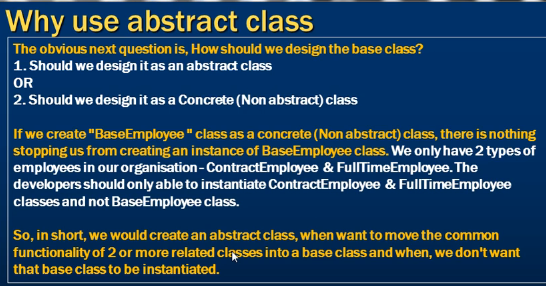
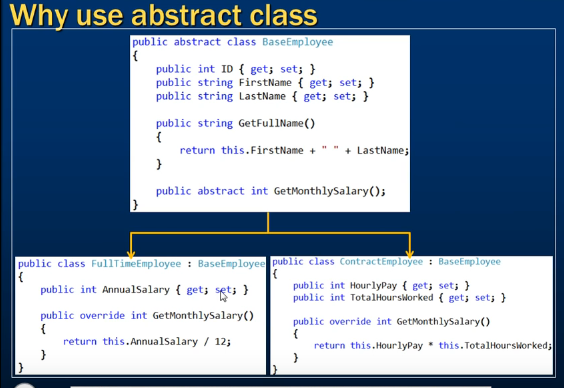
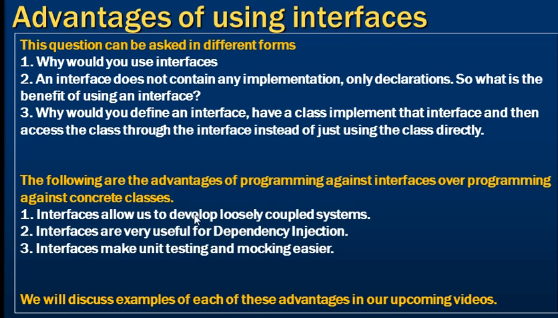
# Abstract class

Common code are duplicated in two classes ContractEmployee and FullTimeEmployee





# Interfaces



# Abstract classes vs Interfaces

Короткое различие.

Абстрактный класс — это класс, у которого не реализован один или больше методов (некоторые языки требуют такие методы помечать специальными ключевыми словами).

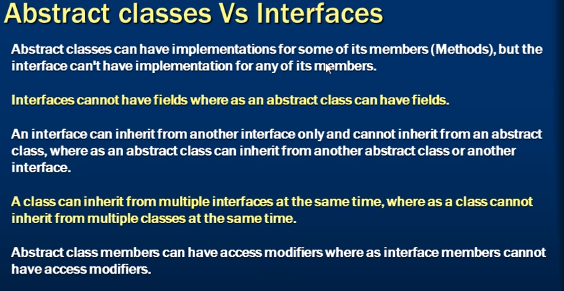
Интерфейс — это абстрактный класс, у которого **ни один** метод не реализован, все они публичные и нет переменных класса.

Интерфейс нужен обычно когда описывается только интерфейс (тавтология). Например, один класс хочет дать другому возможность доступа к некоторым своим методам, но не хочет себя «раскрывать». Поэтому он просто реализует интерфейс.

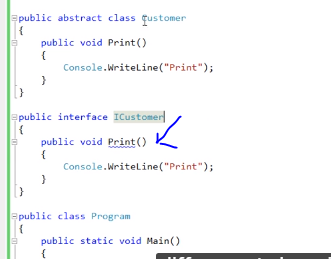
Абстрактный класс нужен, когда нужно семейство классов, у которых есть много общего. Конечно, можно применить и интерфейс, но тогда нужно будет писать много идентичного кода.

В некоторых языках (С++) специального ключевого слова для обозначения интерфейсов нет.

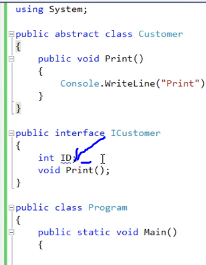
Можно считать, что любой интерфейс — это уже абстрактный класс, но не наоборот.



