//耦合性 -- 降低 高内聚 低耦合

static void Main(string[] args)

{

#region

/\*

ArrayList list = new ArrayList();

Console.WriteLine(list);

Type t = Type.GetType("Lesson13反射.Animal");//类的全称

// t 它是所有反射对象的父类

// Console.WriteLine( (t.GetConstructors().Length));

//int a = t.GetMethods().Length;

// Console.WriteLine();

MethodInfo[] ms = t.GetMethods();

for (int i = 0; i < ms.Length; i++)

{

//Console.WriteLine(ms[i].Name);

}

Console.WriteLine( ms[4].ReturnType.Name);

ParameterInfo[] ps = ms[4].GetParameters();

for (int i = 0; i < ps.Length; i++)

{

Console.WriteLine(ps[i]);

}

//属性的信息，构造器的信息

// 普通方法的调用依赖于对象 ，，反射调用方法，也需要对象

// 那么怎么通过反射来产生一个对象呢？？

//从而去调用某个方法

\*/

#endregion

Type t = Type.GetType("Lesson13反射.Animal");

ConstructorInfo[] cis = t.GetConstructors();

// Console.WriteLine(cis.Length);

//int a = cis[1].GetParameters().Length;

// Console.WriteLine(a);

cis[1].Invoke(new object[] {1,"zz" });

// new Animal();

}

}

public class Animal

{

// 2 个方法？？？ 10个？？

public int Id { get; set; }

public string Name { get; set; }

public int Sleep(int h,string a)

{

Console.WriteLine("呼呼大睡"+h+"个小时");

return 1;

}

public void Eat()

{

Console.WriteLine("狼吞虎咽");

}

public Animal()

{

Console.WriteLine("无参构造器被调用");

}

public Animal(int id,string name)

{

this.Id = id;

this.Name = name;

Console.WriteLine("有参构造器被调用");

}

}