Doing More with Custom Types

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren | www.xpirit.com/gill

Agenda

Grouping classes in namespaces

Introducing static data

Working with null

Understanding garbage collection

Grouping Classes in Namespaces

There are a lot of types...

Organized in “folders”: namespaces

Avoids naming collisions

Namespaces

- Keep class names separate
- Used throughout .NET
- Organize our own classes in custom namespaces
- Make namespace available through using directive

```
namespace BethanysPieShop.HR
{

    public class Employee

    { }

}
```

Putting a Class into a Namespace

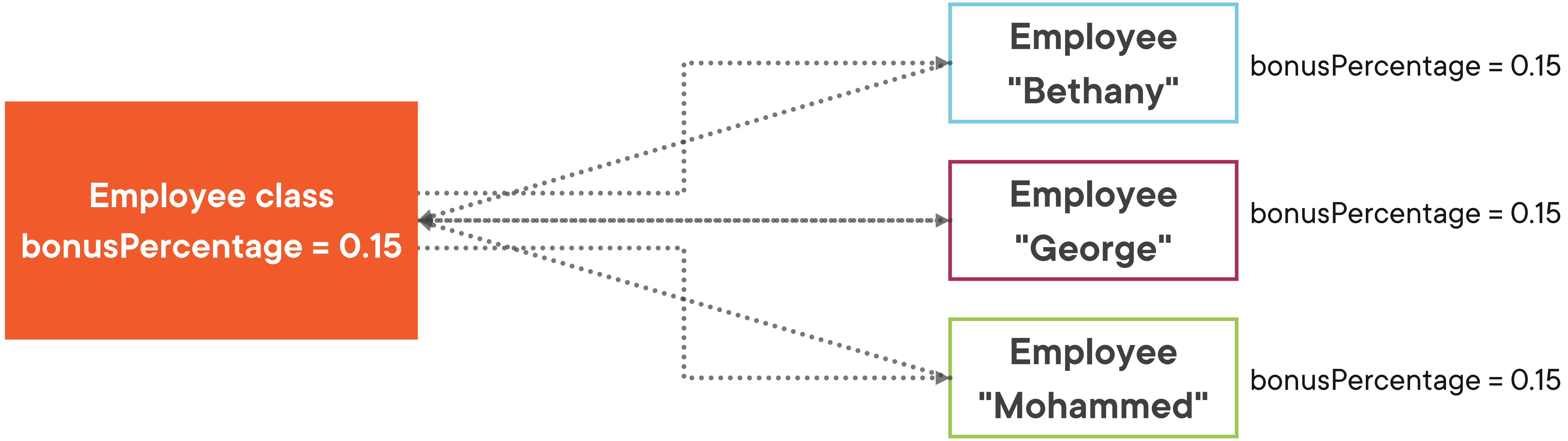
Demo

Grouping multiple classes into namespaces

Introducing the using directive

Introducing Static Data

Objects and Their Data



```java
public class Employee
{

    public static double bonusPercentage = 0.15;

}
```

Adding Static Data

```
public class Employee
{

    public static double bonusPercentage = 0.15;

    public static void IncreaseBonusPercentage(double newPercentage)
    {

        bonusPercentage = newPercentage;

    }
}
```

Changing Static Data with a Static Method

```
static void Main(string[] args)
{
    Employee.IncreaseBonusPercentage(0.2);//Note the class name, not an object!

}
```

Invoking a Static Method

Not on an object but on the class instead

Calling Static Methods

```
class Program
{

    static void Main(string[] args)
    {

        PrintAllEmployeeList();
    }

    public static void PrintAllEmployeeList()
    {
        ...
    }

}
```

Demo

Adding static data

Creating a static method

Using the static functionality from our class

Constant variable value

- Value that can’t be changed in the class
- Use the const keyword
- Is static by default

```
public class Employee
{

    public const double bonusPercentage = 0.15;//We never want this to change!

}
```

