**is-a has-a**

is-a( 是 "a"  小明是人类)表示的是属于得关系。比如兔子属于一种动物（继承关系）。

has-a( 有 "a"  汽车有轮胎) 表示组合，包含关系。比如兔子包含有腿，头等组件；就不能说兔子腿是属于一种兔子（不能说是继承关系）

**7.1.2 "is a"关系和"has a"关系**

假设有两个类：Computer和Employee。明显地，这两个类之间不存在"is a"的关系，即Employee不是计算机，它们之间没有继承关系的必要。因此不可能产生代码重用性。但这两个类之间是"has a"关系，即是支持的关系。例如，Employee"has a"Computer。明显地是一种支持关系。这种支持关系落实到代码中，就是在Employee中创建Computer的对象，调用其方法，到达完成某种运算和操作的目的。

Employee和Manager 类存在的则是"is a"关系，即Manager是Employee。它们之间存在共性，或者共同的属性。Manager是Employee的具体化；Employee是Manager的概括和抽象。概括性和抽象性的类，如Employee，在继承中则定义为超类。具体或代表对象特性的类，如Manager，则定义为子类。如果这是一个用来计算雇员工资的程序，那么在超类Employee中，我们应当包括所有子类都应该具有的、与计算工资有关的数据，例如name、employeeID、jobTitle、seniority、baseSalary以及用来计算基本工资部分的方法，如baseSalary()等。在Manager这个子类中，我们不仅继承Employee的所有数据和方法，还增加针对Manager的新的数据，如是否董事会成员boardMember、职务补贴merit等，因为除基本工资的计算之外，这些都影响到具有经理职务雇员的收入。

对两个类之间"is a"或是"has a"关系的分析，有助于我们确定它们之间是否存在继承关系，避免设计上的错误，因而达到提高代码重用性的目的

1）抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。举个简单的例子，飞机和鸟是不同类的事物，但是它们都有一个共性，就是都会飞。那么在设计的时候，可以将飞机设计为一个类Airplane，将鸟设计为一个类Bird，但是不能将 飞行 这个特性也设计为类，因此它只是一个行为特性，并不是对一类事物的抽象描述。此时可以将 飞行 设计为一个接口Fly，包含方法fly( )，然后Airplane和Bird分别根据自己的需要实现Fly这个接口。然后至于有不同种类的飞机，比如战斗机、民用飞机等直接继承Airplane即可，对于鸟也是类似的，不同种类的鸟直接继承Bird类即可。从这里可以看出，继承是一个 “是不是”的关系，而 接口 实现则是 “有没有”的关系。如果一个类继承了某个抽象类，则子类必定是抽象类的种类，而接口实现则是有没有、具备不具备的关系，比如鸟是否能飞（或者是否具备飞行这个特点），能飞行则可以实现这个接口，不能飞行就不实现这个接口。