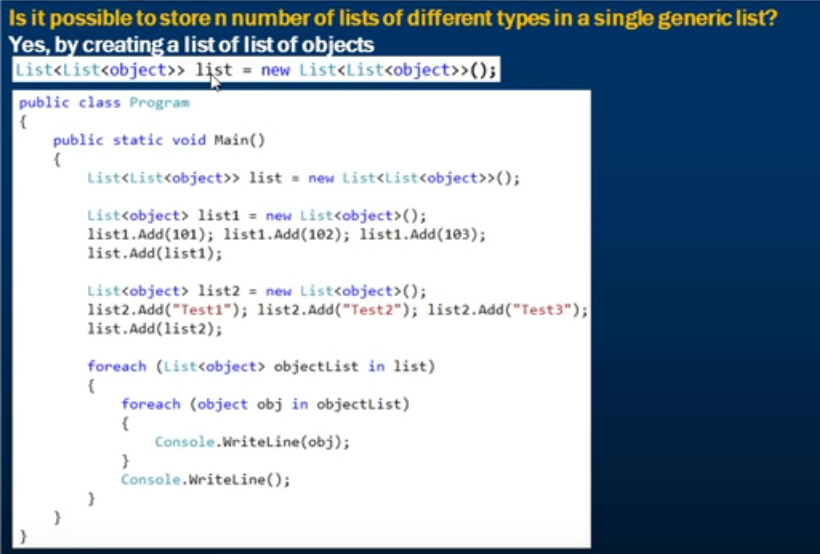
Interfaces cannot have implementation



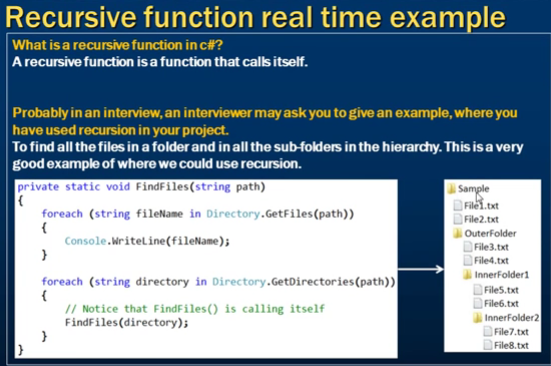
Interfaces cannot have field



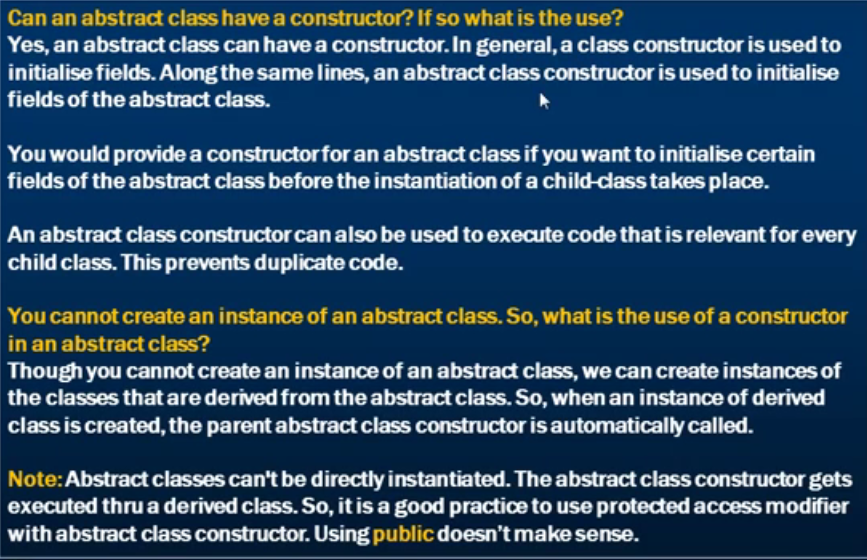
# Storing different types lists in list

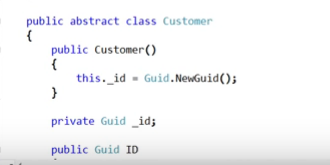


# Recursive

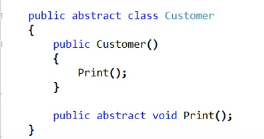


# Abstract class constructor

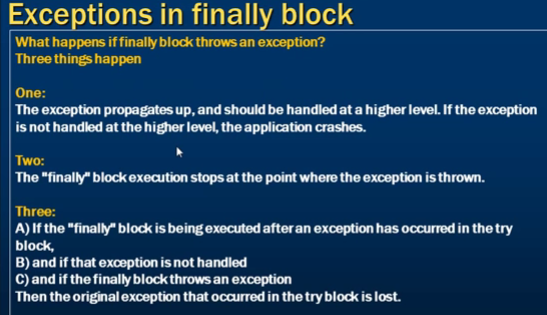




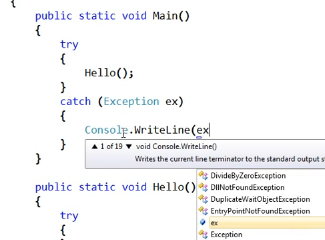
Calling an abstract method from an abstract class



