对象的一种功能或行为，用来描述它的动态特征。以汽车为例，它拥有的发动机，变速箱，车身长、宽、高，重量等是对象的属性，而启动、加速、刹车等是它的方法。

对象是一个封闭体，它向外界提供一组接口，外界通过这组接口与对象交互，这样对象就具有较强的独立性、自治性和模块性。

2. 类

类是一组具有相同属性和相同操作的对象的集合，类的属性用来表示对象的各种特征，类的操作是对于对象行为的描述，包括操作名称和操作实现过程。把众多的事物归纳成一些类是人们认识客观世界时的常用方法。分类所依据的原则是抽象，即忽略事物的非本质特征，只关注与当前目标有关的本质特征，从而找出事物的共性，把具有共性的事物划为一类，得出一个抽象的类的概念。类与对象的关系是抽象与具体的关系，类是多个对象的综合抽象，对象是类的个体实例。

3. 消息

对象与对象之间并不是彼此孤立的，它们之间存在联系，在面向对象的系统中，对象之间的联系是通过消息传递进行的，消息是对象之间相互请求和相互协作的途径，是要求某个对象执行其中某个功能操作的规格说明。通过发送消息操纵对象，对象接收消息，根据消息及消息参数调用相关的服务，进行处理并予以响应，从而实现系统功能。

3.2.4 基本特征

面向对象的方法认为客观世界是由各种“对象”所组成的，任何事物都是对象，每一个对象都有自己的运动规律和内部状态，每一个对象都属于某个“类”。复杂的对象可以是由相对简单的对象以某种方式而构成的。通过类比，发现对象间的相似性，即对象间的共同属性和行为，这就是构成类的依据，对象间的相互联系是通过传递“消息”来完成的，消息就是通知对象去完成一个允许作用于该对象的操作。

(1) 抽象性：关注与当前目标有关的本质特征，忽略非本质特征，找出事物的共性，归为一类，就会得到一个抽象的概念。抽象包括两个方面，一是过程抽象，二是数据抽象。过程抽象是指任何一个明确定义功能的操作都可被使用者看作单个的实体。数据抽象定义了数据类型和施加于该类型对象上的操作，并限定了对象的值只能通过这些操作来访问和修改。

(2) 封装性：它将属性和操作结合在一个类中，对象的属性一般不被外界直接访问，而是通过对象的操作来读取和修改；封装隐蔽对象的内部细节，只留少量接口，接收外界的消息。封装保证模块具有良好的独立性，便于系统维护，对系统的修改仅限于类的内部，是面向对象的特征之一。

(3) 继承性：广义地说，继承是指能够直接获得已有的性质和特征，而不必重复定义。在面向对象的软件技术中，继承是子类自动地共享父类中已定义的属性和方法。子类继承父类的所有属性和方法，避免了许多重复性的工作，因此子类的属性与操作有自己定义的，也有从父类继承来的。继承是传递的，当子类被更下层的子类继承时，它所继承的和自己定

义的属性和操作又被下一层继承下去。继承是面向对象方法中最显著的特点，提高了软件的可重用性，使得软件的可扩充性大大加强。类的继承关系如图 3-2 所示。

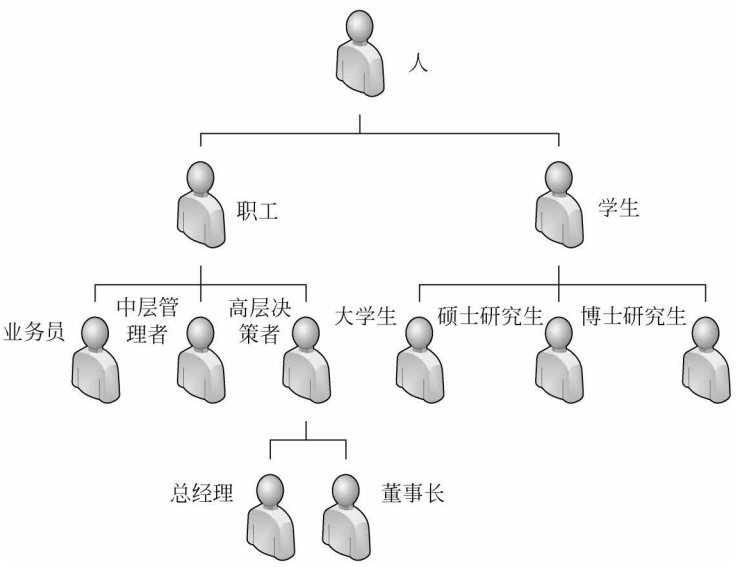

图 3-2　类的继承关系

（4）多态性：相同的方法作用于类型不同的对象上可以得到不一样的结果，每个对象通过满足自身条件的方式响应同样的消息，使得软件的可重用性和灵活性进一步增加。例如打开窗户的操作，基于窗户的不同，可以得到平开、推拉、上悬等多种不同的结果。

继承性和多态性的结合，可以生成一系列相互类似但独一无二的对象。继承性使得这些对象共享许多相似的特征，多态性针对相同的消息，使得不同对象可以有独特的表现方式，实现个性化的设计。

3.2.5　开发过程

按照系统开发生命周期的理论，面向对象的信息系统开发方法可以分为以下 5 个阶段。

1. 面向对象的分析

面向对象分析（Object Oriented Analysis，OOA）是系统开发过程中的问题定义阶段，目标是完成对所求解问题的分析，确定系统“做什么”，并建立系统模型。

