Using and Testing Classes

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren

Overview

Working with objects

Adding static members

Exploring the interface

Writing unit tests

Working with Objects

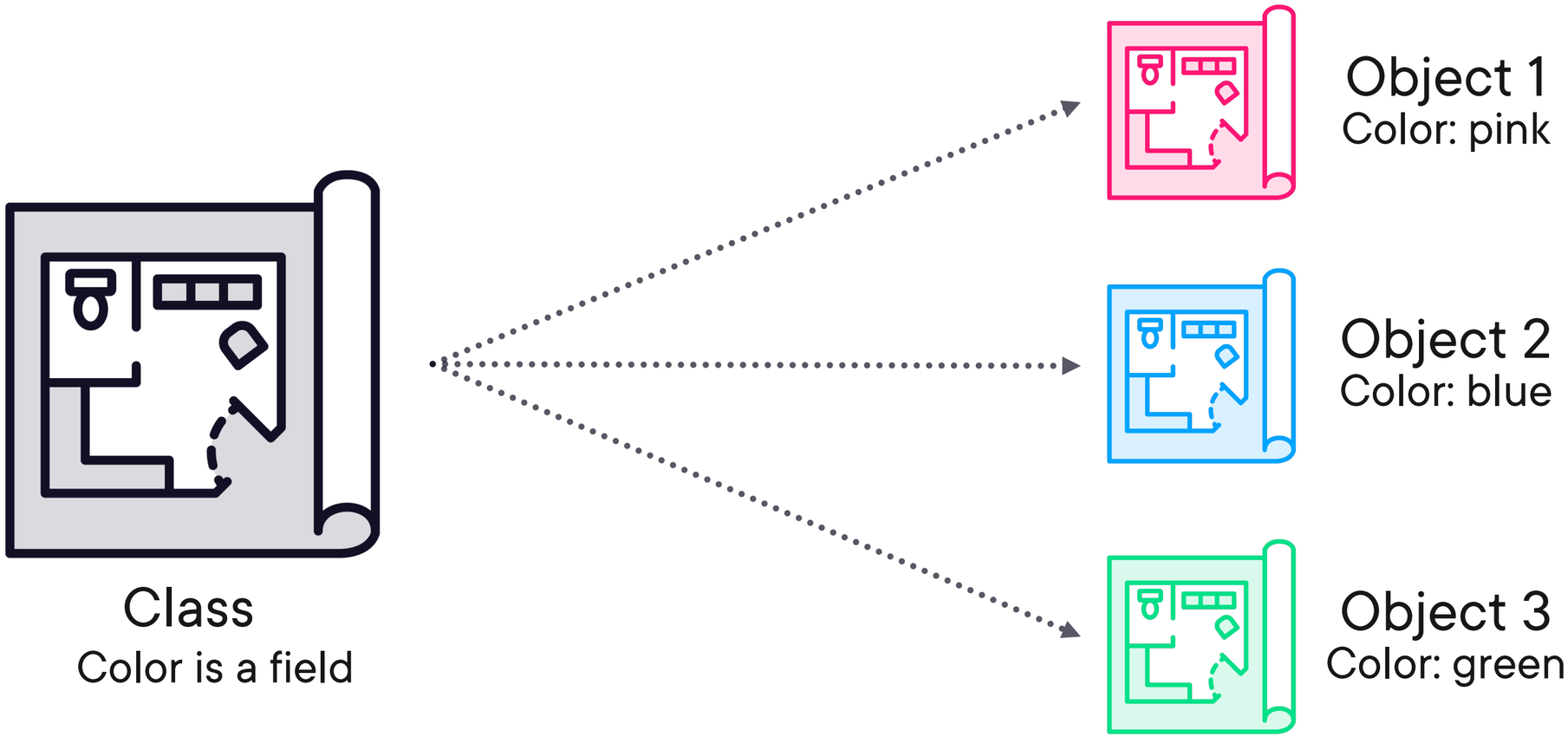
From Classes to Objects

Classes are abstract description, not real objects

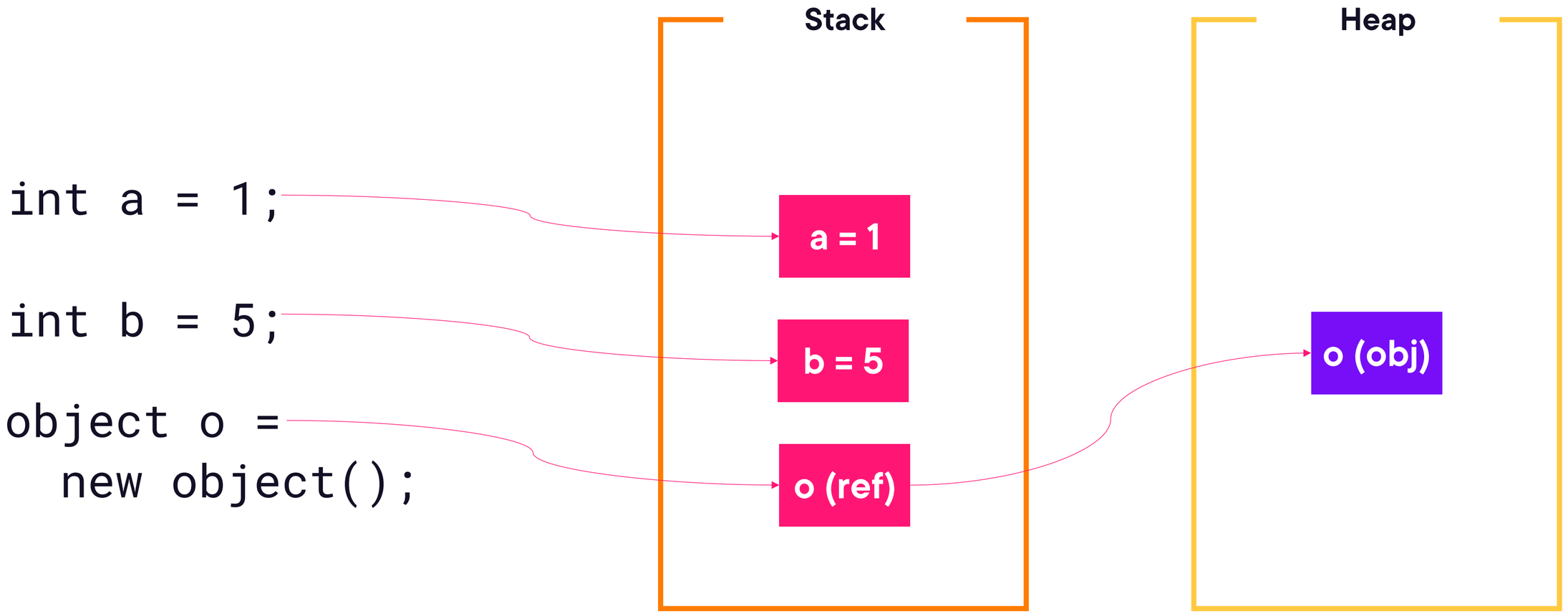
Objects are created on heap and give values

