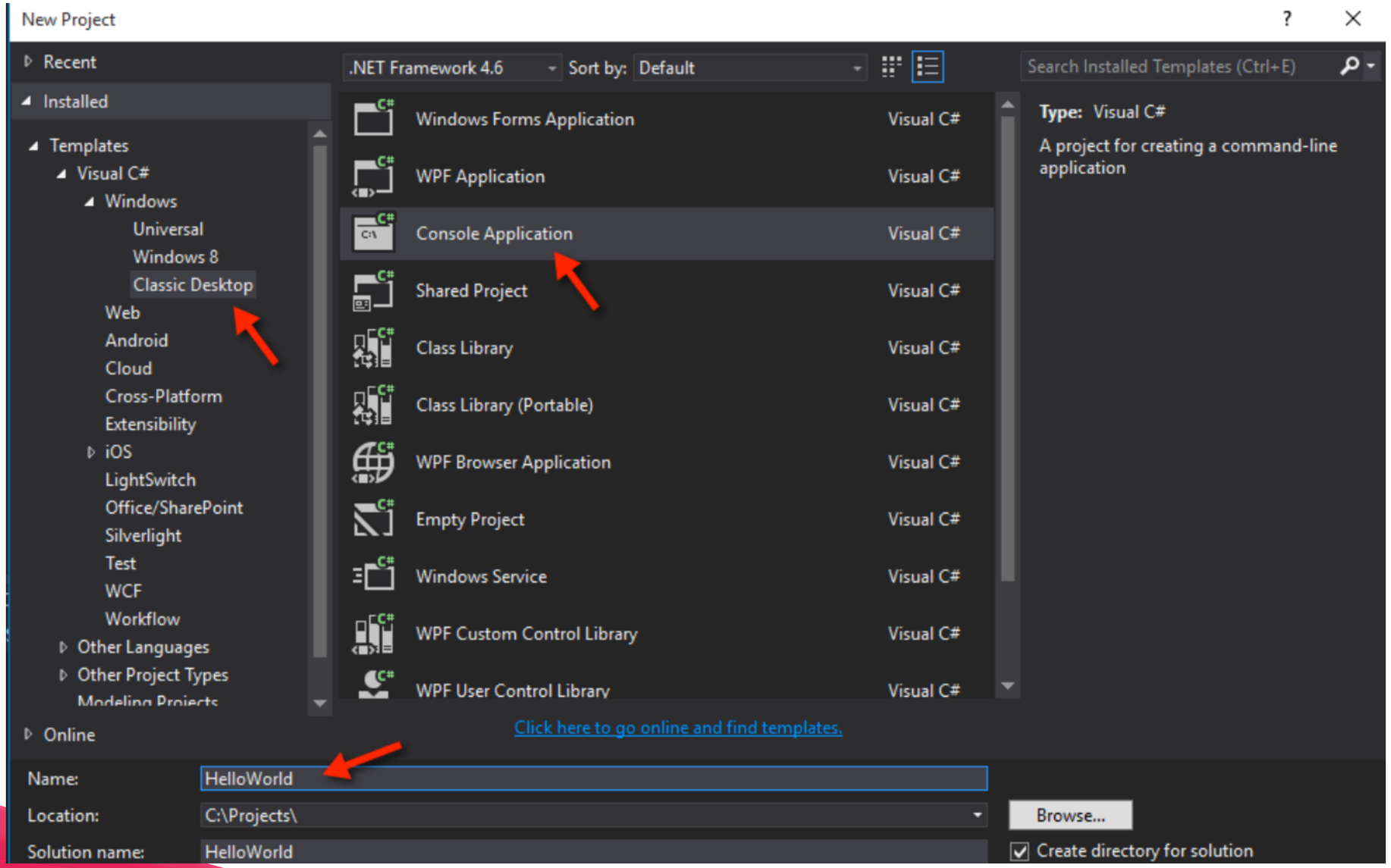