面向对象的分析需要对问题域和系统责任进行分析和理解，问题域有哪些值得考虑的事物，面向对象的分析模型中就有哪些对象，而且强调对象的命名与客观事物一致，找出描述它们的类和对象，定义其属性和操作。用一般-特殊结构描述事物间的继承关系，用整体-部分结构描述事物间的组成关系，用实例连接和消息连接表示事物间的静态联系和动态联系，最终获得一个符合用户需求，并能够反映问题域和系统责任的面向对象分析模型。

通过面向对象分析建立的系统模型是以对象概念为中心的，它由一组相关的类组成。面向对象分析可以采用自顶向下的方法，逐层分解建立系统模型，也可以自底向上从已定义

的类出发，逐步构造新类。

2. 面向对象的设计

面向对象的分析建立了反映问题域的对象分析模型，不考虑与系统具体实现有关的因素（如编程语言、用户界面、数据库等），从而使面向对象的分析模型独立于具体实现。面向对象的设计（Object Oriented Design，OOD）则是面向对象系统开发方法在设计阶段应用与扩展的结果，是将面向对象分析阶段所创建的分析模型转换为设计模型，解决“如何做”的问题。面向对象设计的目标是产生一个满足用户需求、可实现的设计模型。

面向对象的设计内容主要包括两部分：一是不经过转换，仅做部分必要的修改和调整，把面向对象的分析模型直接用于面向对象的设计，作为面向对象设计的一个部分；二是针对具体实现中的人机界面、数据存储、任务管理等因素补充与实现有关的部分，采用面向对象的分析方式分析相同的表示法和模型结构。

3. 面向对象的编程

面向对象编程（Object Oriented Programming，OOP）的任务是将从面向对象分析和设计得到的模型用程序加以实现，即采用面向对象的程序设计语言，为面向对象的设计模型的每个成分编写程序。理想的面向对象开发规范，要求在面向对象的分析、设计阶段就对系统中的对象及其内部构成（属性和操作）与外部关系（静态和动态联系）有透彻的认识和清晰的描述，而不是把这些问题留给程序员去思考。程序员只需用具体的数据结构定义对象的属性，用具体的语句实现流程图所表示的算法。

4. 面向对象的测试

面向对象的测试（Object Oriented Testing，OOT）以类作为基本测试单位，差错范围主要是类定义之内的属性和操作，以及有限的对外接口（消息）所涉及的部分，可以大大减少错误的影响范围。一个类通常包括一组不同的操作，而一个操作也可能存在于一组不同的类中，所以类的测试不能再孤立地测试单个操作，而应该把操作作为类的一部分。此外，由于继承性的存在，面向对象的测试在完成对父类的测试后，子类的测试重点只是那些新定义的属性和操作。

5. 面向对象的维护

在面向对象的维护（Object Oriented System Maintenance，OOSM）中，程序与问题域是一致的，各个阶段的表示也是一致的，从而减少了理解的难度。无论是发现程序中的错误而逆向追溯到问题域，还是需求发生变化而从问题域正向跟踪到程序，都是很方便的。

3.2.6 特点

1. 面向对象开发方法的优点

（1）良好的可复用性：基于类建立的系统模型，与基于“过程”和“数据”建立的系统模

型相比，更具稳定性，增强了系统的适应性，对复用支持程度高。

（2）易于维护：由于对象和类的规范性，维护人员易于理解运行过程和原理，可维护性好。

（3）良好的可扩充性：以对象和类为基础，实现了从对客观世界对象客体的描述到软件结构的直接转换，大大减少了后续软件开发量，缩短了开发周期。

2. 面向对象开发方法的缺点

（1）面向对象方法的关键是从客观世界抽象出对象，但是客观世界的复杂性，使得完成对象的抽象比较困难。

（2）面向对象的开发方法，需要有一定的软件基础支持才能应用。

（3）如果大型系统开发中，一开始就采用自底向上的面向对象方法开发系统，而不经自顶向下的整体划分，易造成系统结构不合理、各部分关系失调等问题。因此，面向对象的开发方法与结构化系统开发方法在系统开发中相互依存、不可替代。

3.3 计算机辅助软件工程

3.3.1 基本思想

计算机辅助软件工程(Computer Aided Software Engineering，CASE)是计算机技术在系统开发活动、技术和方法中的应用，是软件工具与开发方法的结合体，它使得人们能在计算机的辅助下进行软件开发，为计算机软件开发的工程化、自动化进而智能化打下基础。计算机辅助软件工程是提高系统开发效率与质量的重要途径。如果严格地从认知方法论的角度来看，CASE 是技术，但从 CASE 的发展对系统开发过程所支持的程度来看，又不失为一种实用的系统开发方法。

具体地说，CASE 能生成各种需求分析、功能分析和结构图表(如数据流图、结构图、实体联系图等)，进而成为支持整个系统开发全过程的一种大型综合系统，能支持除系统调查之外的所有系统开发过程，它可以按照系统开发商规定的应用规则，由计算机自动生成合适的计算机程序。CASE 的实质是为系统开发人员提供了一组优化的、集成的、且能节省人力的系统开发工具，实现软件生命期各阶段的自动化，帮助开发者方便、快捷地产生出系统开发过程中的各类图表、程序和说明性文档，着眼于系统分析、设计、开发、实施和维护等各个环节的自动化，使开发工作成为以自动化工具和支撑环境支持的自动化过程，并使其成为一个整体。

CASE 的目的是使开发支持工具与开发方法结合起来，通过实现分析、设计与程序开发、维护的自动化，提高信息系统开发的效率和信息系统的质量，最终实现系统开发的自动化。