Defining a const Value

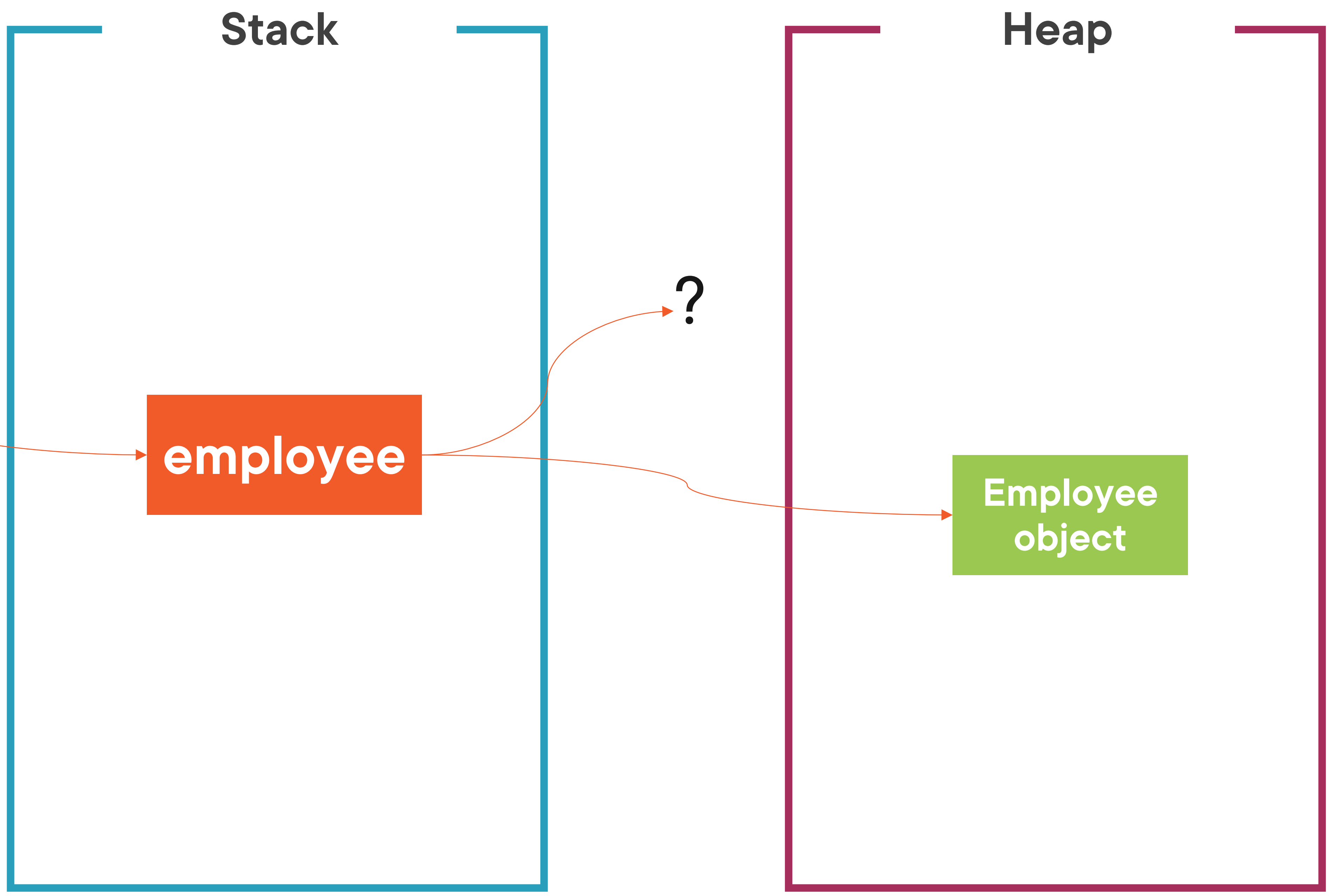
Demo

Adding a const value

Working with null

Understanding null

```
Employee employee;
//employee is null
employee = new Employee();
```



