



河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《计算机程序设计》课程

第09章：类与对象

黄子杰

河南中医药大学信息技术学院（智能医疗行业学院）与河南方和信息科技有限公司 联合建设
河南中医药大学信息技术学院智能医疗教研室

<https://aitcm.hactcm.edu.cn>

2025/2/16

本章概要

概述：

在Java中，类和对象是面向对象编程的核心概念。类是一种定义对象属性和方法的蓝图或模板，而对象是类的实例化，即类的具体实现。类可以包含数据成员（字段或属性）和方法成员（函数或操作）。



9.1 面向对象程序的基本概念

- 面向对象程序的基本概念涵盖了类和对象的重要概念。**类是对现实世界中对象的抽象**，包含属性和方法，而**对象则是类的具体实例**，拥有独立的属性值和共享的方法。举例来说，类如同汽车的定义，而具体的汽车则是对象，每辆车有自己的特征和行为。通过类和对象的概念，我们能够更好地组织和管理程序中的数据和功能。
- **面向对象程序的架构是以类和对象为核心，通过封装、继承、多态等特性来组织代码**，旨在提高代码的复用性、可维护性和扩展性。封装将对象的状态和行为封装在类中，并提供对外的接口访问。继承实现代码复用，子类继承父类的属性和方法。多态则提高系统的灵活性，不同实现却具有相同接口。这种架构方式类似于一家公司，部门是类，员工是对象，实现部门管理员工，员工调动到其他部门（继承），不同部门用同样方式管理员工（多态）。



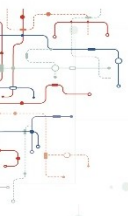
9.1 面向对象程序的基本概念

- 面向对象编程（OOP）是一种程序设计思想，它将数据和对数据的操作方法结合在一起，以对象的形式表示。在Java中，一个对象由一个类来定义。
- 以下是一个简单的Java类和对象的示例：



eg9.1.txt





9.1.1 类与对象

- 概述

类（Class）是面向对象编程（OOP）的核心概念，它是对现实世界中的对象的抽象，是具有相似特性和行为的一组对象的集合。

对象（Object）是类的实例化，是实际存在的实体，拥有类定义的属性和行为。

类与对象的关系：

类定义了对象的结构和行为，而对象则是类的具体实现，每个对象都是根据类的定义独立创建的，体现了面向对象编程的核心思想。

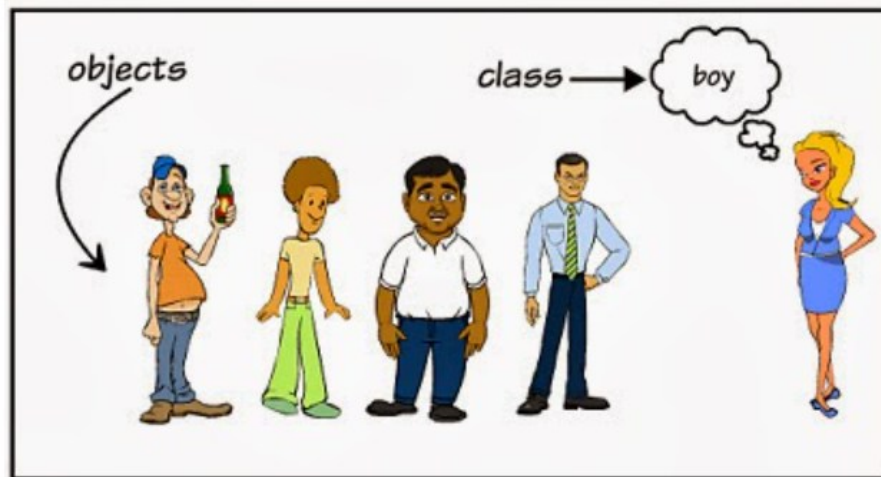
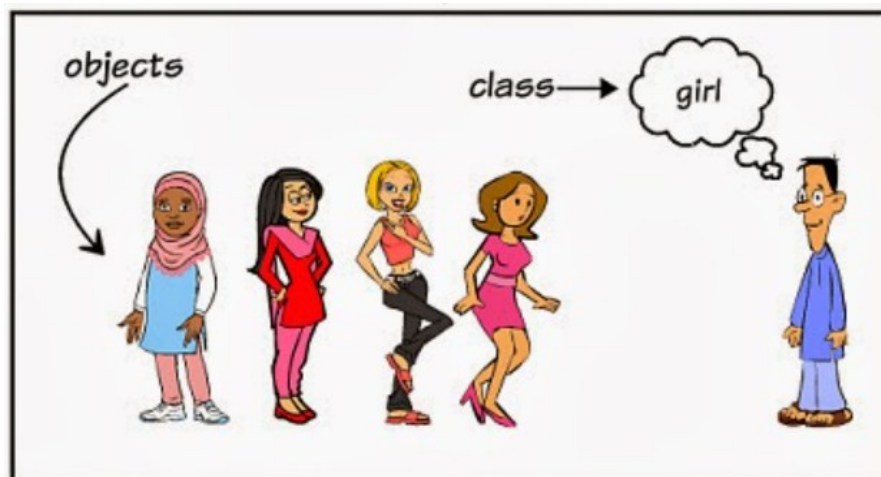


