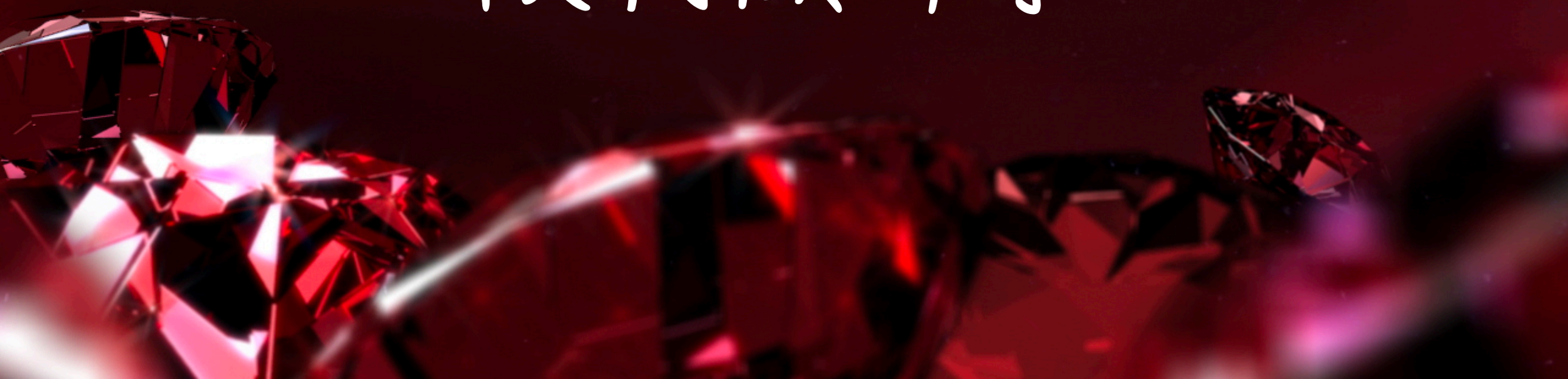
抽象

杨亮

FORBIDDEN

限制做坏事

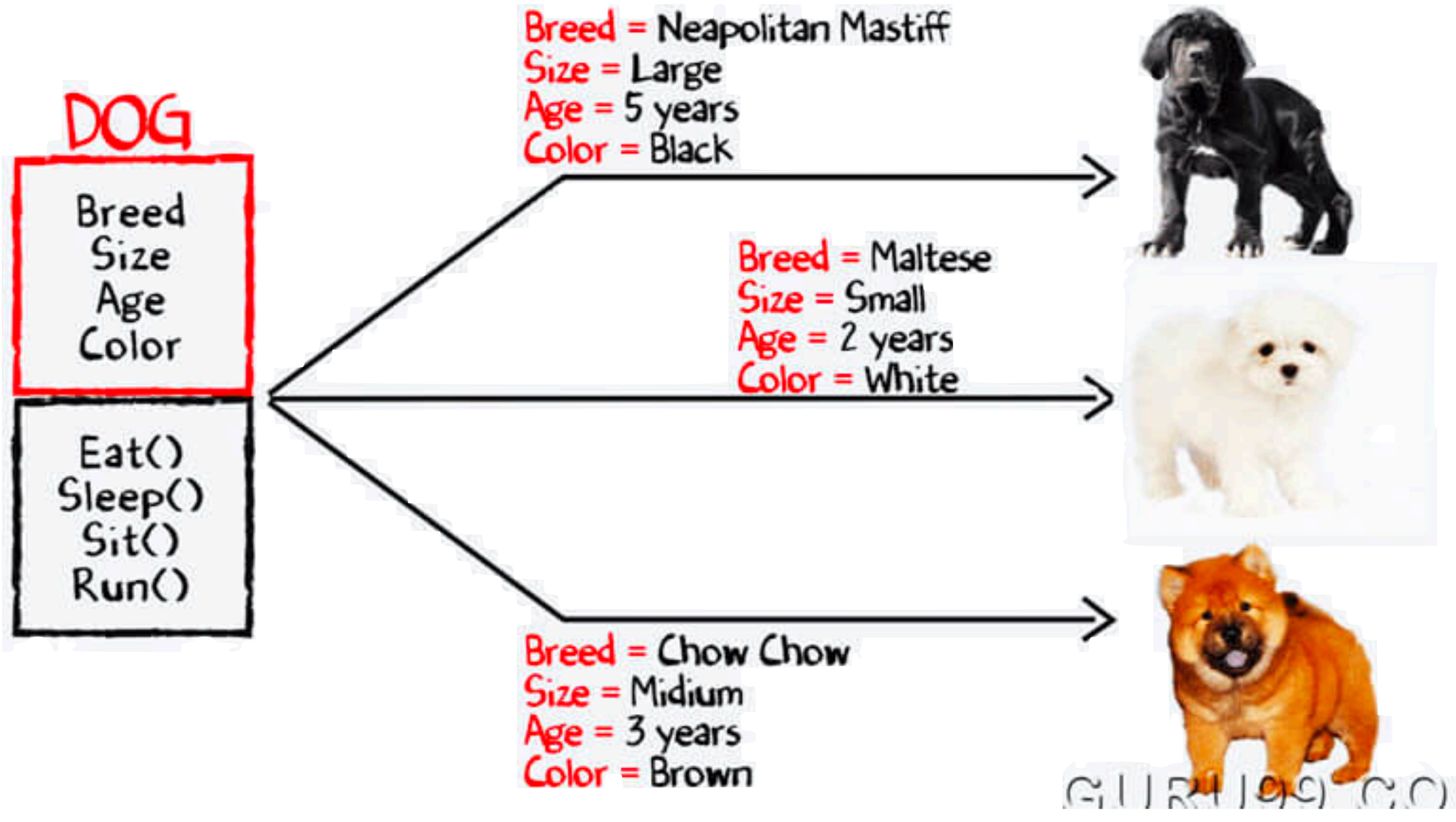


抽象 (Abstraction)