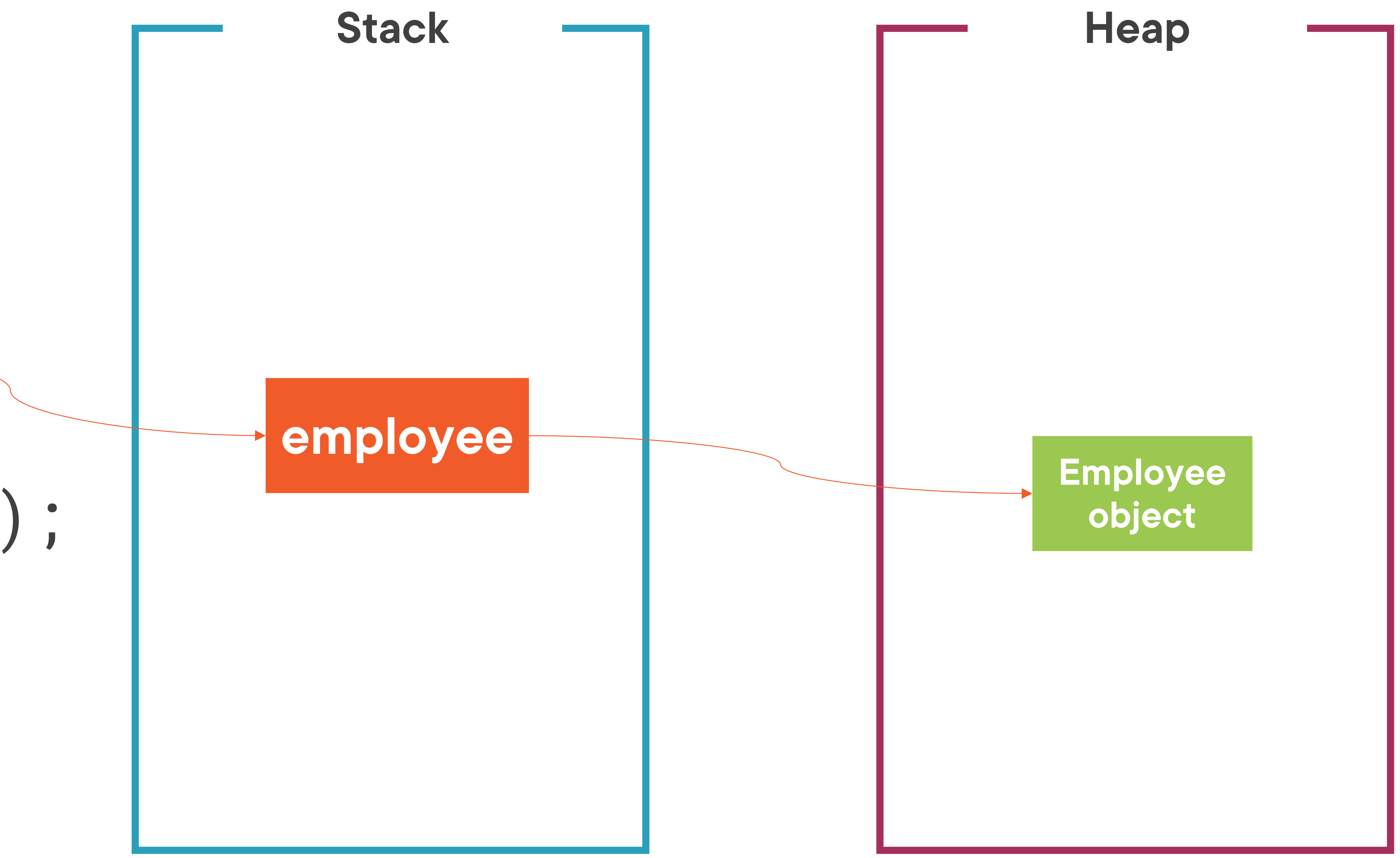
Using a Non-initialized Value

```
Employee employee = null;
employee.PerformWork();//runtime error
```

Setting the Reference to null

```
Employee employee;
//employee is null
employee = new Employee();

employee = null;
```



```
int? a = 10;

int? b = null;

if (b.HasValue)
{

    Console.WriteLine("We have a value");

}
```

