

- 面向对象程序设计
- 抽象
- 封装
- 继承
- 多态

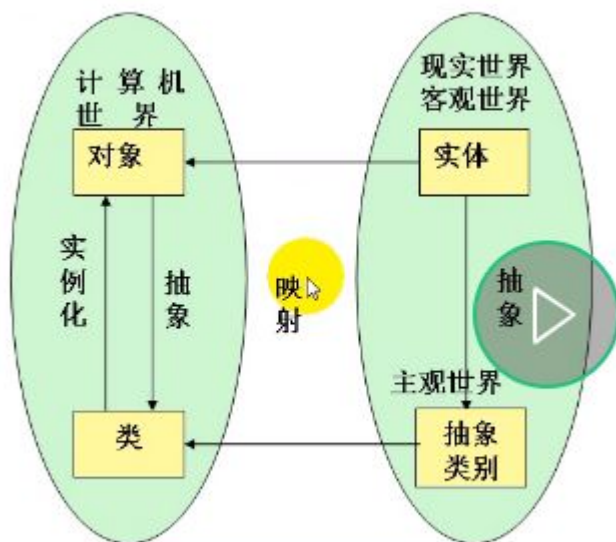
面向对象程序设计：

- ✚ 面向对象（Object Oriented）是认识事务的一种方法，是以对象为中心的思维方式。
- ✚ 面向对象的程序设计
 - ✧ 对象=算法+数据结构，算法和数据结构是紧密结合的，而不是分离的。这就是的对象具有更强的适应性，外部的变化不会影响到对象内部的数据结构。因为这个数据对于外界来说是隐藏的。
 - ✧ 程序=对象+对象+...+对象
- ✚ 面向对象程序设计模拟自然界认识和处理事务的方法，将数据和对数据的操作方法放到一起，形成一个相对独立的整体-----**对象（object）**，同类对象还可以抽象出共性，形成**类（class）**，一个类中的数据通常只能通过本类提供的方法进行处理，这些方法称为该类与外部的**接口**，对象之间通过**消息（message）**进行通讯。

数据抽象

- ✚ 结构化程序设计采用的是**过程抽象**（以**算法为中心**，或者说以**过程为中心**，优先考虑的是**解决问题的过程**，采用**自顶向下，逐步求精的方法**来解决问题，这就要求程序员对解决问题的整个过程非常的清楚，只要一个环节出现了问题，就容易失败。），所谓的过程抽象是将问题域中具有明确功能定义的操作抽取出来，将其作为一个实体来看待。
- ✚ **数据抽象**是较过程抽象**更高级别**的抽象方式，将描述客体的**属性和行为**绑定到一起，实现统一的抽象，从而达到从现实世界客体的真实模拟。也就是说他将系统或者程序看成对象+对象+对象。它优先考虑的不是解决问题的流程，而是这个问题是有什么对象构成的，或者说他是以数据为中心考虑问题。这些对象确定之后才考虑这些对象之间的**相互作用**，这个相互作用就是解决问题的过程，因而他是淡化过程的，只要这个对象的功能相对来说是稳定的，那么他就更容易的达到复用的目的，即使我对对象之间的相互作用没有想明白，这个对象的功能是稳定的，它还是可以复用的，甚至复用到其他的系统当中。并不会因为我们对流程没有理解好，使得这些功能不能复用，使这些对象变得一无是处。结构化程序设计的思想就容易出现这种情况，我们说编写代码是自顶向下，一些数据发生了改变就会使这些代码也跟着发生改变，他是因为操作数据的代码和数据之间不是紧密绑定的，所以说结构化程序设计的思想**最核心的问题就是操作数据和代码是分离的**。因而数据发生了变化，就会影响这些代码。而面向对象呢？对象内部的功能相对来说是比较稳定的，它将对象所相应的数据以及行为封装在对象的内部，所以说这一部分的内容，相对来说是比较稳定的，主要原因是代码与数据是紧密结合的，因为对象与对象之间的作用关系，即使我们没有考虑清楚，这个对象也是可以复用的，这是面向对象的一个观点。它采用的是数据抽象。
- ✚ **实体、对象和类之间的关系**

现实生活中的实体可以抽象出类别的概念，对于计算世界就有一个类（class）的概念，因为类是一个抽象的概念对应体，所以计算机不给它分配内存，只给对象分配内存，图表示计算机世界和现实世界之间的对象关系。