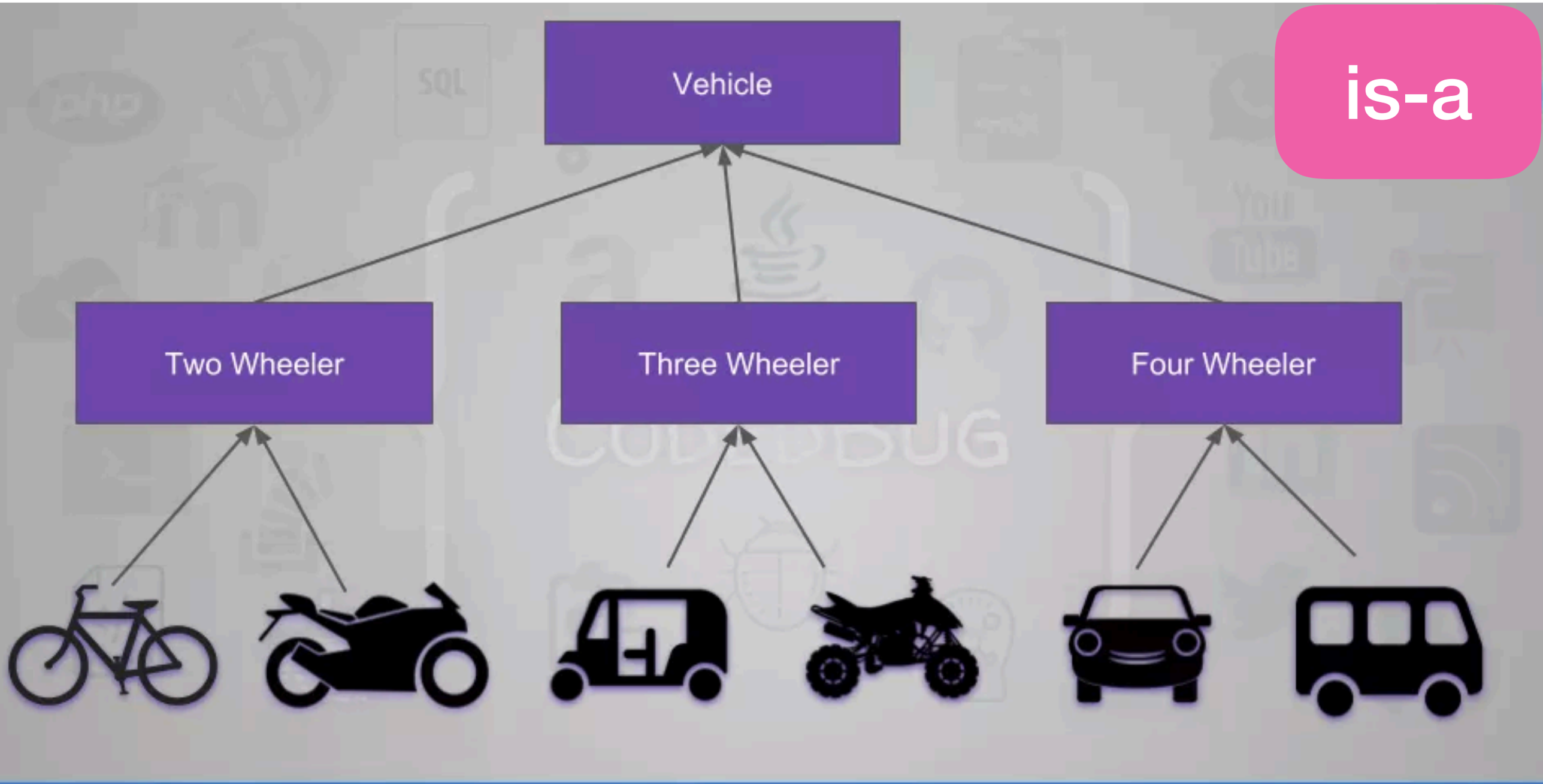
抽象特征与行为
类

共同特征与行为
对象

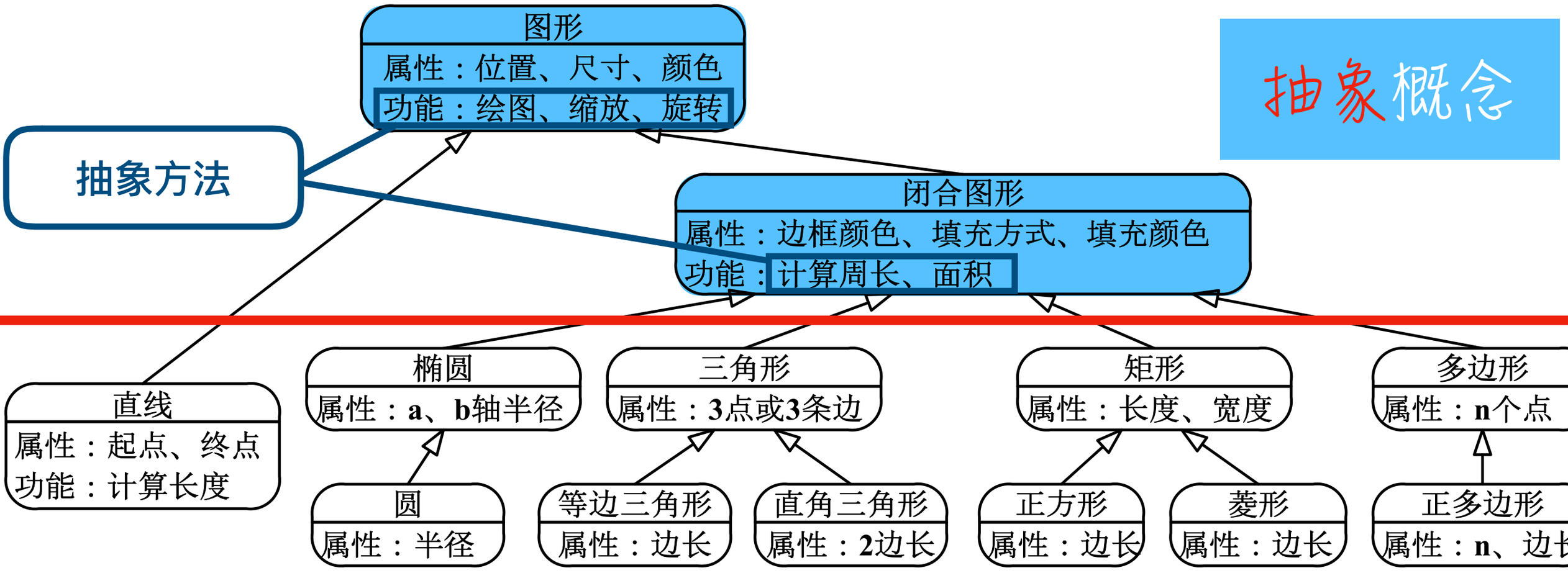
真实事物



继承是进一步的抽象


Vehicle
Two Wheeler
Three Wheeler
Four Wheeler
is-a
Inheritance In Java

更进一步抽象


图形
属性：位置、尺寸、颜色
功能：绘图、缩放、旋转
抽象方法
闭合图形
属性：边框颜色、填充方式、填充颜色
功能：计算周长、面积
抽象概念
直线
属性：起点、终点
功能：计算长度
椭圆
属性：a、b轴半径
三角形
属性：3点或3条边
矩形
属性：长度、宽度
多边形
属性：n个点
圆
属性：半径
等边三角形
属性：边长
直角三角形
属性：2边长
正方形
属性：边长
菱形
属性：边长
正多边形
属性：n、边长
具体概念

成员方法

[修饰符] 返回值类型 方法名([参数列表])

```
{
    语句序列;
    [return [返回值]];
}
```

方法实现

抽象成员方法

[修饰符] **abstract** 返回值类型 方法名([参数列表]);

构造方法、静态成员方法不能被声明为抽象方法。