Introducing Nullable Value Types

Demo

Handling null references at runtime

Working with nullable types

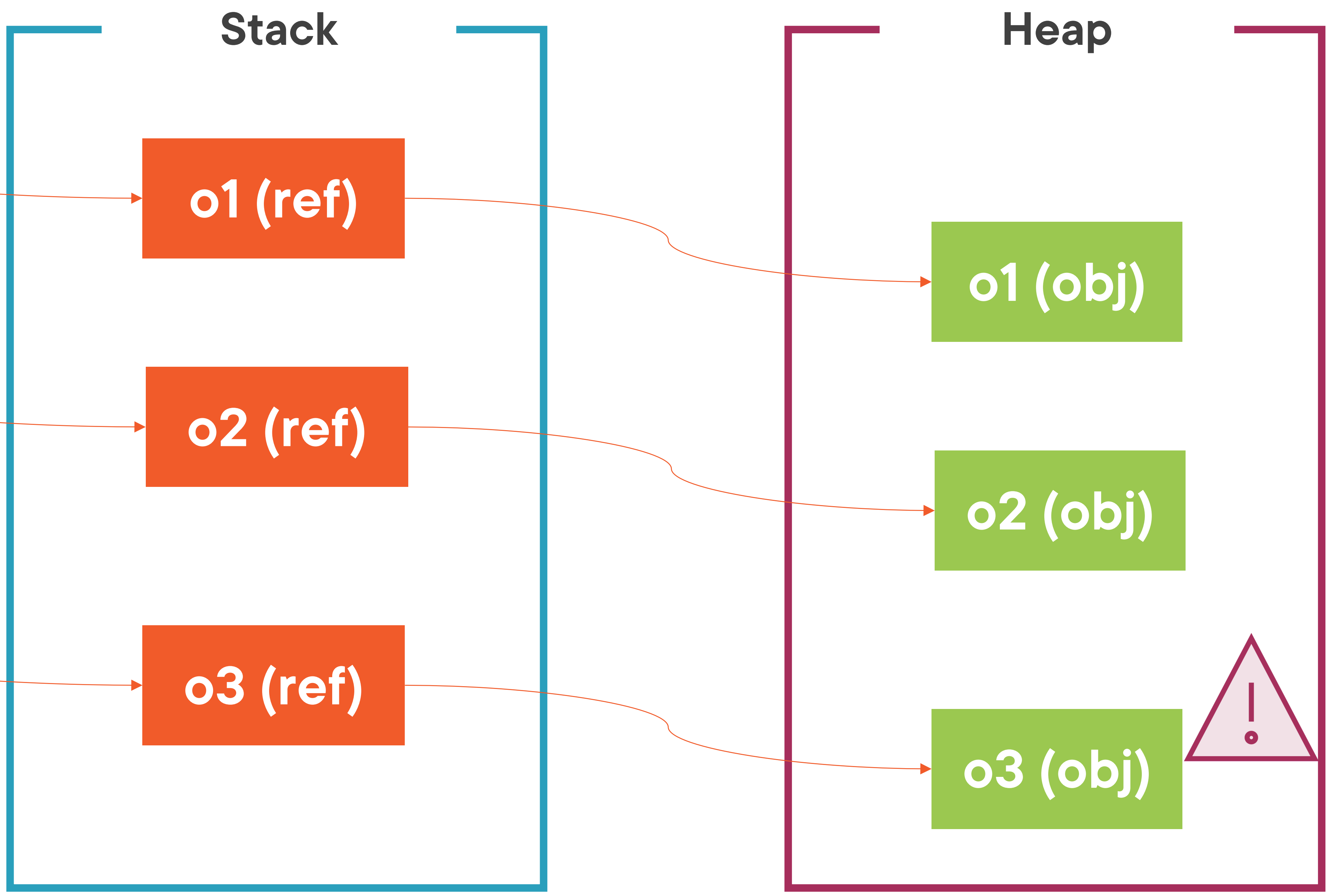
Understanding Garbage Collection

Working with Objects

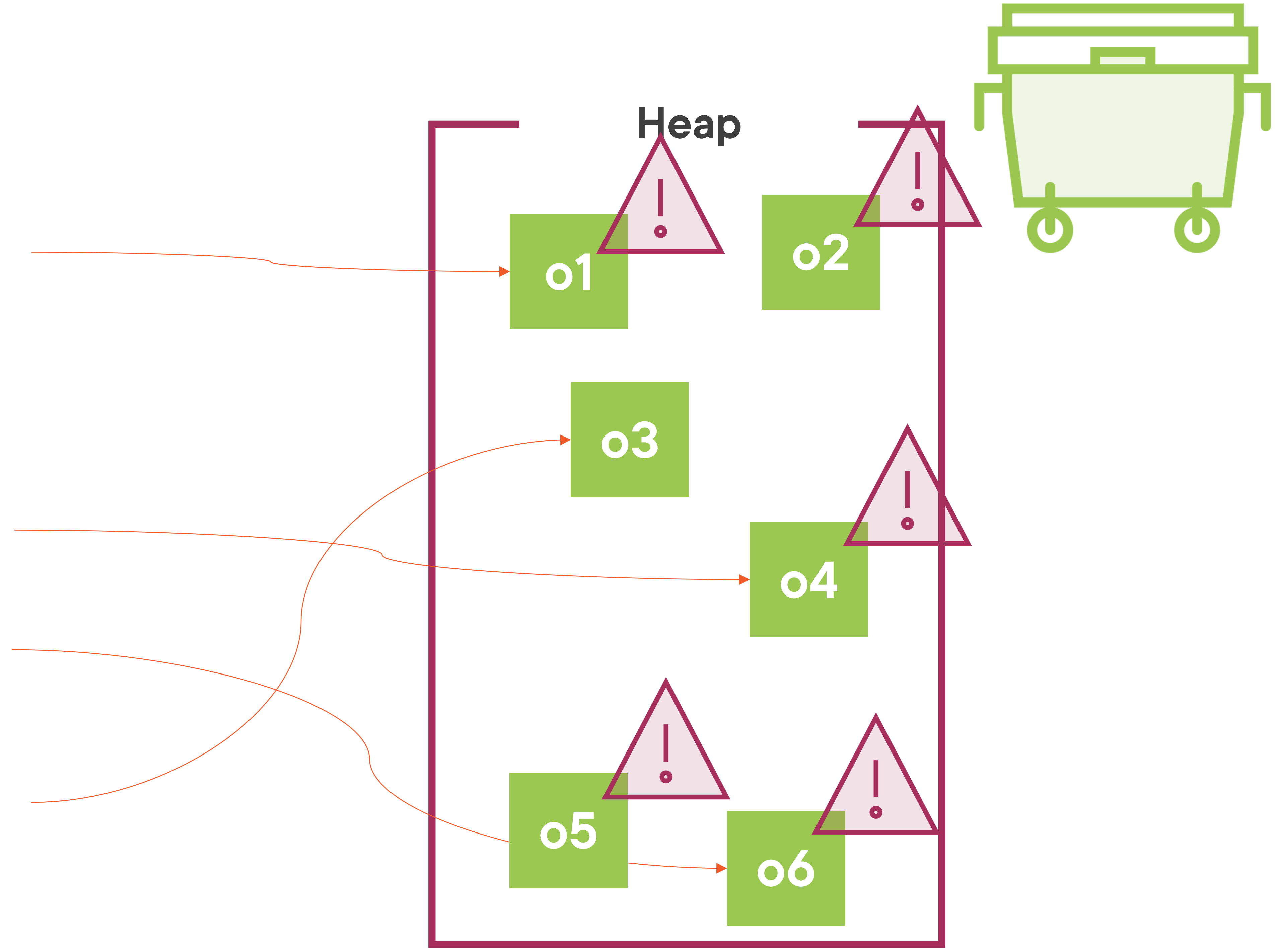
```
object o1 =
   new object();

object o2 =
   new object();

object o3 =
   new object();
```



Understanding Garbage Collection



Understanding Garbage Collection

Automatic process, part of CLR

Works with several generations

Can be triggered using GC.Collect(), often not required

Demo

Looking at garbage collection

Summary

Namespaces are used to group classes

Static data is class-level data

References can be null

- Can cause null reference exceptions
- Garbage collection will clean up unused objects

Resources

Other relevant courses in the C# path:

- Object Oriented development in C#
  - Deborah Kurata
- Working with Nulls in C#
  - Jason Roberts

Up next:

Adding inheritance