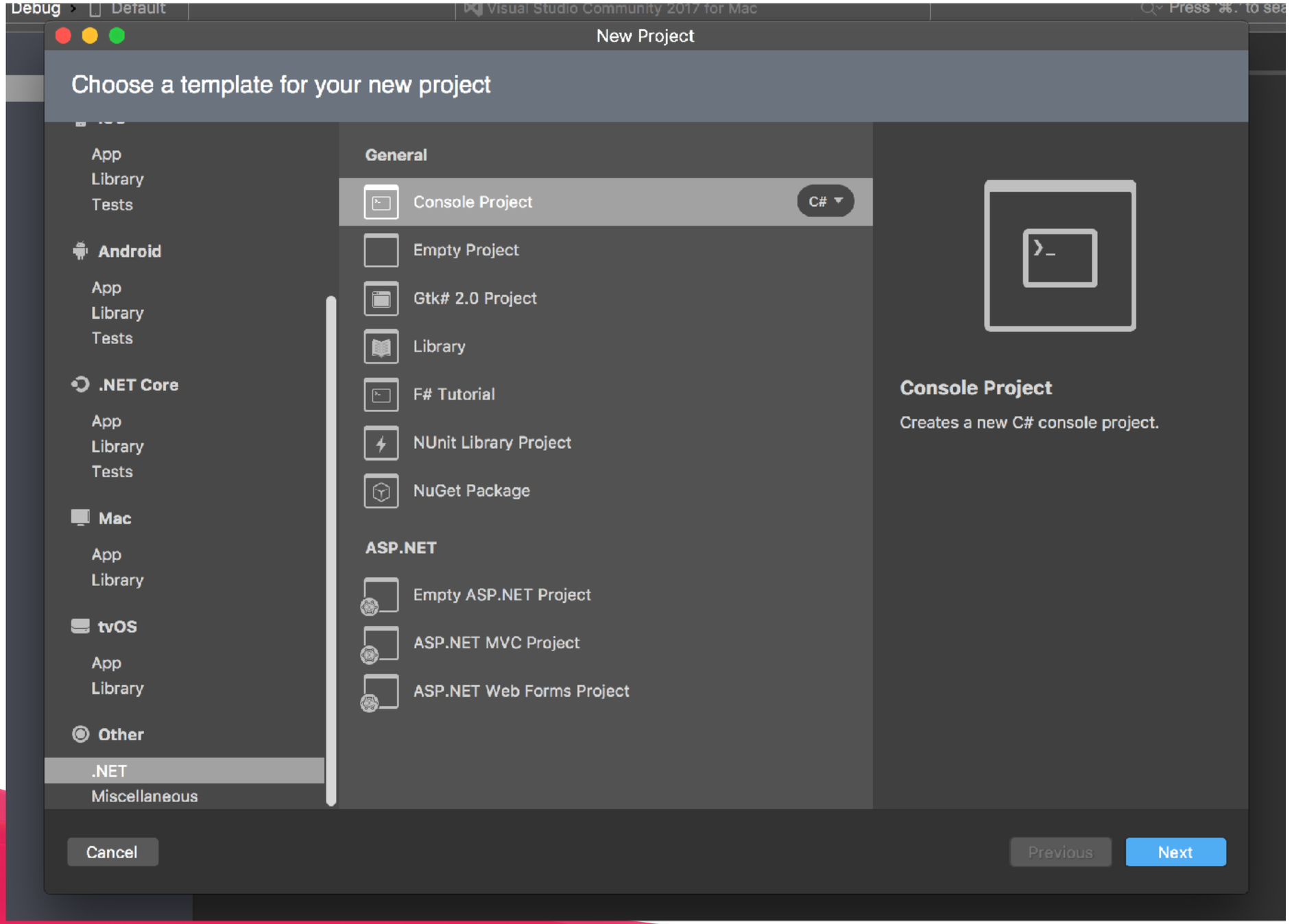
Web 开发 基础班