包含抽象方法的类叫做抽象类

防止被实例化

[修饰符] **abstract** class 类名

```
{
    成员变量的声明;
    成员方法的声明及实现;
}
```

抽象方法没有实现!!!

抽象类不能被实例化!!!

抽象概念

抽象方法

图形
属性：位置、尺寸、颜色
功能：绘图、缩放、旋转

闭合图形
属性：边框颜色、填充方式、填充颜色
功能：计算周长、面积

直线
属性：起点、终点
功能：计算长度

椭圆
属性：a、b轴半径

圆
属性：半径

三角形
属性：3点或3条边

等边三角形
属性：边长

直角三角形
属性：2边长

矩形
属性：长度、宽度

正方形
属性：边长

菱形
属性：边长

多边形
属性：n个点

正多边形
属性：n、边长

一个非抽象类必须实现从父类继承来的所有抽象方法

具体概念

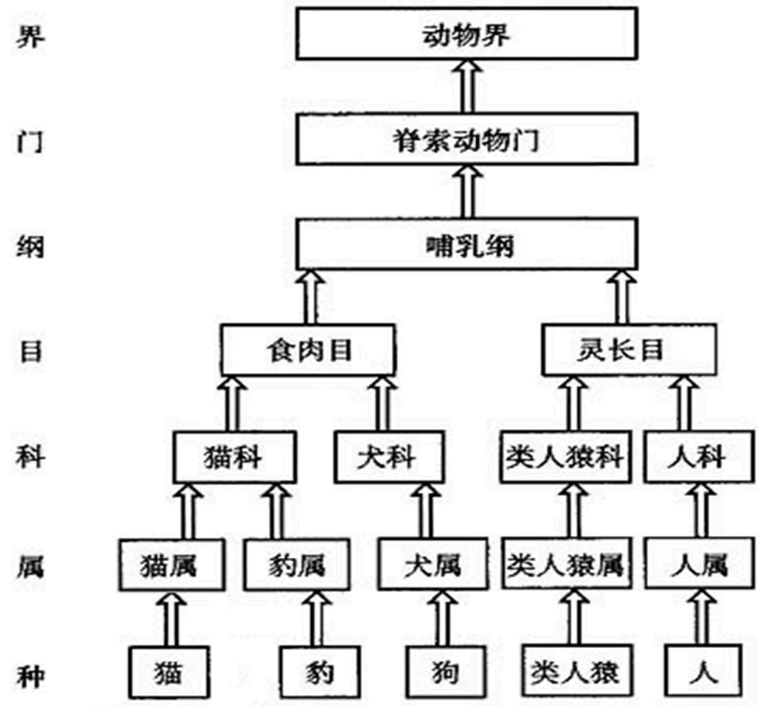
多重继承

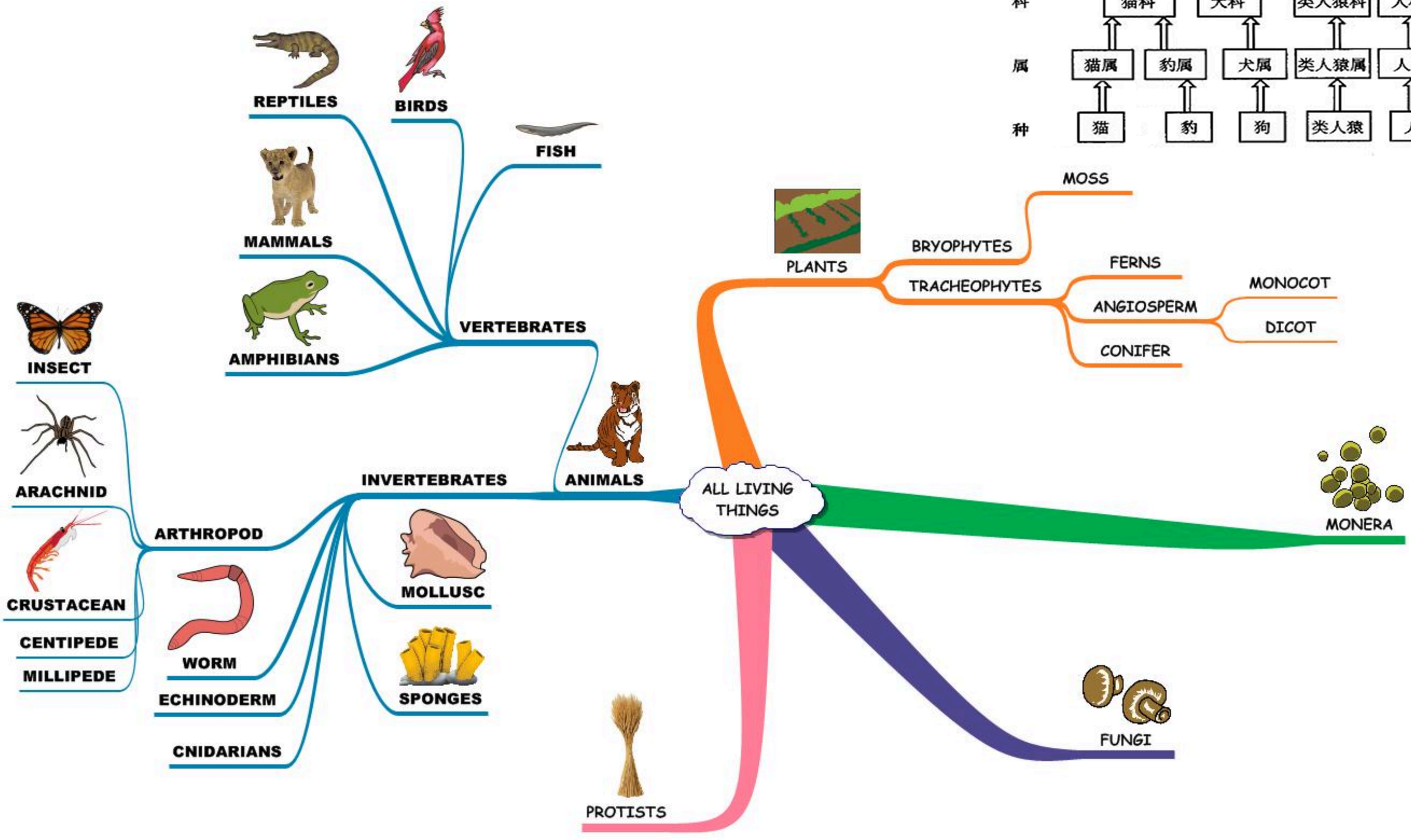
Java不支持多重继承

```
[修饰符]  class  类名  [extends  父类列表]
{
    新增成员变量的声明;
    新增或重写成员方法的声明及实现;
}
```