对象、实体与类

从计算机的观点如何看待对象：

- ✚ 对象是计算机内存中的一块区域，通过内存分块每个对象在功能上相对独立。这个对象不仅包含了数据也包含了代码，并且这些数据是受保护的。
- ✚ 这些内存不但存储数据，也存储代码，这保证对象是受保护的，只有对象中的代码能访问存储于对象中的数据，这清楚的界定了对象所具有的功能，并使得对象不受未知外部事件的影响，从而使自己的数据和功能不会因此遭受破坏。
- ✚ 对象之间只能通过函数调用也就是发送消息来实现相互通信。
- ✚ 当对象的一个函数被调用的时候，对象执行内部的代码来响应该调用，从而使得对象呈现一定的行为，这个行为及其呈现的结果就是该对象所具备的功能。

面向对象的基本特征



抽象

- 抽象是人认识事物的一种方法。
- 抓住事物本质，而不是抓住内部的细节或者具体的实现。比如说我们看待房屋，我们认为房屋是可以居住的，而不会局限于它内部的细节，比如说它是有钉子构成的，木材，钢筋水泥，也就是说我们关注的是事物的本质，而不是内部的细节，具体细节，具体实现，把木材，水泥，钢筋，钉子进行抽象，形成了房屋。这就是人们认识事物的一种方法。人们认识事物天然就具有一种抽象事物的一种方式，它将一些更加具体的事物，进行抽象，形成一个更加一般的事物，从具体到一般的一个过程就是抽象。
- 从具体到一般的过程就是抽象。
- 房屋：钉子，水泥，木材，钢筋等构成。人们认识房屋的时候，人们总是能够认识到房屋的一个本质是居住的，而不会局限于房屋是什么做的具体的细节。
- 如果我们进一步的进行抽象，将一群房屋可以抽象成一个城镇。它是有一些的房屋构成的，建筑物构成的，
- 从具体到一般的过程是一个归纳的过程，也是抽象的过程。
- 将对象一些共同的特征抽取出来就形成了类的一个概念：

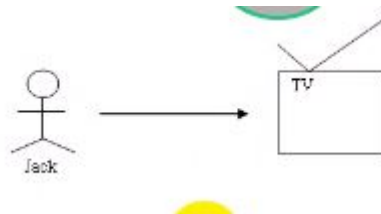
对象->类（归类）

- 人类认识事物时还会有另外一种思想，就是分类的思想，将大类分为小类，比如说将动物分为哺乳动物、猫科动物、两栖动物等等，分成更小的类别，这运用的就是分类的思想。



封装

- 封装是指按照**信息屏蔽**的原则，把对象的**属性和操作**结合在一起，构成一个独立的对象。（外部的代码是无法操纵这些数据的）。外部的代码只能通过函数调用来调用对象内部的代码。操纵这些数据，因为数据是封装的。我们只能提供一定的接口，让外界来通过接口对对象进行操作。
- 通过限制对属性和操作的访问权限，可以将属性“隐蔽”在对象的内部，对外提供一定的接口，在对象之外只通过接口对对象进行操作。
- 封装增加了对象的独立性，从而保证了数据的可靠性。这里我们可以举例，比如我们的插座：插座上面都有插口，
插座 插口
将插头插入插口，
就能够获得电流（而我们根本不关心这个电流是怎么来的，它可能是风能或者水能发电，甚至是核能发电，但是我们只管通过插口获得电流，根本就不知道内部的细节）
- 封装使得外部的使用者更加的专注。如果从哲学的观点是封装是对我们更好的服务。虽然他将信息隐藏了，但是他将我们需要的信息给我们。这才是对我们最好的服务。如果他暴露了很多我们不关心的事物，反而不能更好的对我们服务。这是辩证的，跟我们有时候越少的东西，相反有时候是越好的。
- 外部对象不能直接操作对象的属性，只能使用对象提供的服务。

