**Tutorial C# Basics 9**

**OOP 2/2 – the four pillars**

V1.0

18/08/2022

This is second part of Object-Oriented Programming. In this tutorial, you will study the four fundamental concepts about OOP. They are also known as the four pillars. By the end of this tutorial, you will be able to

* Understand the four pillars of OOP
* Apply the concepts into coding

**Topics**

[**1.** **The four pillars of Object Orientation** 1](#_Toc111747186)

[**2.** **Encapsulation** 2](#_Toc111747187)

[**Class activity 1** 2](#_Toc111747188)

[**Class activity 2** 3](#_Toc111747189)

[**3.** **Inheritance** 4](#_Toc111747190)

[**Class activity 3** 4](#_Toc111747191)

[**Step 1.** **Define “Person” class** 5](#_Toc111747192)

[**Step 2.** **Define “Teacher” class** 5](#_Toc111747193)

[**Step 3.** **Define “Student” class** 5](#_Toc111747194)

[**Step 4.** **Define “InternationalStudent” class** 5](#_Toc111747195)

[**Step 5.** **Run some test** 5](#_Toc111747196)

[**Challenge** 5](#_Toc111747197)

[**4.** **Polymorphism** 6](#_Toc111747198)

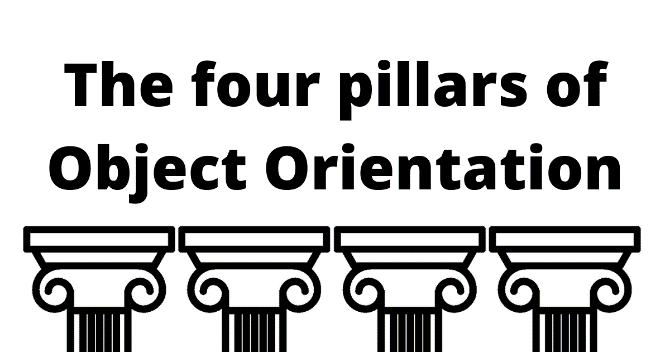
[**5.** **Abstraction** 6](#_Toc111747199)

[**Model answers** 6](#_Toc111747200)

[**Class activity 1** 6](#_Toc111747201)

[**Class activity 2** 7](#_Toc111747202)

1. **The four pillars of Object Orientation**



OOP moves around its four most important pillars. These are: Encapsulation, Inheritance, Polymorphism, and Abstraction.

1. **Encapsulation**

“Encapsulation is a mechanism to bundle the data and code acting on the data together as a single unit.” *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

Encapsulation involves the following concepts discussed in the previous tutorial (OOP part 1)

* Class members – including member **fields** and member **methods**. Member fields are used to store data and member methods are used to perform actions.
* Access modifiers – including “**public**” and “**private**”. Access modifiers are used to control the visibility of class members

Advantages of encapsulation *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

* Loosely coupled code
* Better access control and security

Review the following tutorials on w3schools to understand the concept of encapsulation.

* <https://www.w3schools.com/cs/cs_class_members.php>
* <https://www.w3schools.com/cs/cs_access_modifiers.php>

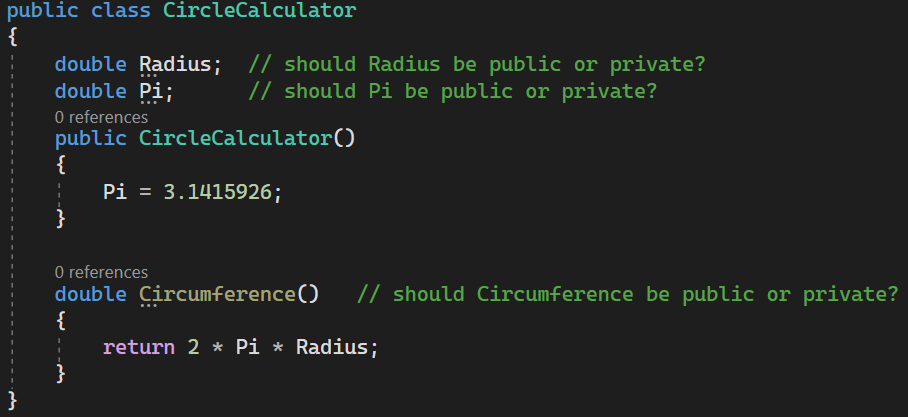
(optional content) C# has a feature called “property”, which is an elegant syntax to achieve "Encapsulation". Study [C# Properties](https://www.w3schools.com/cs/cs_properties.php) tutorial on w3schools for more details.