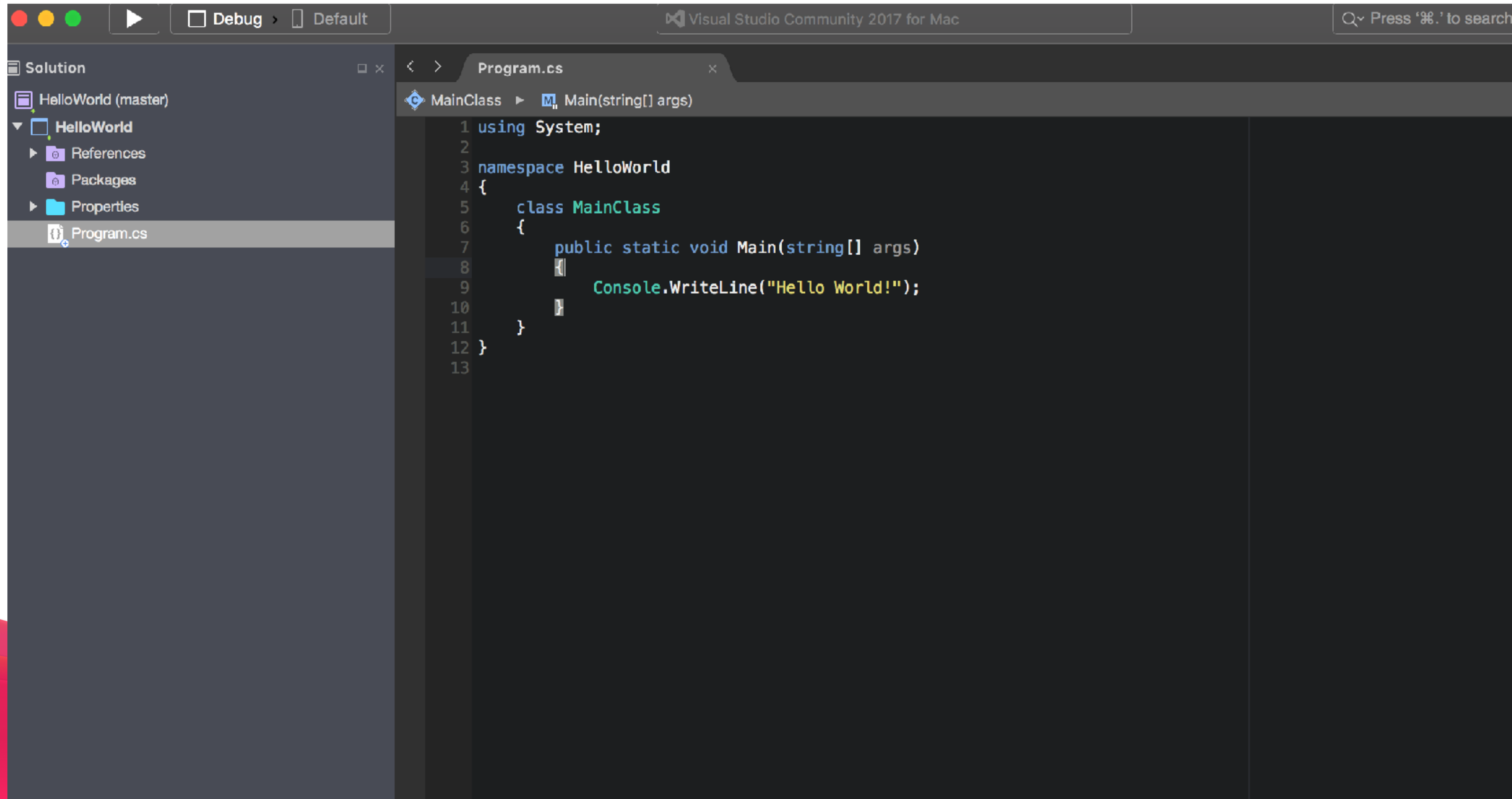
Creating Hello World

In Visual Studio

- Create a new project
- Select Templates > Visual C# > Windows > Classic Desktop
- Select Console Application
- Choose a name and location and click OK