容易导致不好的设计

Java只支持单一继承





行为的抽象-接口

like-a

继承

跑

飞

叫

行为的抽象

接口 interface

[public] interface 接口 **[extends** 父接口列表**]**
{
    [public] [static] [final] 数据类型 成员变量=常量值;
    [public] [abstract] 返回值类型 成员方法**[(**参数列表**)]**;
}

接口定义

[修饰符] class 类名 [extends 父类名] **[implements 接口列表]**
{
    新增成员变量的声明;
    新增或重写成员方法的声明及实现;
    实现接口列表中的所有成员方法;
}

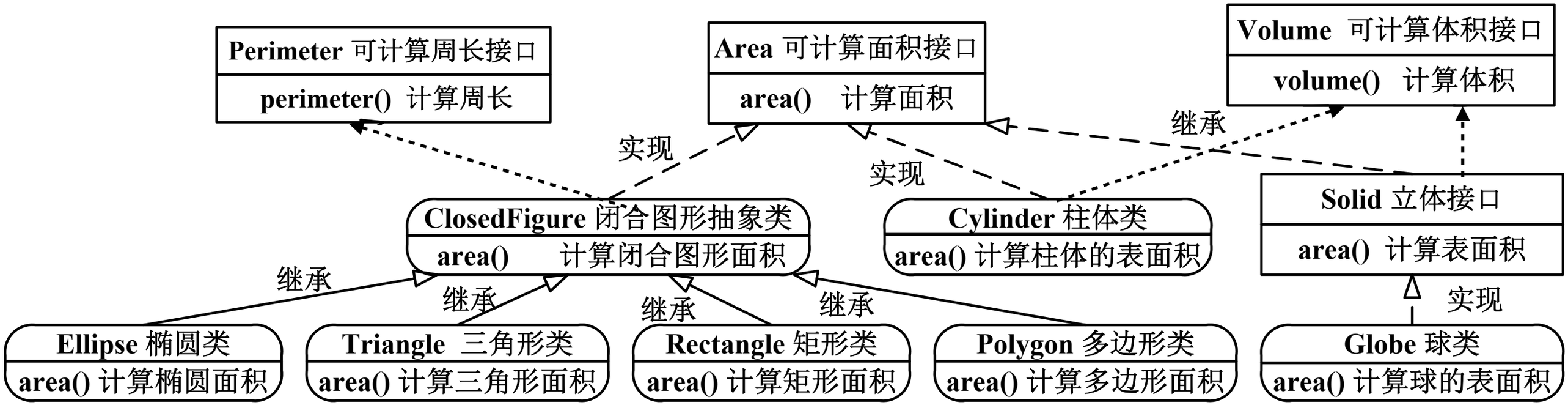
实现接口的类

否则声明为抽象类

接口的多继承性

[修饰符] class 类名 **[extends 父类列表]**
{
    新增成员变量的声明;
    新增或重写成员方法的声明及实现;
}

[public] interface 接口 [extends 父接口列表]
{
    [public] [static] [final] 数据类型 成员变量=常量值;
    [public] [abstract] 返回值类型 成员方法**[(**参数列表**)]**;
}