**Class activity 1**

For the following class “CircleCalculator”, determine

* should Radius be public or private? Why?
* should Pi be public or private? Why?
* should Circumference be public or private? Why?

Code the class in Visual Studio to verify your theory.



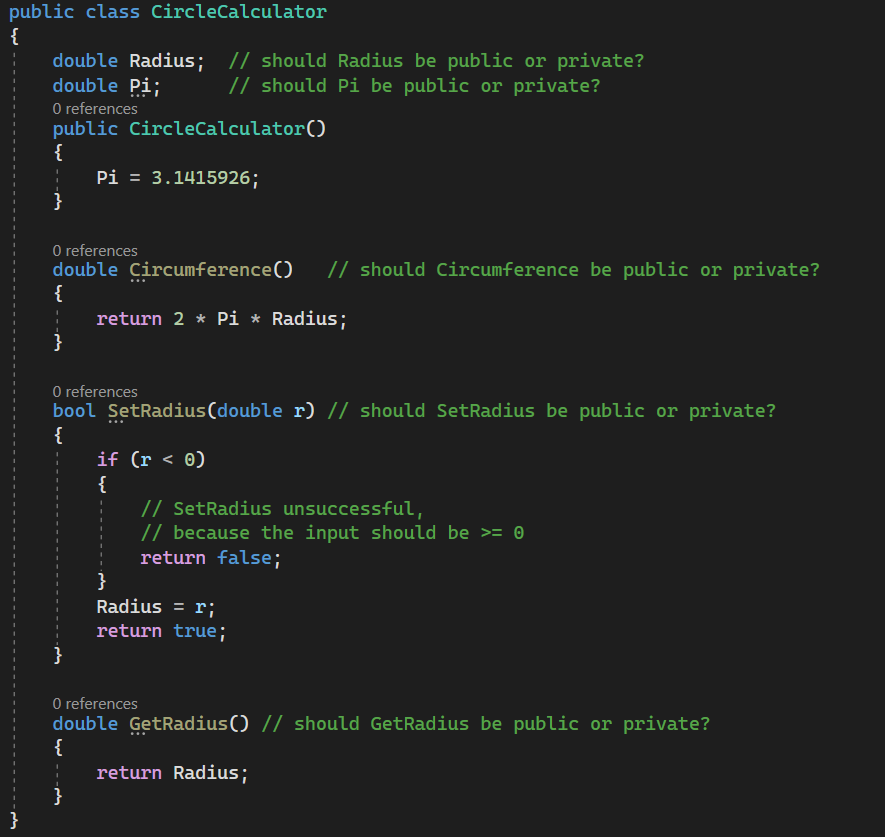
|  |
| --- |
| **Your answers** |

**Class activity 2**

Some enhancements were introduced to the above “CircleCalculator” class. Once again, for each member (field or method), determine if it should be public or private, and why.

Also discuss the enhancements, what are the benefits?

Code the class in Visual Studio to verify your theory.



|  |
| --- |
| **Your answers** |

1. **Inheritance**

“Inheritance is a procedure to inherit the features from parent to child in the real world. Similarly, Inheritance in OOPS is a procedure by which one Class acquires all the properties and behaviors of the parent class. Inheritance ensures the reusability of the code” *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

Study the [C# Inheritance](https://www.w3schools.com/cs/cs_inheritance.php) tutorial on w3schools website. Takeaway notes:

* Inheritance defies the parent-child relationship between classes.
* A class inherits members from its parent, grandparent, grandgrandparent …
* [Access modifiers (private and protected)](https://www.w3schools.com/cs/cs_access_modifiers.php) can be used to control what can be passed to the children and what can’t