Reference points to object on heap

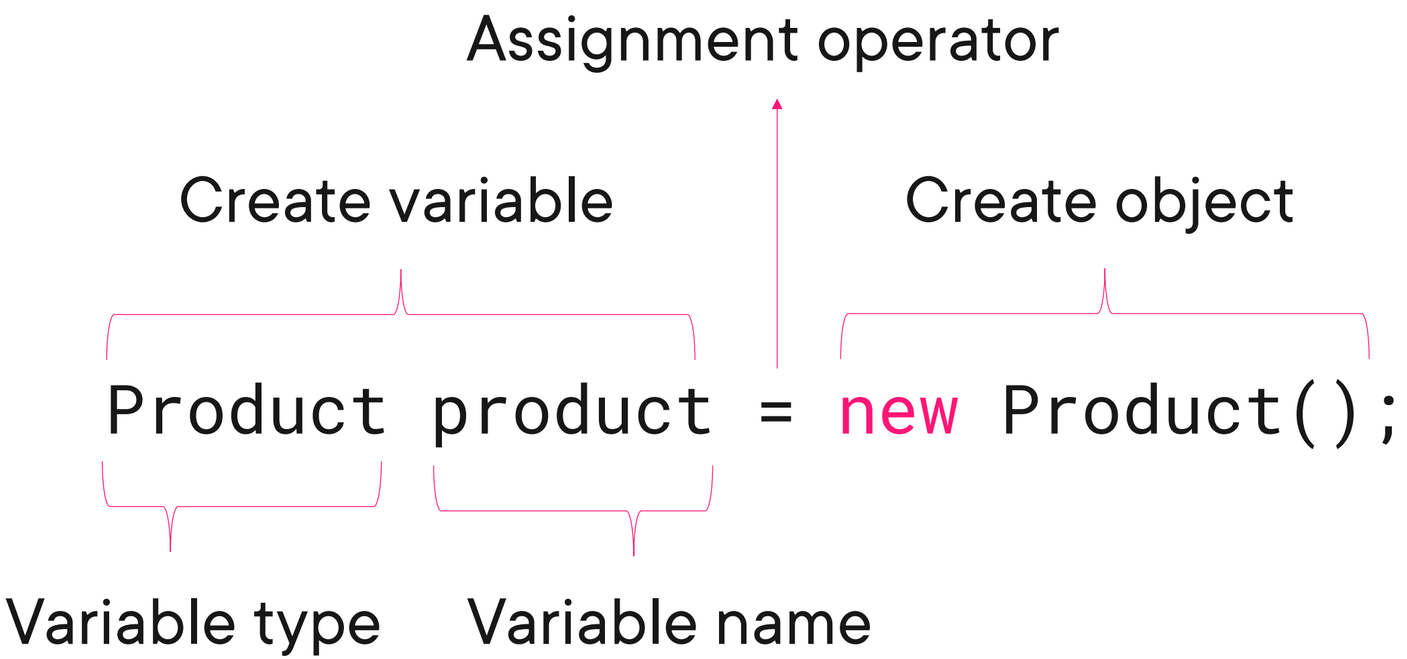
From Classes to Objects

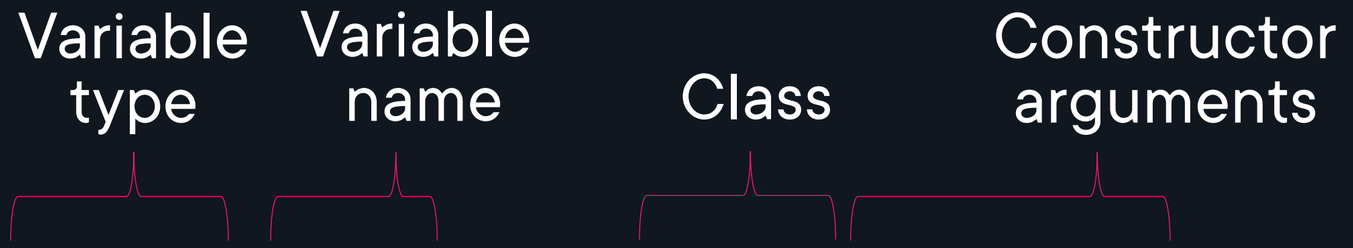


Classes Are Reference Types



Creating a New Object





Using a Constructor with Parameters

```
product.UseProduct();
```

- Invoking a method

```
product.Name = "Milk";
```

- Changing a property

```
int number = product.UpdateStock();
```

- Returning a value from a method

```
Product product = new ();
```

New shorthand Syntax

Demo

Creating objects

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }

    public Product()
    {
    }
}
```

Using Object Initialization

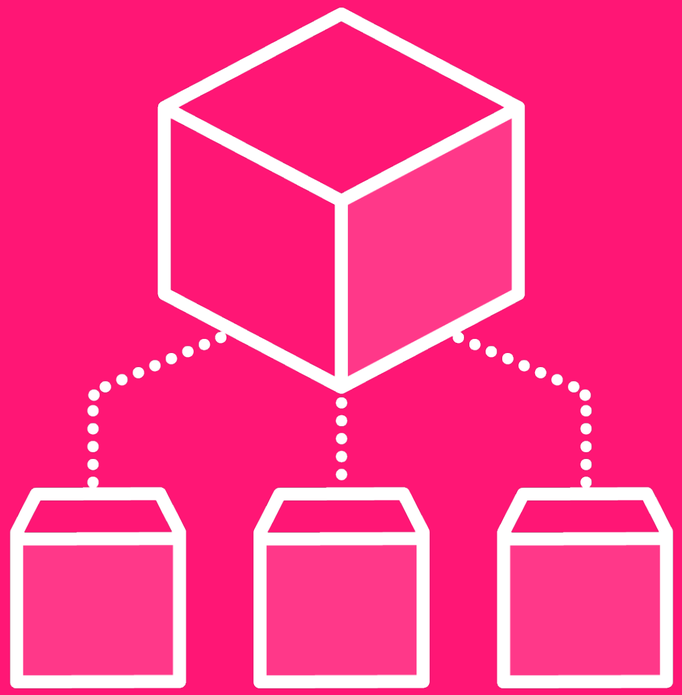
```
Product p = new Product() { Name = "Eggs", Id = 123 };
```

Using Object Initialization

Demo

Using object initializers

Adding Static Members



Working with objects

Objects get their own copy of data of a class

Data can also be stored at class level

Static members are used for this