9.1.1 类与对象

- 对象：对象是类的一个实例（对象不是找个女朋友），有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。
- 类：类是一个模板，它描述一类对象的行为和状态。

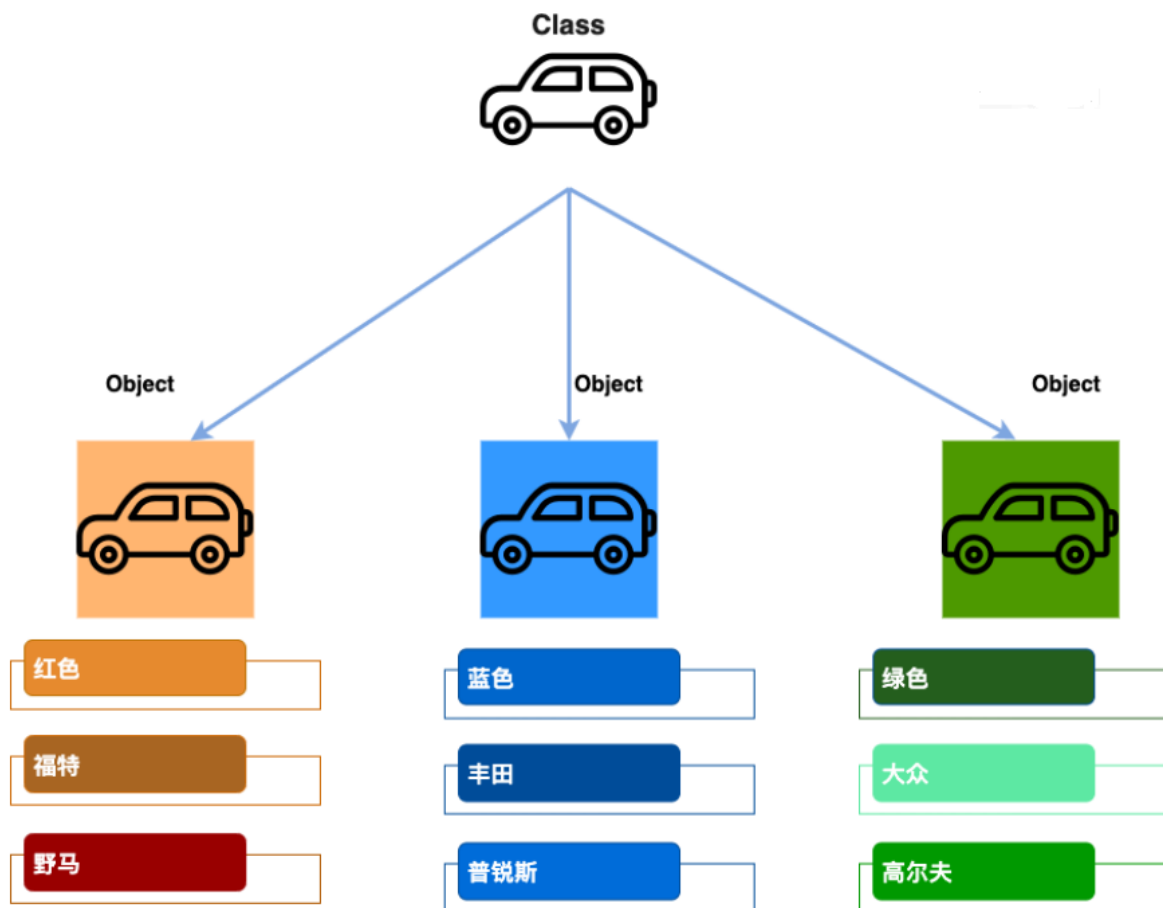
9.1.1 类与对象

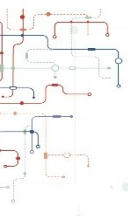
- 下图中男孩 (boy)、女孩 (girl) 为类 (class)，而具体的每个人为该类的对象 (object)：