Graphical user interface, text, application

Description automatically generated

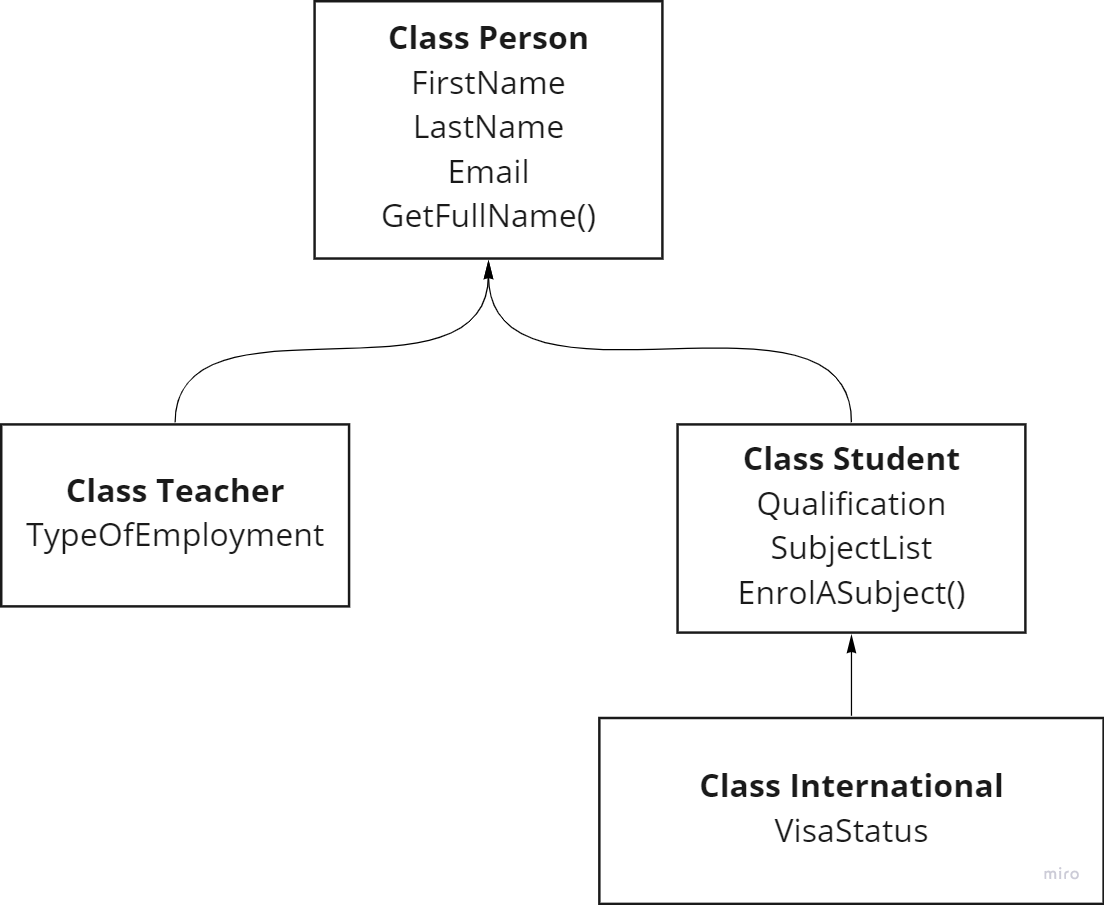
* If you don't want other classes to inherit from a class, use the sealed keyword:

Advantages of Inheritance *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

* Code reusability
* We can achieve Polymorphism using Inheritance.

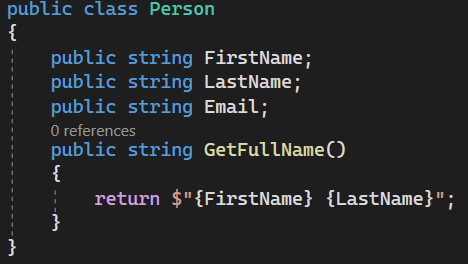
**Class activity 3**

Implement the following class inheritance hierarchy. And create some objects (teacher, student, international student) to verify the implementation

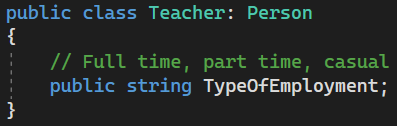


Let’s do it step by step together. You may define all classes in one single .cs file or each class in its own .cs file. In the codes below, constructors are not implemented, you may feel free to add constructors yourself.

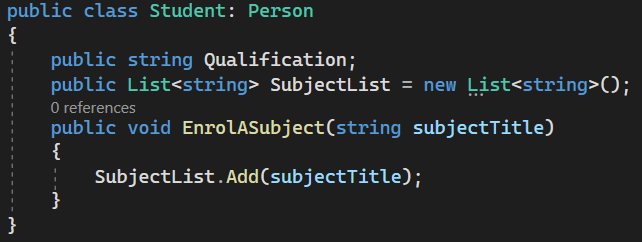
* 1. **Define “Person” class**



* 1. **Define “Teacher” class**



* 1. **Define “Student” class**



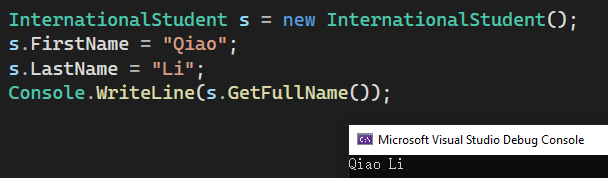
* 1. **Define “InternationalStudent” class**