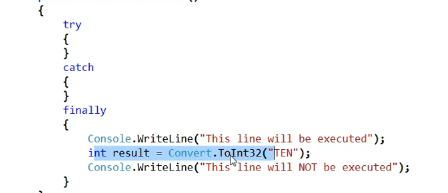
# Finally block exception



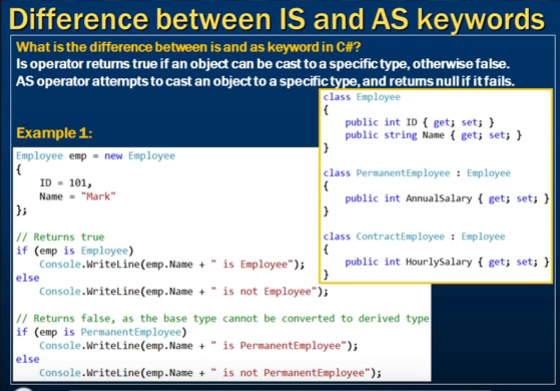
Higher level catching

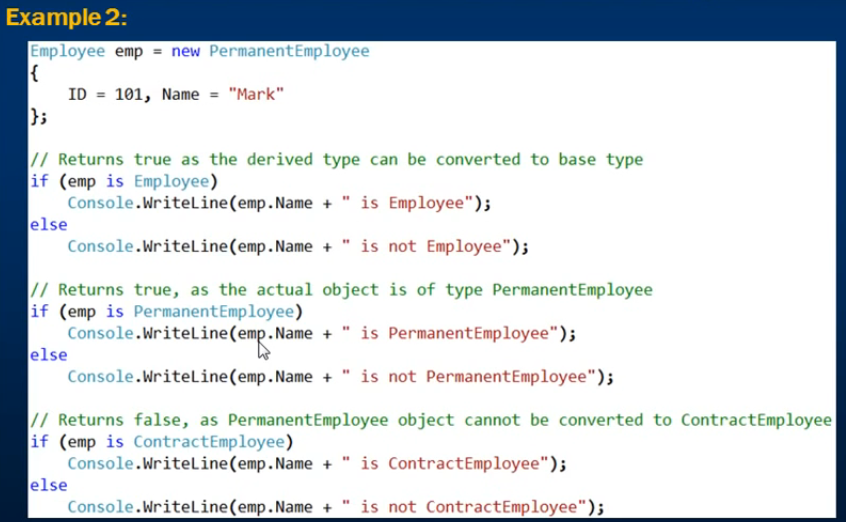


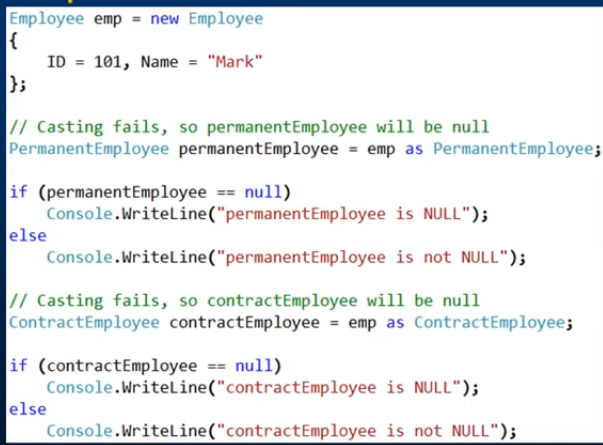
Stop at the point where the exception throw