9.1.1 类与对象

- 下图中汽车为类（class），而具体的每辆车为该汽车类的对象（object），对象包含了汽车的颜色、品牌、名称等。





9.1.1 类与对象

- 现在让我们深入了解什么是对象。看看周围真实的世界，会发现身边有很多对象，车，狗，人等等。所有这些对象都有自己的状态和行为。

拿一条狗来举例，它的状态有：名字、品种、颜色，行为有：叫、摇尾巴和跑。

对比现实对象和软件对象，它们之间十分相似。

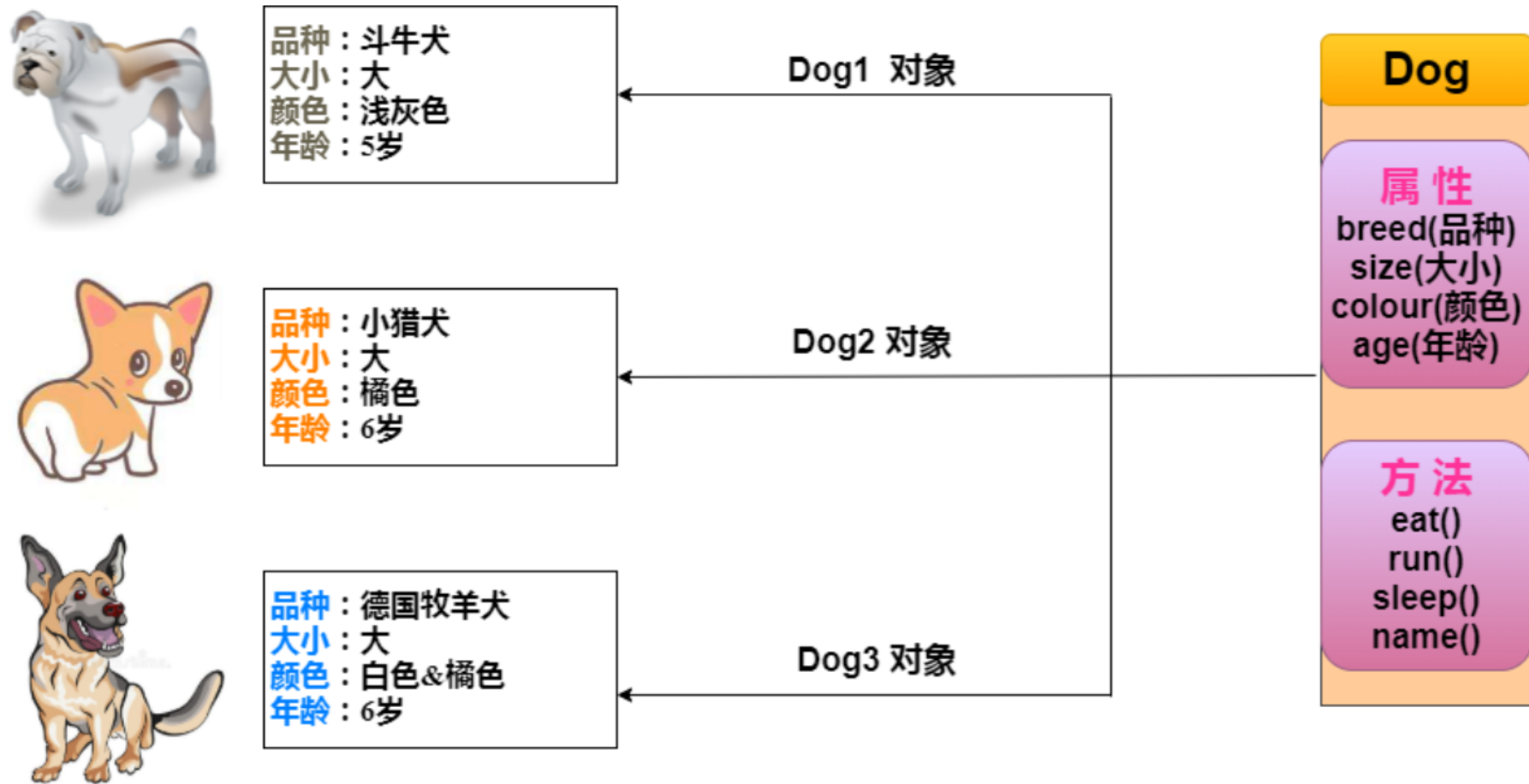
软件对象也有状态和行为。软件对象的状态就是属性，行为通过方法体现。

在软件开发中，方法操作对象内部状态的改变，对象的相互调用也是通过方法来完成。

9.1.1 类与对象

- Java 中的类

类可以看成是创建 Java 对象的模板。



9.1.1 类与对象

- 通过上图创建一个简单的类来理解下 Java 中类的定义：



eg9.1.1.txt



9.1.1 类与对象

- 构造方法

每个类都有构造方法。如果没有显式地为类定义构造方法，Java 编译器将会为该提供一个默认构造方法。

在创建一个对象的时候，至少要调用一个构造方法。构造方法的名称必须与类同名，一个类可以有多个构造方法。

下面是一个构造方法示例：



eg9.1.1-2.txt



9.1.1 类与对象

- 创建对象

对象是根据类创建的。在Java中，使用关键字 `new` 来创建一个新的对象。创建对象需要以下三步：

声明：声明一个对象，包括对象名称和对象类型。

实例化：使用关键字 `new` 来创建一个对象。

初始化：使用 `new` 创建对象时，会调用构造方法初始化对象。

下面是一个创建对象的例子：



eg9.1.1-3.txt



9.1.1 类与对象

- 访问实例变量和方法

通过已创建的对象来访问成员变量和成员方法，如下所示：

```
/* 实例化对象 */
```

```
Object referenceVariable = new Constructor();
```

```
/* 访问类中的变量 */
```

```
referenceVariable.variableName;
```

```
/* 访问类中的方法 */
```

```
referenceVariable.methodName();
```

使用 Object 类型声明变量只能在编译时访问 Object 类中的方法和属性，但在运行时，你可以通过强制类型转换将其转换为特定类型，以便访问特定类型的方法和属性。