New Project
Recent
Installed
Templates
Visual C#
Windows
Universal
Windows 8
Classic Desktop
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
LightSwitch
Office/SharePoint
Silverlight
Test
WCF
Workflow
Other Languages
Other Project Types
Online
.NET Framework 4.6
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Shared Project Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
WPF User Control Library Visual C#
Type: Visual C#
A project for creating a command-line application
Click here to go online and find templates.
Name: HelloWorld
Location: C:\Projects\
Solution name: HelloWorld
Browse...
Create directory for solution


Debug
Default
Visual Studio Community 2017 for Mac
New Project
Choose a template for your new project
App
Library
Tests
Android
App
Library
Tests
.NET Core
App
Library
Tests
Mac
App
Library
tvOS
App
Library
Other
.NET
Miscellaneous
General
Console Project
C#
Empty Project
Gtk# 2.0 Project
Library
F# Tutorial
NUnit Library Project
NuGet Package
ASP.NET
Empty ASP.NET Project
ASP.NET MVC Project
ASP.NET Web Forms Project
Console Project
Creates a new C# console project.
Cancel
Previous
Next


Debug
Default
Visual Studio Community 2017 for Mac
Press '⌘.' to search
Solution
HelloWorld (master)
HelloWorld
References
Packages
Properties
Program.cs
Program.cs
MainClass
Main(string[] args)
using System;
namespace HelloWorld
{
class MainClass
{
public static void Main(string[] args)
{
Console.WriteLine("Hello World!");
}
}
}

Variables

A variable is a value holder

It has a type (e.g., int)

It has an identifier (e.g., myAge)

It has a value that may change during the running of the program

```
int myAge = 39;
```

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0}$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

if Statements

if (some expression) evaluates to true, do this action...

else: if the if statement is false, then do this...

Notes

= is used for assignmet: int x = 5;

== is used for equality: 5 + 3 == 8 // true

!= is used for inequality: 5 + 3 != 9 // true

Incrementing

```
int x = 5;          // x == 5
x = x + 1;          // x == 6
x += 1;             // x == 7
x++;                // x = 8

int y = ++x;        // y == 9, x == 9
y = x++;            // y == 9, x == 10
```

Decrementing

```
int x = 5;          // x == 5
x = x - 1;          // x == 4
x -= 1;             // x == 3
X--;                // x == 2

int y = --x;        // y == 1, x == 1
y = x--;            // y == 1, x == 0
```

For Loop

Simplest loop is for

for has three parts:

Initialization	for (**int i = 0**
Test	for (int i = 0; **i < 10**
Increment	for (int i = 0; i< 10; **i++**)

Beware of “off by one” or “fencepost” errors!

Constants

Constants are assigned a value that never changes (hence the name!)

Can make your code more understandable

Makes it easier to fix a value used in many places

```
const double pi = 3.141592653589;
const int canDrinkAge = 21;
```

There Are Bugs, and Then There Are Bugs...

Bugs (errors) come in two flavors:

- Compile Time Bugs
- Run Time Bugs

If you are going to have a bug, you *really* want it to be a compile time bug

String and Array

```
String text = “JianRen”;
Console.WriteLine(text.Length) // 7
Console.WriteLine(text[0]); // J
```

```
int[] arrayOfNum = {1,2,3,5}
```

For Loop

```
for (int i = 0; i < 100; i++) {
        Console.WriteLine("This is a test:" +i);
}

string word = "Hello World”;

for (int i = 0; i < word.Length;i++) {
        Console.WriteLine(word[i]);
}
```

While Loop and Do... While Loop

While some condition is true, do the following work

(while)

While Loop

```
int a = 10;
int i = 0;
while(i < a){
    Console.WriteLine("a" + i);
    i++;
}
```

Lab/ Challenge
(or take a rest)

Implement strStr().

Return the index of the first occurrence of needle in haystack, or **-1** if needle is not part of haystack.