```
public class Product
{
    public static int StockTreshold = 5;
}
```

Adding static Data

```
public partial class Product
{
    public static int StockTreshold = 5;

    public static void ChangeStockTreshold(int newStockTreshhold)
    {
        //Do something with static data here
    }
}
```

Adding a static Method

```
Product.StockTreshold = 10;
Product.ChangeStockTreshold(5);
```

Using a static Member

Only accessible through class since it’s created at class-level

Adding static members to the Product class

Exploring the Interface of the Application

Demo

Looking at the console application

Adding Support for Loading Data

“Hey, it’s Bethany! Would it be possible that we load in the current inventory from a file?”

“Yes, we can develop that!”

A New Requirement

Start from a file with the current inventory

Have a class that reads the file and create Products → Repository

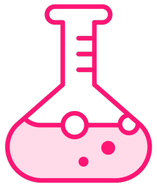
Repository contains all code to abstract away interaction with the data

Loading data using a repository

Writing Tests for the Class

Adding Unit Tests

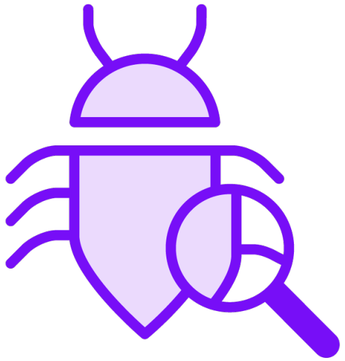
Code to test other code



Test small, isolated portions of the code (typically methods)

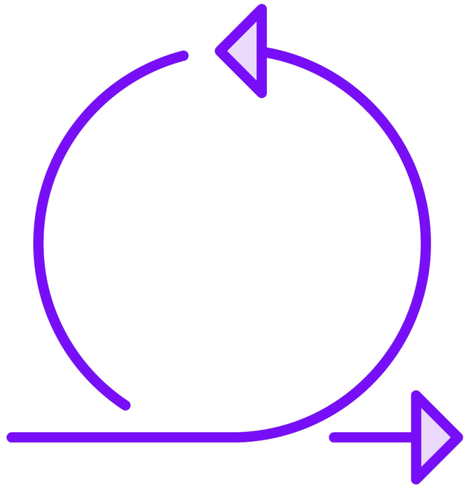
Validate result

Advantages of Unit Tests

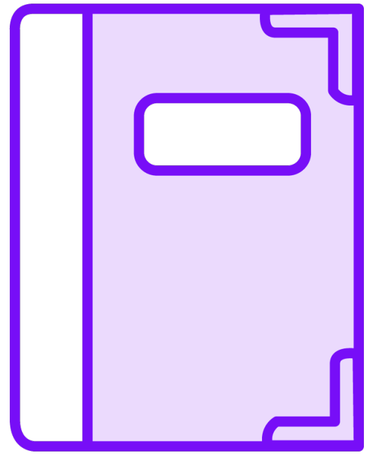


Find bugs

Change without fear of breaking something



Improve quality



Documentation of the code

Structure of a Unit Test

Writing a Unit Test

```
public void IncreaseStock_CorrectAmountInStock()
{
    //Arrange
    Product product = new Product();

    //Act
    product.IncreaseStock(100);

    //Assert
    Assert.Equal(100, product.AmountInStock);
}
```

Demo

Adding tests for the Product class

Summary

Objects are instantiated based on a class

Classes are reference types

Using unit tests, we can validate the correct working of a class

Up Next:

Working with Class Hierarchies