
public sealed class Money : ValueObject<Money>
{
    public int OneCentCount { get; }
    public int TenCentCount { get; }
    public int QuarterCount { get; }
    public int OneDollarCount { get; }
    public int FiveDollarCount { get; }
    public int TwentyDollarCount { get; }
}
```

```
public sealed class SnackMachine : Entity
{
    public Money MoneyInside { get; private set; } = None;
    public Money MoneyInTransaction { get; private set; } = None;
}
```

Recap: Money and Snack Machine

TDD



DDD

- [ ] Code-first approach for experiments
- [ ] Test-first approach after the experiments
- [ ] Always cover the model with unit tests

Summary

- Start off by working on the core domain
- Begin with a single bounded context
- Constantly search for hidden abstractions
- 3 distinctions between Entities and Value Objects
  - Reference vs structural equality
  - Mutability vs immutability
  - Lifespan: Value Objects should belong to Entities

Summary

- Compare Value Object to Integers
- Move logic from Entities to Value Objects
- Don’t use .NET structs to represent Value Objects
- TDD and DDD

In the Next Module

User Interface and Persistence