Example 1:

```
Input: haystack = "hello", needle = "ll"
Output: 2
```

Example 2:

```
Input: haystack = "aaaaa", needle = "bba"
Output: -1
```

Class

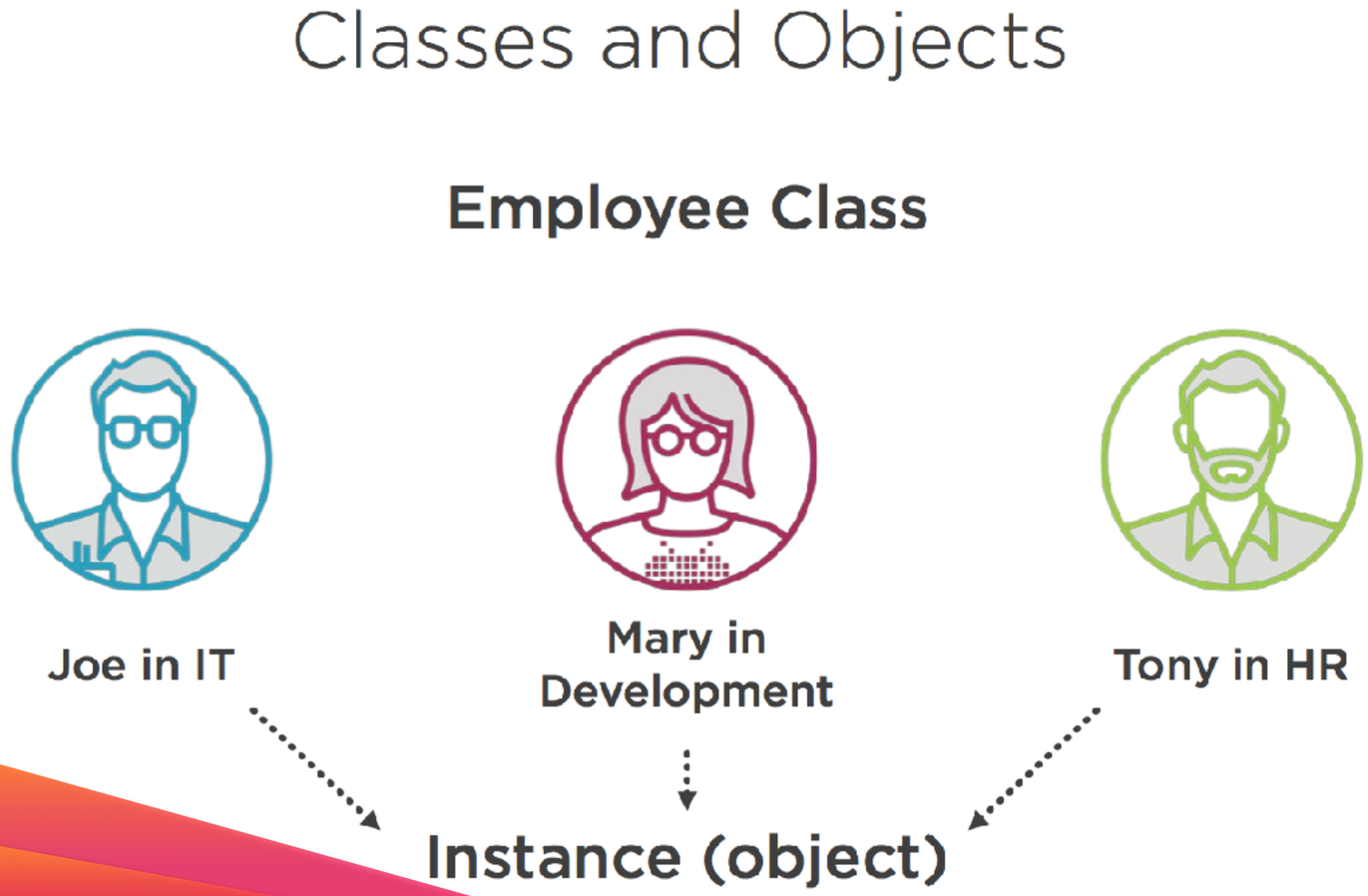
A class is a construct that enables you to create your own custom types by grouping together variables of other types, **methods** and **events**. A class is like a **blueprint**. It defines the data and behavior of a type. If the class is not declared as static, client code can create instances of it. These instances are objects which are assigned to a variable.