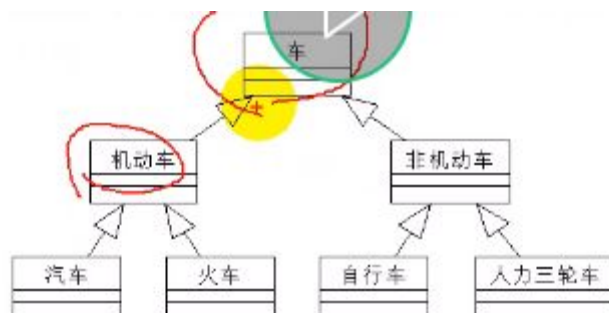


这里举的例子是人操作电视机，我们并不关心电视机的内部的工作原理，电视机提供了选台、调音量等功能供我们使用。



继承

- 继承表达了对象一般与特殊的关系，特殊的类的对象具有一般的类的全部的属性和服务。
 - （1）从一般到特殊的过程就是分类。 **对象->类（归类）**
 - （2）从特殊到一般的过程叫做抽象 **大类->小类（分类）**
- 继承可以使我们用**现成的一些类**来构造**新的类**，就是**拿来主义**。那么现有一些工作良好的类，它也派生出新的类，这些新的类或者说跟这些现有的事物呢？大相径庭，那么我们就拿来就用，只不过需要对它进行一些稍微的扩充修改，大大提高了一些程序设计的效率。
- 继承性大大的简化了对问题的描述，大大的提高了程序的可重用性，从而提高了程序设计、修改、扩充的效率。我们一些良好的类直接拿过来就使用了。比如说这里有个例子：比如说车子：



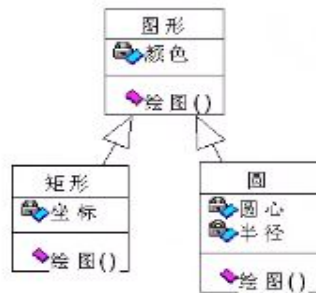
继承具有传递性，如汽车具有车的全部属性和行为。



多态

- 多态性：同一个消息可以被不同的对象接收的时候，产生不同的效果，即**同一接口（相同的消息），不同方法。**
- 多态可以使我们可以以一致的观点看待不同（但有大相径庭的对象）的对象。这些对象有一致的接口。同一个消息发送给不同的对象就会呈现出不同的行为。
- 一般类中定义的属性和服务，在特殊类中不改变其名字，但是通过不同的实现，可以具有不同的数据类型或者不同的行为。

具有不同的行为。



当向图形对象发送消息进行绘图服务请求后，图形对象会自动判断自己的所属类然后执行相应的绘图服务。



继承和多态性组合，可以生成很多相似但是独一无二的对象，继承性使得这些对象可以共享许多的相似特性，而多态又会使同一个操作对不同对象产生不同表现形式，这样不仅提高了程序设计的灵活性，而且减轻了分别设计的负担。

面向对象的思想总结：

- 面向对象是一种认识世界的方法，也是一种程序设计方法。
- 面向对象的观点认为，客观世界是由各种各样的设计实体，也就是对象组成的，没有对象都有自己**内部状态（静态数据）**和**运动规律（动态行为）**，不同的对象之间的相互关系和相互作用就构成了各种不同的系统，并进而构成了整个客观世界。

面向对象编程方法的特性：

- 程序设计的重点在数据而不是函数。
- 程序由对象组成，建立对象的目的是为了完成一个步骤，而是为了描述某个事物在整个解决的步骤的行为。
- 对象之间通过相互协作来完成功能。
- 函数与相关的数据紧密结合。
- 数据可以被隐藏。
- 很容易扩充新的数据和函数。

面向对象的一个很重要的原则，：

开闭原则：对增加开放，对修改、删除关闭，这样就不会使外界使用该对象的原有系统不会发生改变。如果我们要新增加一个功能，我们就要新增加一个系统来增加他的功能。但是原有的系统仍然是可以工作的，仅仅是没有这个功能而已。


面向对象编程的优缺点：

面向对象的优点

- **易维护：**可读性高，即使改变需求，由于继承的存在，维护也只是在局部模块，维护起来是非常方便和较低成本的。
- **质量高：**可重用现有的，在以前的项目的领域中已被测试的类使系统满足业务需求并具有较高的质量。
- **效率高：**在软件开发时，根据设计的需要，对现实世界的事物进行抽象、产生类。这样的方法解决问题，接近于日常生活和自然的思考方式，势必提高软件开发的效率和质量。
- **易扩展：**由于**继承、封装、多态**的特性，自然设计出**高内聚，低耦合**的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

机器语言站在机器的空间解决问题。

汇编语言也是站在机器空间（访问内存等）解决问题。
面向过程是站在问题空间解决问题。以算法为中心，当系统比较大的时候，我们无法把握整个系统的流程。
而面向对象的是站在问题空间解决问题，以对象为中心，把系统看成对象组成的。所以说它更符合人们解决问题的方法。

 运行效率会下降 10%，通常会比 C 语言低一些。