接口不能被实例化，但可以引用实现它的类的对象

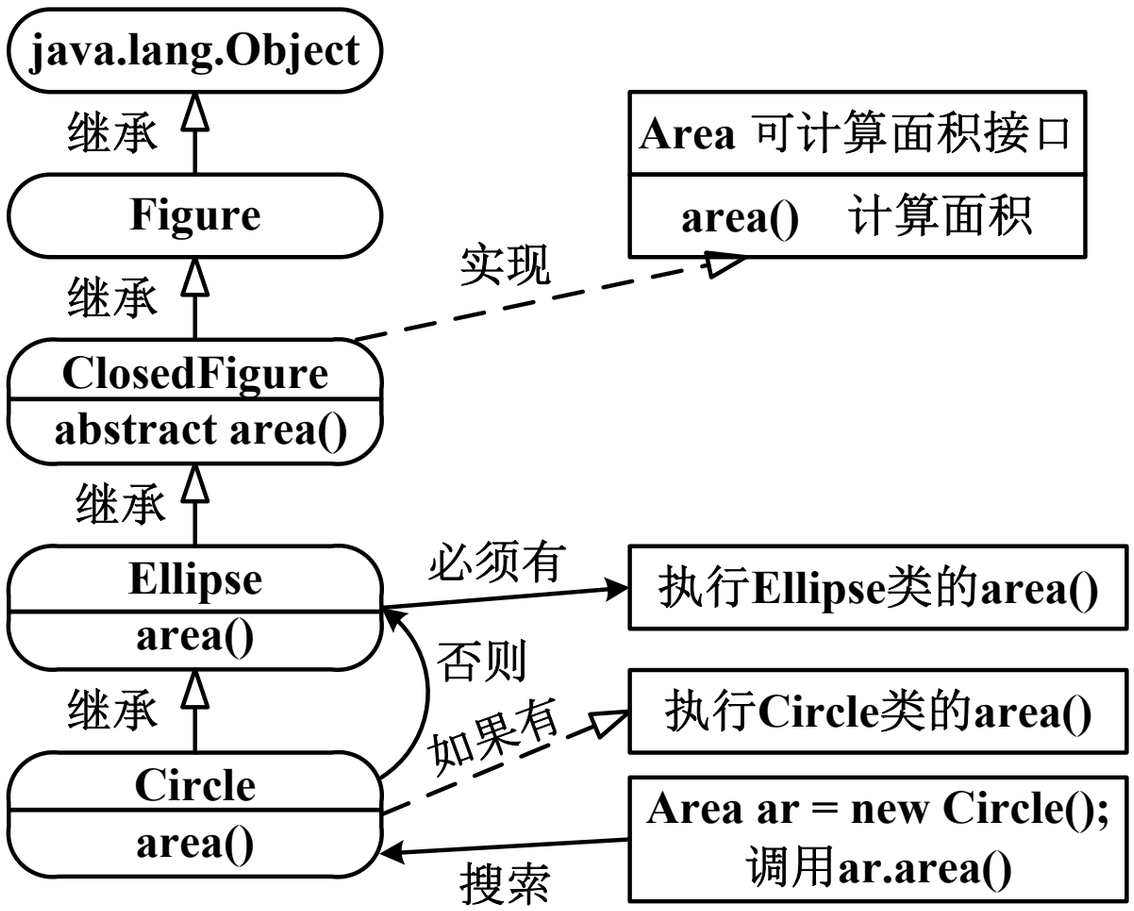
```
Cylinder cylinder = new Cylinder(fig,10);  //椭圆柱
Area ar  = cylinder;
Volume vol = cylinder;
ar.area()
vol.volume()
```

消除类型之间的

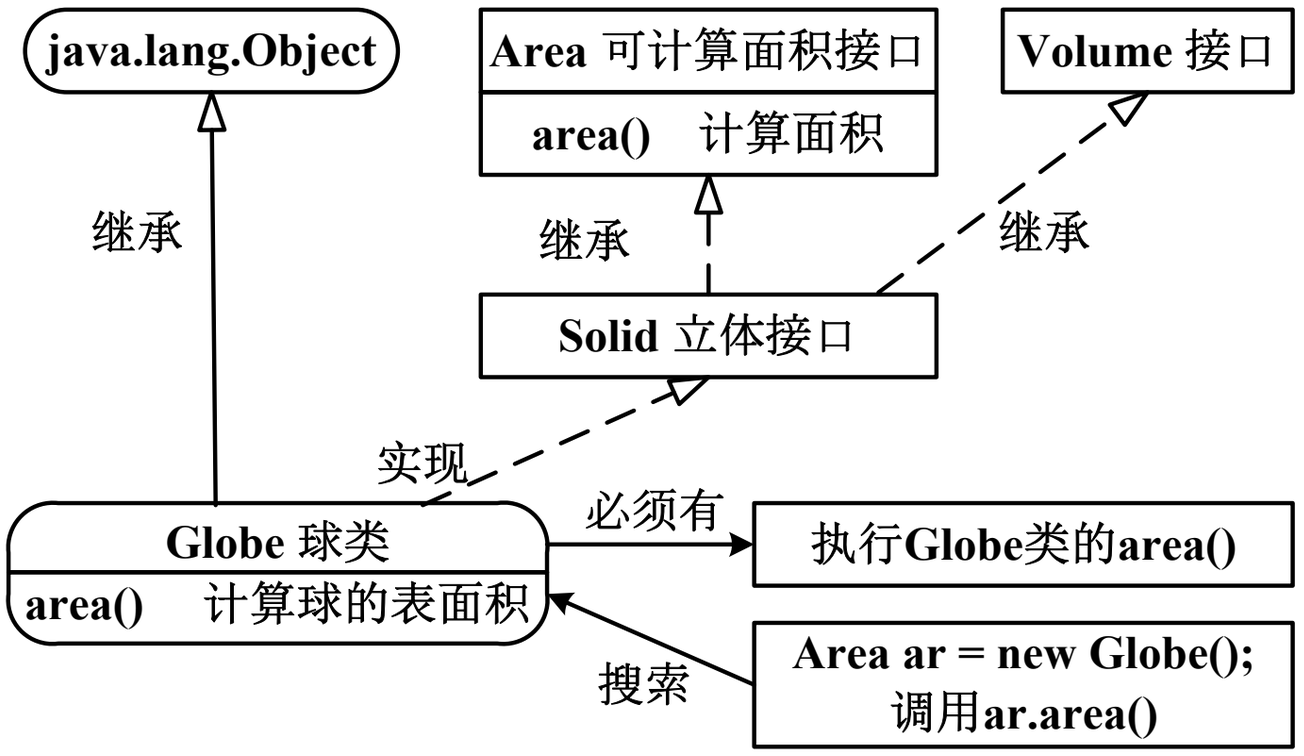
耦合关系

	抽象类 （is-a）	接口 (like-a)
默认的方法实现	它可以有默认的方法实现	接口完全是抽象的。它根本不存在方法的实现
成员变量	各种类型的	只能是public static final类型的
实现	子类使用extends关键字来继承抽象类。如果子类不是抽象类的话，它需要提供抽象类中所有声明的方法的实现。	子类使用关键字implements来实现接口。它需要提供接口中所有声明的方法的实现
构造器	抽象类可以有构造器	接口不能有构造器
与正常Java类的区别	除了你不能实例化抽象类之外，它和普通Java类没有任何区别	接口是完全不同的类型
访问修饰符	抽象方法可以有public、protected和default这些修饰符	接口方法默认修饰符是public。你不可以使用其它修饰符。
多继承	抽象方法可以继承一个类和实现多个接口	接口只可以继承一个或多个其它接口
添加新方法	如果你往抽象类中添加新的方法，你可以给它提供默认的实现。因此你不需要改变你现在的代码。	如果你往接口中添加方法，那么你必须改变实现该接口的类。