Object

An instance of a class is called an object.
Or class is the blueprint of the object.

```
Student student = new Student();
student.age = 18;
student.name = “Tony”;
stuent.class = “Fullstack 101”;
```

Class




```
Student student1 = new Student("Tony", "Fullstack 101", 25);
Student student2 = new Student("Joe", "Mobile Dev 101", 25);
Student student3 = new Student("Mary", "Interview Drill", 25);
```

```
class Student
{
    private string _name;
    private string _course;
    private int _age;
    public Student(string name, string course, int age) {
        this.name = name;
        this.course = course;
        this.age = age;
    }
    public string description(){
        return $"Student :{this.name}  is in class {this.course}";
    }
}
```

Classes Have ...

Fields

Properties

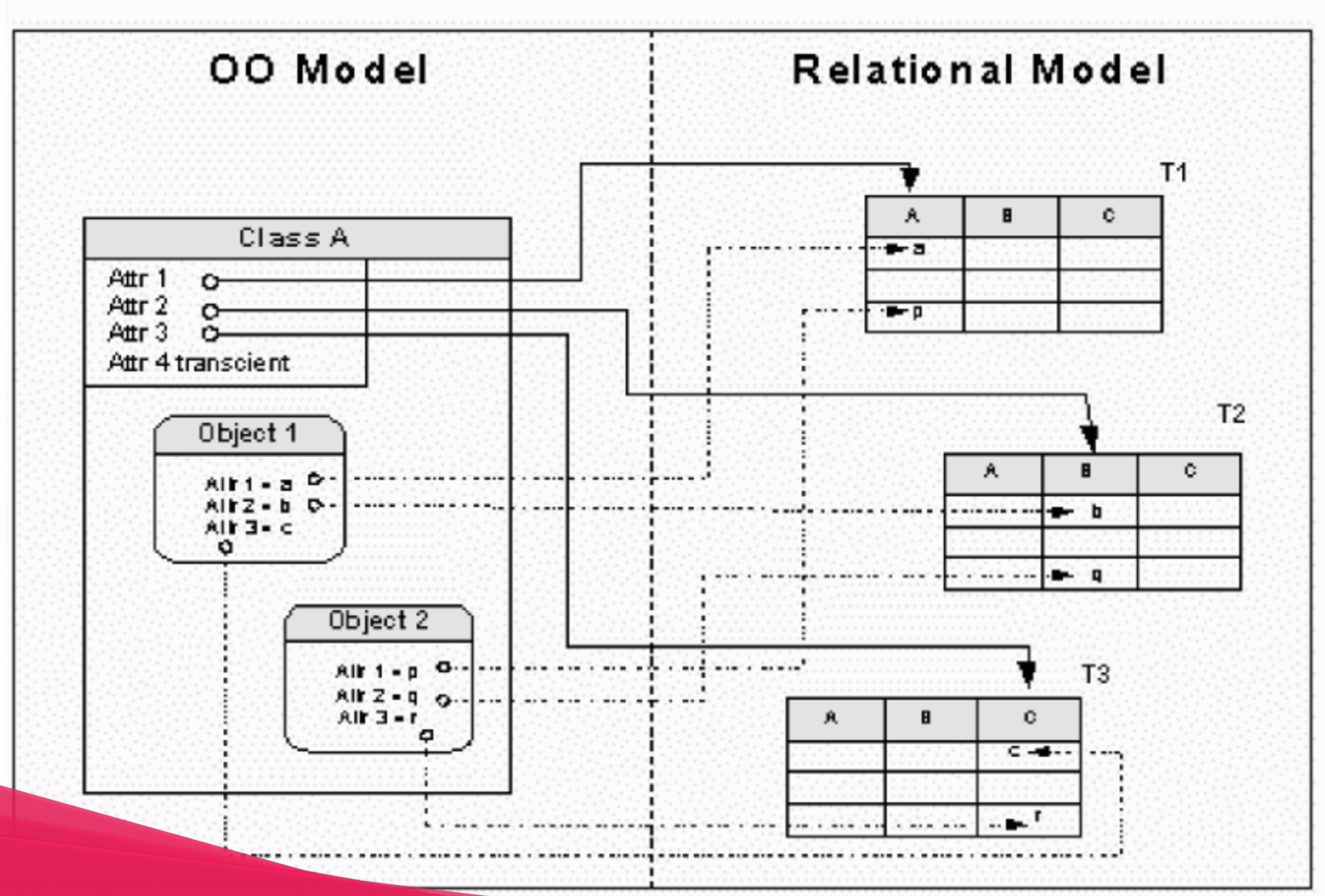
Methods

```
public class MyClass
{
    // this is a field.  It is private to your class and stores the actual data.
    private string _myField;

    // this is a property. When accessed it uses the underlying field,
    public string MyProperty
    {
        get
        {
            return _myField;
        }
        set
        {
            _myField = value;
        }
    }

    public void MyMethod(){
        Console.WriteLine();
    }
}
```