9.1.1 类与对象

- 实例

下面的例子展示如何访问实例变量和调用成员方法：



eg9.1.1-4.txt

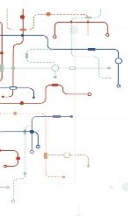
编译并运行上面的程序，产生如下结果：

小狗的名字是：tommy

小狗的年龄为：2

变量值：2





9.1.1 类与对象

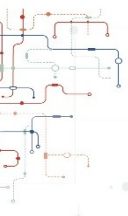
- 源文件声明规则

当在一个源文件中定义多个类，并且还有import语句和package语句时，要特别注意这些规则。

- 1 一个源文件中只能有一个 public 类
- 2 一个源文件可以有多个非 public 类
- 3 源文件的名称应该和 public 类的类名保持一致。例如：源文件中 public 类的类名是 Employee，那么源文件应该命名为Employee.java。
- 4 如果一个类定义在某个包中，那么 package 语句应该在源文件的首行。
- 5 如果源文件包含 import 语句，那么应该放在 package 语句和类定义之间。如果没有 package 语句，那么 import 语句应该在源文件中最前面。
- 6 import 语句和 package 语句对源文件中定义的所有类都有效。在同一源文件中，不能给不同的类不同的包声明。

类有若干种访问级别，并且类也分不同的类型：抽象类和 final 类等。

除了上面提到的几种类型，Java 还有一些特殊的类，如：内部类、匿名类。



9.1.1 类与对象

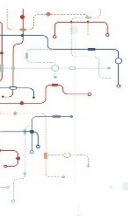
- 一个类可以包含以下类型变量：

局部变量：在方法、构造方法或者语句块中定义的变量被称为**局部变量**。变量声明和初始化都是在方法中，方法结束后，变量就会自动销毁。

成员变量：成员变量是**定义在类中，方法体之外的变量**。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。

类变量：类变量也声明在类中，方法体之外，**但必须声明为 static 类型**。

一个类可以拥有多个方法，在上面的例子中：eat()、run()、sleep() 和 name() 都是 Dog 类的方法。



9.1.1 类与对象

- Java 包

包主要用来对类和接口进行分类。当开发 Java 程序时，可能编写成百上千的类，因此很有必要对类和接口进行分类。

- import 语句

在 Java 中，如果给出一个**完整的限定名**，包括包名、类名，那么 Java 编译器就可以很容易地定位到源代码或者类。import 语句就是用来提供一个合理的路径，使得编译器可以找到某个类。

例如，下面的命令行将会命令编译器载入
java_installation/java/io 路径下的所有类

```
import java.io.*;
```

9.1.1 类与对象

- 一个简单的例子

在该例子中，我们创建两个类：Employee 和 EmployeeTest。

首先打开代码编辑器，把下面的代码粘贴进去，将文件保存为 Employee.java。

Employee 类有四个成员变量：name、age、designation 和 salary。该类显式声明了一个构造方法，该方法只有一个参数。



eg9.1.1-5.txt



9.1.1 类与对象

Java 程序都是从 main 方法开始执行，为了能运行这个程序，必须包含 main 方法并且创建一个实例对象。

下面给出 EmployeeTest 类，该类实例化 2 个 Employee 类的实例，并调用方法设置变量的值。

将下面的代码保存在 EmployeeTest.java 文件中。



eg9.1.1-6.txt





9.1.1 类与对象

编译这两个文件并且运行 EmployeeTest 类，可以看到如下结果：

```
javac EmployeeTest.java
```

```
java EmployeeTest
```

```
名字:RUNOOB1
```

```
年龄:26
```

```
职位:高级程序员
```

```
薪水:1000.0
```

```
名字:RUNOOB2
```

```
年龄:21
```

```
职位:菜鸟程序员
```

```
薪水:500.0
```