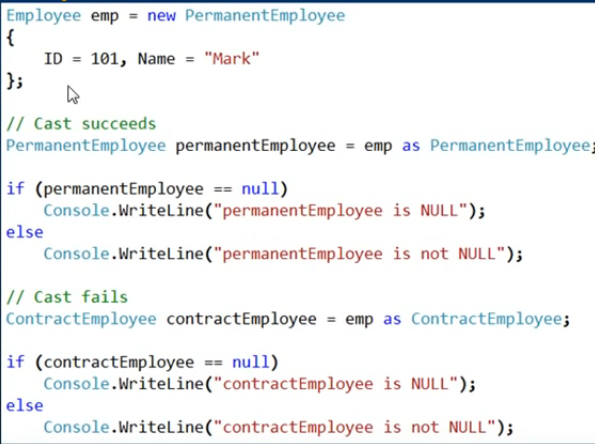


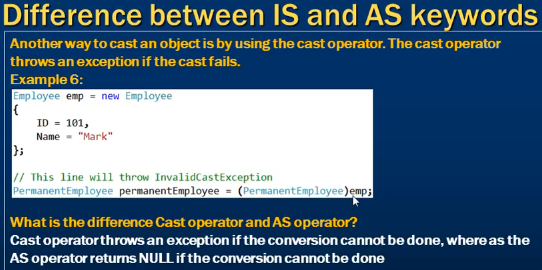
# Is and As differences



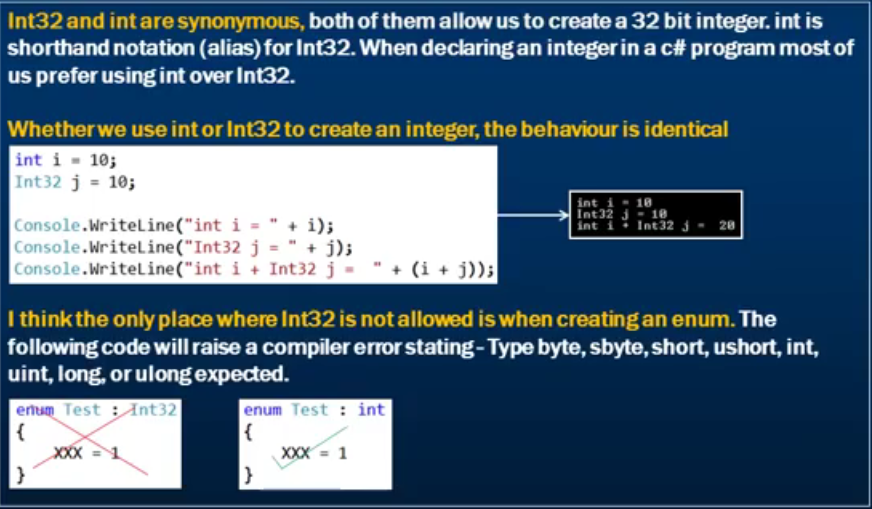




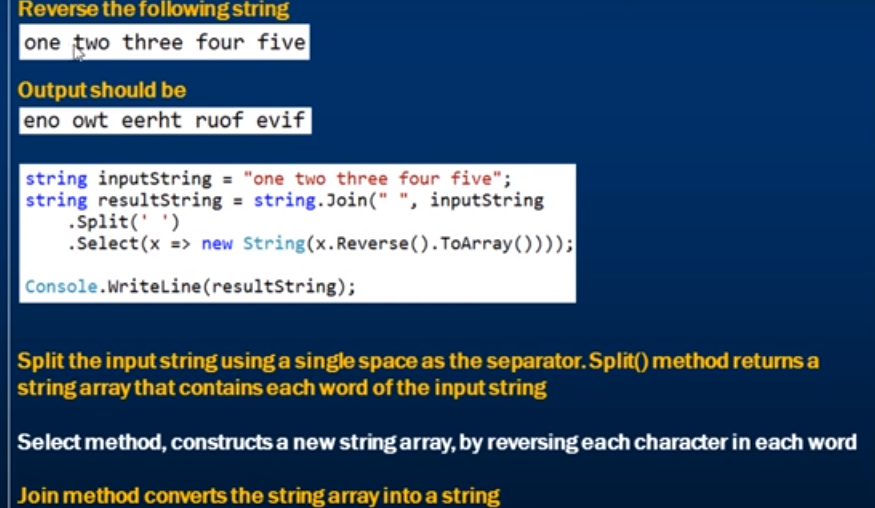




# Differences between Int32 and int



# Reverse each word in string



# Test c# beginer

1. Which of the followings is not allowed in C# as access modifier?

public

friend

internal

protected

 Correct!

2. In the C# code below, what is this[int i]?   
  
class MyClass   
{   
// ...   
  
public string this[int i]   
{   
get{ return arr[i];}   
set{ arr[i] = value; }   
}   
}

Property

Event

Indexer

Delegate

 Correct!

3. Which of the following C# keywords has nothing to do with multithreading?

async

await

sealed

lock

 Correct!

4. Find an invalid expression among the following C# Generics examples.

class A where T : class, new()

class A where T : struct, IComparable

class A where T : class, struct

class A where T : Stream where U : IDisposable

 Correct answer is 3

5. new keyword in C# is used to creat new object from the type. Which of the followings is not allowed to use new keyword?

Class: var a = new Class1();

Interface : var a = new IComparable();