Later Class in team project - Model



Object

- Constructor
- The method to create a new object. Usually set up the property in initial status.

```
public Student(string name, string course, int age) {
        this.name = name;
        this.course = course;
        this.age = age;
}
```

Static Variable/Method vs Object Variable/Method

When you have code that can be shared across all instances of the same class, put that portion of code into static method.

```
class Student {
 public static int StudentCount == 0

 public Student(string name, string course, int age) {
            StudentCount++
            this.name = name;
            this.course = course;
            this.age = age;
  }
}
```

A Word About Capitalization

camelCase vs. PascalCase

Identical except PascalCase begins with an upper case letter, while camelCase begins with a lower case letter

Fields

Variables and parameters are camelCase. (myValue)

Classes

Constants and properties are PascalCase. (MyValue)

Note

This is just a convention, but you violate it at your own risk.

Collection

Must declare size

- Too large: waste space
- Too small, run out of room

Easy to run past the end (crash)

- int[] someNumbersArray = new int[7];

Collection/List

```
List<int> intList = new List<int>() { 10, 20, 30, 40 };

for (int i = 0; i < intList.Count; i++)
        Console.WriteLine(intList[i]);

foreach (var num in intList)
          Console.WriteLine(num);

intList.Add(50);
intList.Add(60);

intList.Remove(10); // removes the 10 from a list
intList.RemoveAt(2);
```

Collection/Dictionary

```
Dictionary<int, string> dict = new Dictionary<int, string>();
dict.Add(1,"One");
dict.Add(2,"Two");
dict.Add(3,"Three");
foreach (KeyValuePair<int, string> item in dict)
{
    Console.WriteLine(item.Key+” : “+ item.Value);
}

for (int i = 0; i < dict.Count; i++)
{
    Console.WriteLine(dict.Keys.ElementAt(i)+” “+dict[ dict.Keys.ElementAt(i)]);
}

dict.ContainsKey(1); // returns true
dict.ContainsKey(4); // returns false
```

Lab 2

Homework/Lab

1. Create a School object which can
    1. Add / Remove course
    2. Enrol the student/ Remove student
        1. With limitation on credit
    3. Print the all student with
        1. Different GPA grade
    4. Generate student fee receipt
        1. 1 course for $100 etc
2. Create a Course object which can
    1. Has maximum number of student
    2. has its own credit
3. Create a Student object which can
    1. show its own credit limit
    2. show GPA

新的社交媒介

微信公共号企业号

jiangren.com.au

facebook.com/jiangrenquan

weibo.com/ozitquan

meetup.com/en-AU/itgroup/