单继承的优点



（a）类的单继承使Java搜索匹配执行方法的路径是线性的



（b）接口的多继承对方法的运行时多态搜索没有影响

```java
class IntegerPoint{
    private Integer x ;
    private Integer y ;
    public void setX(Integer x){
        this.x = x ;
    }
    public void setY(Integer y){
        this.y = y ;
    }
     public Integer getX(){
         return this.x ;
     }
     public Integer getY(){
         return this.y ;
     }
}
```

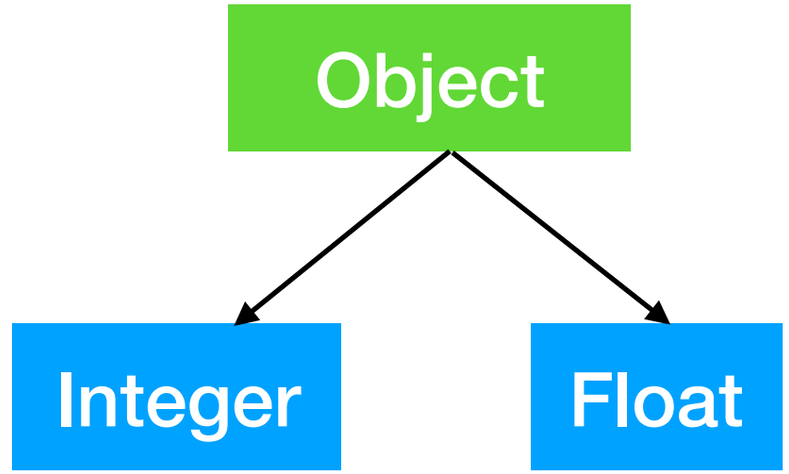
```java
class FloatPoint{
    private Float x ;
    private Float y ;
    public void setX(Float x){
        this.x = x ;
    }
    public void setY(Float y){
        this.y = y ;
    }
     public Float getX(){
         return this.x ;
     }
     public Float getY(){
         return this.y ;
     }
}
```

Programming Terms

DRY

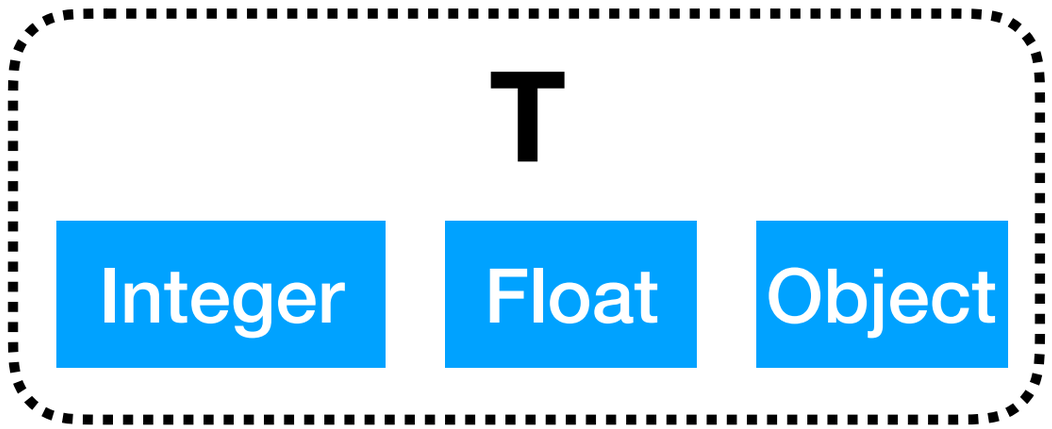
Don't Repeat Yourself

```
1.class ObjectPoint{
2.     private Object x ;
3.     private Object y ;
4.     public void setX(Object x){
5.          this.x = x ;
6.      }
7.     public void setY(Object y){
8.          this.y = y ;
9.     }
10.     public Object getX(){
11.          return this.x ;
12.      }
13.     public Object getY(){
14.          return this.y ;
15.     }
16.}
17.
```



泛型

```
class Point<T>{// 此处可以随便写标识符号
    private T x ;
    private T y ;
    public void setX(T x){//作为参数
        this.x = x ;
    }
    public void setY(T y){
        this.y = y ;
    }
     public T getX(){//作为返回值
         return this.x ;
     }
    public T getY(){
        return this.y ;
    }
};
```



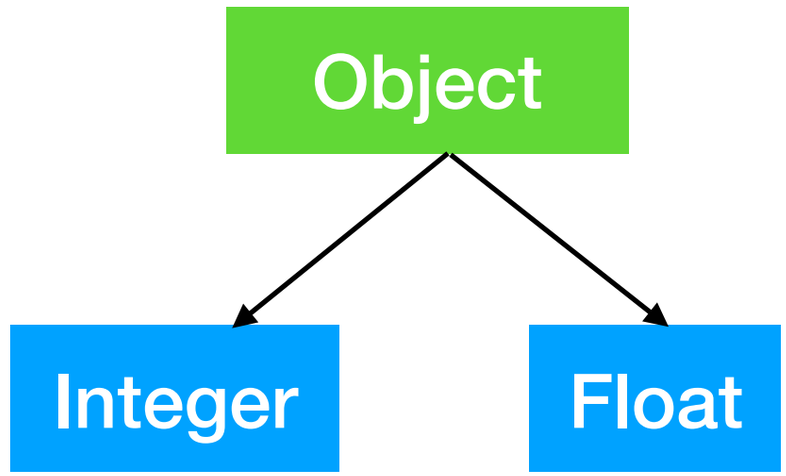

```
//IntegerPoint使用
Point<Integer> p = new Point<Integer>() ;
p.setX(new Integer(100)) ;
System.out.println(p.getX());

//FloatPoint使用
Point<Float> p = new Point<Float>() ;
p.setX(new Float(100.12f)) ;
System.out.println(p.getX());
```

