### Пример абстрактных классов

public abstract class Shape

{

protected int magicValue;

protected Shape(int val)

{

magicValue = val;

}

public abstract string ShapeType { get; }

public abstract double Area();

}

public class Circle : Shape

{

public double Radius { get; set; }

public Circle(int val)

: base(val)

{

}

public override string ShapeType { get; } = nameof(Circle);

public override double Area() => Math.PI \* Math.Pow(Radius, 2);

}

public class Square : Shape

{

public double Side { get; set; }

public Square(int val)

: base(val)

{

}

public override string ToString()

{

return base.magicValue.ToString();

}

public override string ShapeType { get; } = nameof(Square);

public override double Area() => Math.Pow(Side, 2);

}

public class Triangle : Shape

{

public double Base { get; set; }

public double Height { get; set; }

public Triangle(int val)

: base(val)

{

}

public override string ShapeType { get; } = nameof(Triangle);

public override double Area() => (Height \* Base) / 2;

}

class Program

{

static void Main(string[] args)

{

var circle = new Circle(1) { Radius = 5 };

var square = new Square(2) { Side = 2 };

var triangle = new Triangle(3) { Base = 7, Height = 4 };

Console.WriteLine($"Площадь: {circle.Area()}, Тип: {circle.ShapeType}");

Console.WriteLine($"Площадь: {square.Area()}, Тип: {square.ShapeType}");

Console.WriteLine($"Площадь: {triangle.Area()}, Тип: {triangle.ShapeType}");

Console.WriteLine(square.ToString());

Console.WriteLine(triangle.ToString());

Console.ReadKey();

}

}

### Интерфейсы

class Test

{

static void Main()

{

SampleClass sc = new SampleClass();

ISurface srfc = (ISurface)sc;

sc.Paint();

srfc.Paint();

}

}

interface ISurface

{

void Paint();

}

class SampleClass : ISurface

{

public void Paint()

{

Console.WriteLine("Paint method in SampleClass");

}

}

### Явная реализация интерфейса

class Program

{

interface IControl

{

void Paint();

}

interface ISurface

{

void Paint();

}

public class SampleClass : IControl, ISurface

{

void IControl.Paint()

{

System.Console.WriteLine("IControl.Paint");

}

void ISurface.Paint()

{

System.Console.WriteLine("ISurface.Paint");

}

}

static void Main()

{

SampleClass sc = new SampleClass();

IControl ctrl = (IControl)sc;

ISurface srfc = (ISurface)sc;

sc.Paint();

ctrl.Paint();

srfc.Paint();

}

}

### Пример, животные

//Файл ISniff

public interface ISniff

{

string Sniff();

}

//Файл IRunable

public interface IRunable

{

string Run();

}

//Файл IFlyable

public interface IFlyable

{

string Fly();

}

//Файл AbstractAnimal

public abstract class AbstractAnimal : IDisposable

{

private int legsCount;

private string animalName;

private AnimalType animalType;

public string Name

{

get { return this.animalName; }

}

public AnimalType Type

{

get { return this.animalType; }

}

public int LegsCount

{

get { return this.legsCount; }

}

protected AbstractAnimal(string name, int legsCount, AnimalType animalType)

{

animalName = name;

this.legsCount = legsCount;

this.animalType = animalType;

}

//Заглушка требующая реализации в наследниках

public abstract string Sound();

public abstract void Dispose();

}

//Файл AnimalType

public enum AnimalType

{

Mammals = 10,

Birds

}

public class Dog : AbstractAnimal, IRunable, ISniff

{

public Dog(string name) : base(name, 4, AnimalType.Mammals)

{

}

public override string Sound()

{

return "Bow-vow";

}

public override void Dispose()

{

throw new NotImplementedException();

}

public string Run()

{

return "I am running";

}

public string Sniff()

{

return "I am sniffing";

}

}

public class Cow : AbstractAnimal, IRunable

{

public Cow(string name) : base(name, 4, AnimalType.Mammals)

{

}

public override string Sound()

{

return "Moo";

}

public override void Dispose()

{

throw new NotImplementedException();

}

public string Run()

{

return "I am running";

}

}

public class Duck : AbstractAnimal, IFlyable

{

public Duck(string name) : base(name, 2, AnimalType.Birds)

{

}

public string Fly()

{

return "I am flying";

}

public override void Dispose()

{

throw new NotImplementedException();

}

public override string Sound()

{

return "Quack-quack";

}

}

static void Main(string[] args)

{

Dog dog = new Dog("Goofy");

Cow cow = new Cow("Maggie");

Duck duck = new Duck("McQuack");

Console.WriteLine(dog.Sound());

Console.WriteLine(dog.Sniff());

Console.WriteLine(cow.Sound());

Console.WriteLine(duck.Sound());

Console.WriteLine(duck.Fly());

ISniff dogsniff = new Dog("Goofy");

dogsniff.Sniff();

}

### Необобщенные коллекции

public class Zoo

{

public Zoo()

{

\_list = new ArrayList();

}

public ArrayList list { get; set;}

}

class Programm

{

static void Main(string[] args)

{

Dog dog = new Dog("Goofy");

Cow cow = new Cow("Maggie");

Duck duck = new Duck("McQuack");