Text

Description automatically generated

* 1. **Run some test**

Run some test by yourself. Here is an example



**Challenge**

Create an CLI application using the class hierarchy above. The application can

* Add teacher
* Add student
* Enrol student to subjects
* Generate report (list all teachers, list all students, list all international students whose visa expired …)

1. **Polymorphism**

**“Poly** means many, and **morphism** means forms. We know that water also exists in multiple states, such as solid, liquid, and gas. So water shows Polymorphism. In C#, we can achieve Polymorphism using methods.” *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

Study the [C# Polymorphism](https://www.w3schools.com/cs/cs_polymorphism.php) tutorial on w3schools website. Takeaway notes:

* C# provides an option to override the base class method, by adding the **virtual** keyword to the method inside the base class, and by using the **override** keyword for each derived class methods.

1. **Abstraction**

“Abstraction is a property of hiding the internal implementation and highlighting the setup services beneficial to the user. For example, the smartphone user does not know the internal performance of the Smartphone and its workings; instead, they are interested in the services provided by the Smartphone” *quote from* [*this article*](https://www.codingninjas.com/codestudio/library/four-pillars-of-oops-in-java)

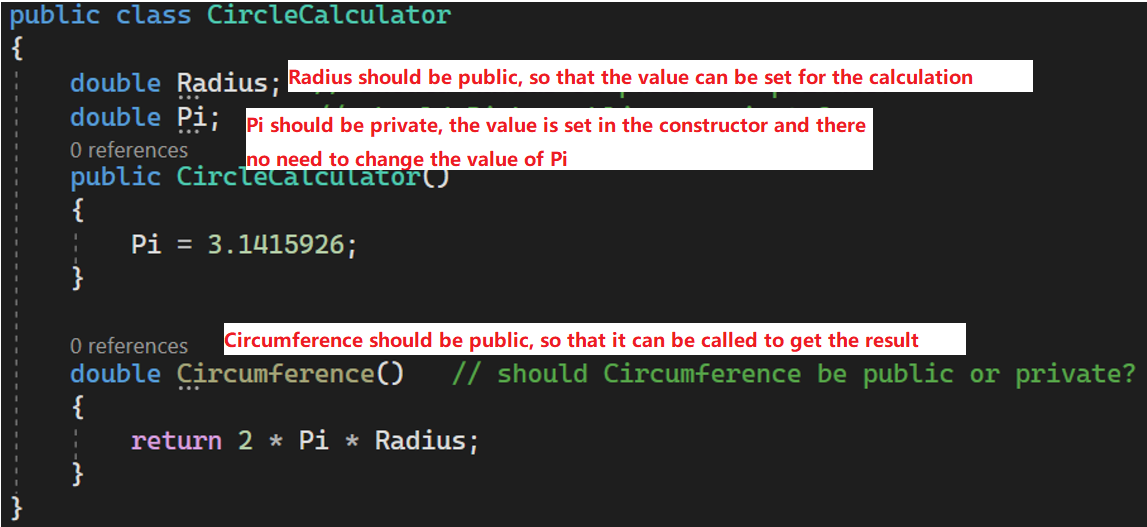
Study the [C# Abstraction](https://www.w3schools.com/cs/cs_abstract.php) tutorial on w3schools website. Takeaway notes:

* Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
  + An abstract class can have both abstract and regular methods
  + A class must be declared to be abstract, if it has one or more virtual method(s).
* Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

(optional content) Pule abstraction can also be achieved with [Interfaces](https://www.w3schools.com/cs/cs_interface.php).

**Model answers**

**Class activity 1**



**Class activity 2**

|  |
| --- |
| public class CircleCalculator  {  // Radius should be private.  // Accessing Radius should go through the SetRadius and GetRadius method  private double Radius;  // Pi should be private, because there is no need to access the value  private double Pi;    public CircleCalculator()  {  Pi = 3.1415926;  }  // Circumference should be public for getting the circumference  public double Circumference()  {  return 2 \* Pi \* Radius;  }  // SetRadius should be public for setting the value of Radius  public bool SetRadius(double r)  {  if (r < 0)  {  // SetRadius unsuccessful,  // because the input should be >= 0  return false;  }  Radius = r;  return true;  }  // SetRadius should be public for getting the value of Radius  public double GetRadius()  {  return Radius;  }  } |

End of this tutorial