Java自带泛型类

class 类名**<类型参数列表>** **[extends** 父类**]** **[implements** 接口列表**]**

[public] interface 接口**<类型参数列表>** **[extends** 父接口列表**]**

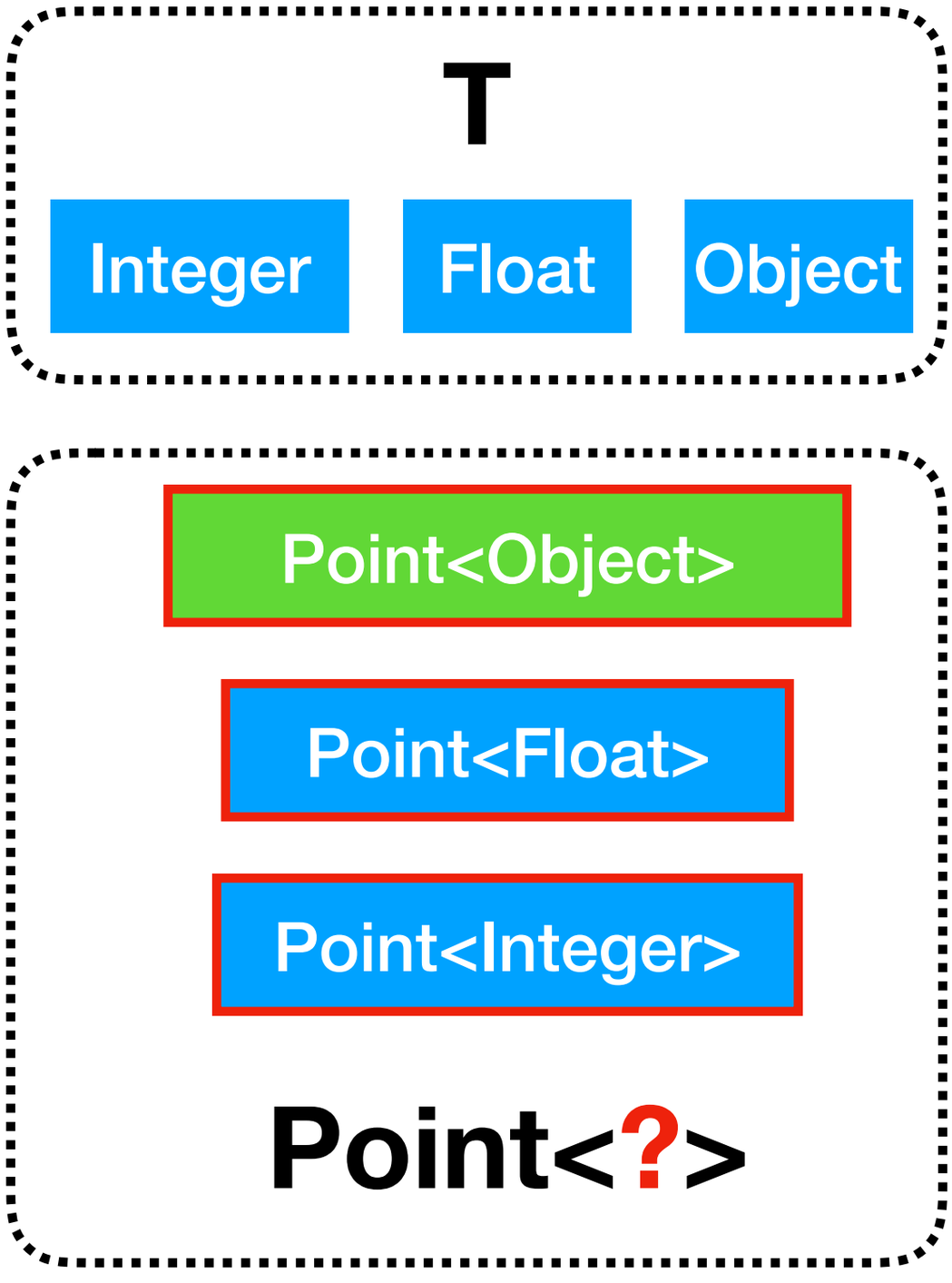
```java
class Point<T>{// 此处可以随便写标识符号
    private T x ;
    private T y ;
    public void setX(T x){//作为参数
        this.x = x ;
    }
    public void setY(T y){
        this.y = y ;
    }
     public T getX(){//作为返回值
         return this.x ;
     }
    public T getY(){
         return this.y ;
    }
};
```



Point<Object>

Point<Integer>

Point<Float>



如何使用统一的接口?

消除类型之间的

耦合关系

```
2.//IntegerPoint使用
3.Point<?> p = new Point<Integer>() ;
4.p.setX(new Integer(100)) ;
5.System.out.println(p.getX());
6.
7.//FloatPoint使用
8.Point<?> p = new Point<Float>() ;
9.p.setX(new Float(100.12f)) ;
10.System.out.println(p.getX());
11.
```

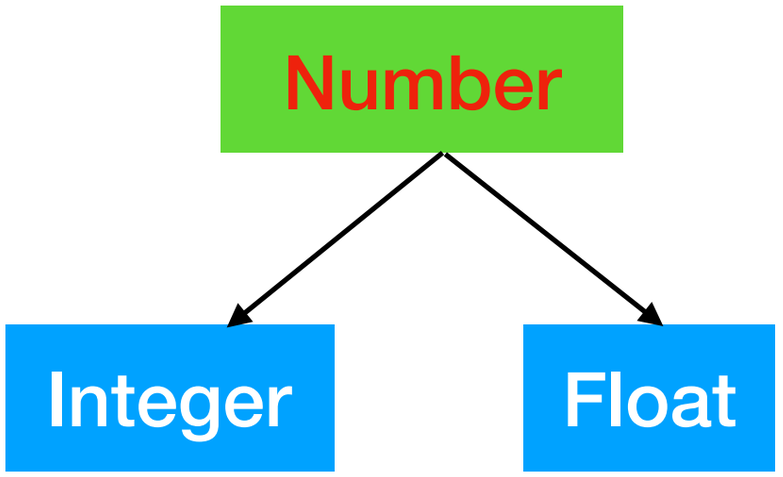
与多态思想相联系

```
public void showKeyValue1(Point<?> obj){
    Log.d("泛型测试","key value is " + obj.getX());
}
```

泛型的边界

```
class Point<T extends Number>{// 此处可以随便写标识符号
    private T x ;
    private T y ;
    public void setX(T x){//作为参数
        this.x = x ;
    }
    public void setY(T y){
        this.y = y ;
    }
    public T getX(){//作为返回值
        return this.x ;
    }
    public T getY(){
        return this.y ;
    }
};
```

```
Point<String> p = new Point<String>() ;
p.setX(new Integer("abc")) ;
System.out.println(p.getX());
```



泛型方法

与重载区别!!!