Console.WriteLine(dog.Sound());

Console.WriteLine(dog.Sniff());

Console.WriteLine(cow.Sound());

Console.WriteLine(duck.Sound());

Console.WriteLine(duck.Fly());

ISniff dogsniff = new Dog("Goofy");

dogsniff.Sniff();

Zoo zoo = new Zoo();

zoo.list = new ArrayList() { new Dog("Goofy"), new Cow("Maggie"), new Duck("McQuack") };

var animal = zoo.list[0];

Console.WriteLine((animal as AbstractAnimal).Sound());

foreach(AbstractAnimal aa in zoo.list)

{

Console.WriteLine(aa.Name);

}

Console.ReadKey();

}

}

### Обобщенная коллекция

public class Zoo : IEnumerable<AbstractAnimal>

{

private List<AbstractAnimal> \_list;

public Zoo()

{

\_list = new List<AbstractAnimal>();

}

//Метод для обращения по индексу

public AbstractAnimal this[int index]

{

get

{

if (index >= \_list.Count)

return null;

return \_list[index];

}

}

public void Add(AbstractAnimal animal)

{

\_list.Add(animal);

}

public string GetAnimals()

{

return String.Join(", ", \_list.Select(x => x.Name));

}

//При реализации IEnumerable(T),

// необходимо реализовать IEnumerable and IEnumerator(T).

public IEnumerator<AbstractAnimal> GetEnumerator()

{

return \_list.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

}

static void Main(string[] args)

{

Dog dog = new Dog("Goofy");

Cow cow = new Cow("Maggie");

Duck duck = new Duck("McQuack");

Console.WriteLine(dog.Sound());

Console.WriteLine(dog.Sniff());

Console.WriteLine(cow.Sound());

Console.WriteLine(duck.Sound());

Console.WriteLine(duck.Fly());

ISniff dogsniff = new Dog("Goofy");

dogsniff.Sniff();

Zoo zoo = new Zoo();

zoo.Add(new Dog("Goofy"));

zoo.Add(new Cow("Maggie"));

zoo.Add(new Duck("McQuack"));

Console.WriteLine(zoo.GetAnimals());

var animal = zoo[0];

Console.WriteLine(animal.Sound());

Console.ReadKey();

}

### IEnumerable

class MyInt : IEnumerable, IEnumerator

{

int[] ints = { 12, 13, 1, 4 };

int index = -1;

// Реализуем интерфейс IEnumerable

public IEnumerator GetEnumerator()

{

return this;

}

// Реализуем интерфейс IEnumerator

public bool MoveNext()

{

if (index == ints.Length - 1)

{

Reset();

return false;

}

index++;

return true;

}

public void Reset()

{

index = -1;

}

public object Current

{

get

{

return ints[index];

}

}

}

class Program

{

static void Main()

{

MyInt mi = new MyInt();

foreach (int i in mi)

Console.Write(i + "\n");

Console.ReadLine();

}

}

### Общие интерфейсы

### IComparable, ICloneable

public class Person : IComparable, ICloneable

{

public string Name { get; set; }

public int Age { get; set; }

public int CompareTo(object o)

{

Person p = o as Person;

if (p != null)

return this.Age.CompareTo(p.Age);

else

throw new Exception("Невозможно сравнить два объекта");

}

public object Clone()

{

return this.MemberwiseClone(); //неглубокое копирование

}

}

class Programm

{

static void Main(string[] args)

{

Person p1 = new Person { Name = "Сергей ", Age = 34 };

Person p2 = new Person { Name = "Александр", Age = 23 };

Person p3 = new Person { Name = "Елена", Age = 21 };

Person[] people = new Person[] { p1, p2, p3 };

Array.Sort(people);

foreach(Person p in people)

{

Console.WriteLine("{0} - {1}", p.Name, p.Age);

}

Person p4 = (Person)p1.Clone();

p4.Name = "Света";

Console.WriteLine(p1.Name); // Сергей

Console.Read();

}

}

### IDisposable

public interface IDisposable

{

void Dispose ();

}

public class SomeClass : IDisposable

{

private bool disposed = false;

// реализация интерфейса IDisposable.

public void Dispose()

{

Dispose(true);

// подавляем финализацию

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!disposed)

{

if (disposing)

{

// Освобождаем управляемые ресурсы

}

// освобождаем неуправляемые объекты

disposed = true;

}

}

// Деструктор

~SomeClass()

{

Dispose(false);

}

}

class Program

{

static void Main(string[] args)

{

SomeClass p = new SomeClass();

//p.Dispose();

}

}

### Исключительная ситуация

static void Main()

{

int a = 98, b = 0;

int result = 0;

try

{

result = a/b;

Console.WriteLine("{0} divided by {1} = {2}", a, b, result);

}

catch (DivideByZeroException e)

{

Console.WriteLine("Attempted divide by zero.");

}

}

### CustomException

class CustomException : Exception

{

public CustomException(string message)

{

}

}

private static void TestThrow()

{

CustomException ex =

new CustomException("Custom exception in TestThrow()");

throw ex;

}

static void TestCatch()

{

try

{

TestThrow();

}

catch (CustomException ex)

{

System.Console.WriteLine(ex.ToString());

}

}

static void Main()

{

TestCatch();

}