Struct : var a = new Struct1();

C# object : var a = new object();

 Correct!

6. In the example below, button1 is an object of Button class in WinForms. Which one is a wrong expression as a click event handler?

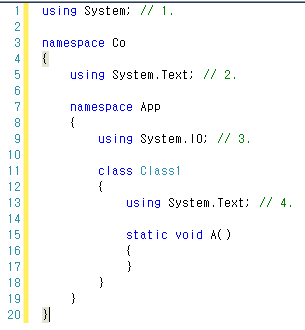
button1.Click += new System.EventHandler(button1\_Click);

button1.Click += delegate { MessageBox.Show("Click"); };

button1.Click += delegate(EventArgs e){MessageBox.Show("Click");};

button1.Click += (s, e) => MessageBox.Show("Click");

 Correct answer is 3

7. In the C# example below, which using statement is wrong?   


1

2

3

4

 Correct!

8. What is the output of this C# code?   
  
int? i = 8 >> 5;   
int? j = i > 0 ? i : null;   
var a = j ?? int.MinValue;   
Console.WriteLine(a);

1

null

0

-2147483648

 Correct!

9. Find a correct statement about C# exception

C# exception occrs at compile time

C# exception occrs at linking time

C# exception occrs at JIT compile time

C# exception occrs at run time

 Correct!

10. Find an invalid Main() method prototype, which is entry point in C#?

public static void Main()

public static int Main()

public static void Main(string[] s)

public static long Main(string[] args)

 Correct answer is 4