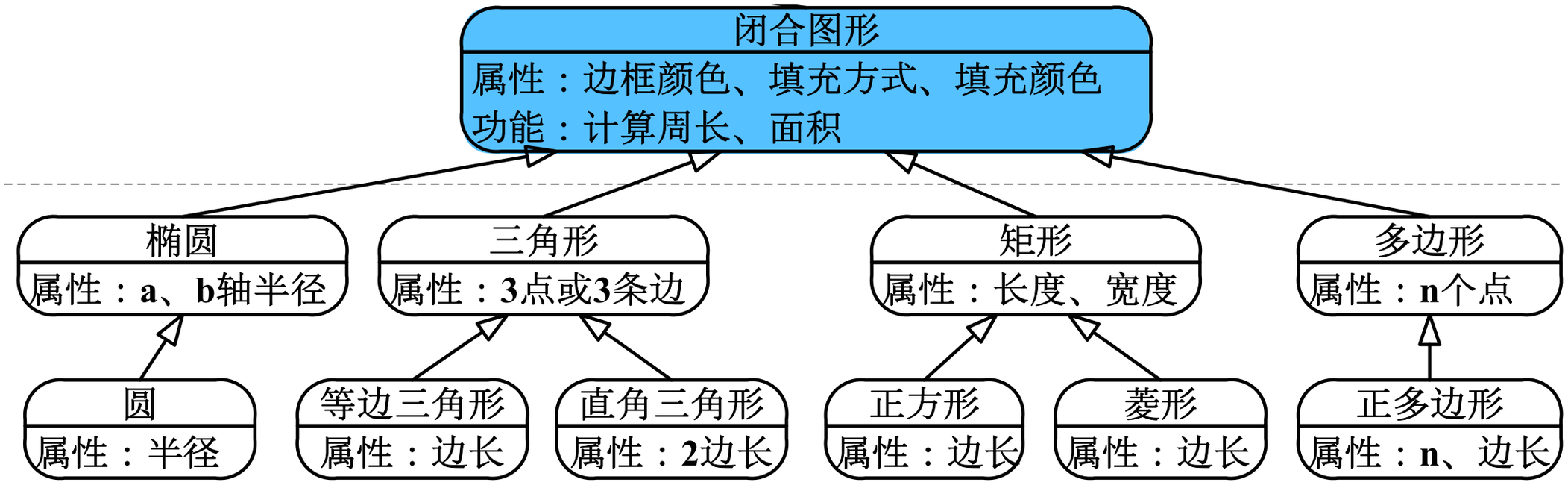
[public] [static] **<类型参数列表>** 返回值类型 方法([参数列表])

```
1.public class StaticFans {
2.     //静态函数
3.     public static  <T> void StaticMethod(T a){
4.          Log.d("harvic","StaticMethod: "+a.toString());
5.     }
6.     //普通函数
7.     public  <T> void OtherMethod(T a){
8.          Log.d("harvic","OtherMethod: "+a.toString());
9.     }
10.}
11.
```

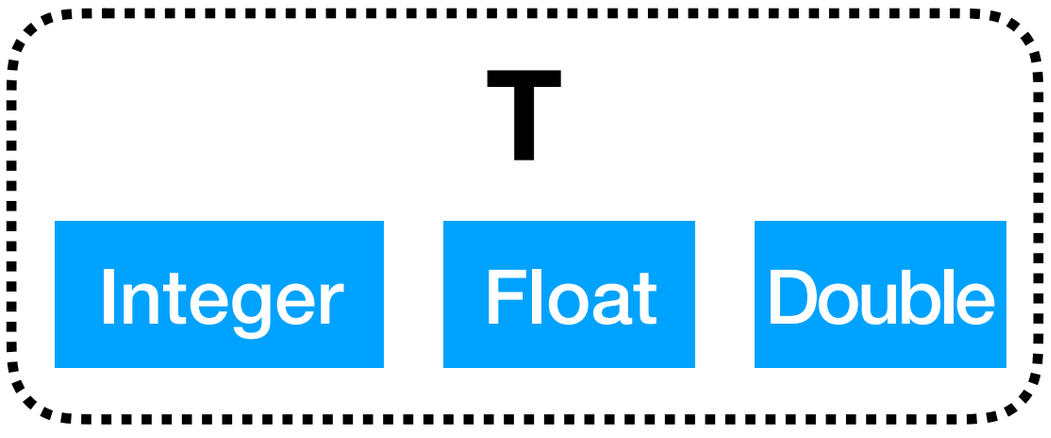
```
1.//静态方法
2.StaticFans.StaticMethod("adfdsa");//使用方法一
3.StaticFans.<String>StaticMethod("adfdsa");//使用方法二
4.
5.//常规方法
6.StaticFans staticFans = new StaticFans();
7.staticFans.OtherMethod(new Integer(123));//使用方法一
8.staticFans.<Integer>OtherMethod(new Integer(123));//使用方法二
9.
```

推荐

三种抽象方法


闭合图形
属性：边框颜色、填充方式、填充颜色
功能：计算周长、面积
椭圆
属性：a、b轴半径
三角形
属性：3点或3条边
矩形
属性：长度、宽度
多边形
属性：n个点
圆
属性：半径
等边三角形
属性：边长
直角三角形
属性：2边长
正方形
属性：边长
菱形
属性：边长
正多边形
属性：n、边长


跑
飞
叫
行为的抽象


T
Integer
Float
Double