9.1.1 类与对象

- 我们再来回顾一下：

类：在Java中，类是由属性（变量）和方法（函数）组成的。属性代表对象的状态，方法代表对象的行为。

对象：对象是根据类创建的具体实例，每个对象都有独立的属性值和共享的方法。



9.1.2 面向对象程序的架构

- 概述

面向对象程序的架构是以类和对象为核心，通过**封装、继承、多态**等特性来组织代码，旨在提高代码的复用性、可维护性和扩展性。



9.1.2 面向对象程序的架构

- 详细说明：

封装：将对象的状态（属性）和行为（方法）封装在类中，并提供对外的接口访问。

继承：子类继承父类的属性和方法，实现代码复用。

多态：相同接口的不同实现，提高系统的灵活性。



9.1.2 面向对象程序的架构

- 生活举例：

面向对象架构：就像一家大公司，部门是类，员工是对象，部门管理自己的员工，员工可以调动到其他部门（继承），不同部门可以用同样的方式管理员工（多态）。



9.2 功能驱动的设计方法

- 以一个实例阐述如何使用面向对象的方式来设计一个简单的用户管理系统。它包括了用户类、用户存储库、用户服务类以及用于测试这些类的JUnit测试类。这个例子旨在展示如何组织代码以便于维护和测试。



eg9.2.txt



9.2.1 功能驱动的系统架构设计方法

- 概述

功能驱动的系统架构设计方法关注系统需要实现的功能，通过将功能分解为子功能，逐步设计类和对象。



9.2.1 功能驱动的系统架构设计方法

- 详细说明：

功能分解：将系统的功能需求分解为独立的模块或组件。

类的划分：根据功能划分出不同的类，每个类负责实现一个或多个功能。



9.2.1 功能驱动的系统架构设计方法

- 生活举例：

功能驱动设计：像建造一栋楼，首先设计楼的功能（如住宅、商铺），然后再设计每层楼的结构。

以一个实例阐述：



eg9.2.1.txt

在这个示例中，我们定义了一个Feature接口，并创建了两个实现类FeatureA和FeatureB。SystemArchitecture类接收一个Feature列表，并可以遍历这个列表来执行所有功能。在Main类的main方法中，我们创建了一个featureList，包含了FeatureA和FeatureB的实例，然后构造了SystemArchitecture实例并执行了所有功能。这样的设计方法可以灵活应对不同的功能需求，增加或者移除功能只需要修改featureList即可。



9.2.2 功能驱动的类型设计方法

- 概述：

根据系统功能设计类，通过分析每个功能所需的行为和数据，将其封装为类的方法和属性。



9.2.2 功能驱动的类型设计方法

- 详细说明：

职责分配：为每个类分配清晰的职责，使得类之间低耦合、高内聚。

类设计原则：遵循单一职责原则（SRP）、开闭原则（OCP）等设计原则。



9.2.2 功能驱动的类型设计方法

- 生活举例：

类设计：像设计一个团队，每个人都有明确的职责分工，互相配合完成任务。

以一个实例来阐述：



eg9.2.2.txt

这个例子展示了一个简单的信用卡类及其使用方法。它包括构造函数、用于获取信息和更新信息的方法，以及用于打印信息的方法。这种类型的设计方法遵循了面向对象编程的原则，使得代码易于理解和维护。



9.3 面向对象程序设计的过程

- 概述：

面向对象程序设计的过程包括从问题分析、系统建模、类设计、到代码实现的完整流程，强调自顶向下逐步细化和自底向上逐步集成。





9.3 面向对象程序设计的过程

- 详细说明：

分析需求：理解系统需求，识别系统中涉及的实体和操作。

建模：使用UML图等工具进行系统建模，确定类和对象的关系。

类设计：根据模型设计类，编写类的规格说明和代码实现。

实现与测试：将类进行实现，集成测试，确保系统功能正确。



9.3 面向对象程序设计的过程

- 生活举例：

面向对象设计过程：就像设计一款新产品，从市场调研、产品设计、制造、到最终的测试和上市。

9.3 面向对象程序设计的过程

- 用一个例子来说明

面向对象程序设计(OOP)是一种程序设计范式，其基于对象的概念来构建和组织应用程序。在Java中，一个对象由一个类来定义，类是一个模板，它描述一个对象的数据和方法。

以下是一个简单的Java类定义的例子，展示了面向对象程序设计的基本步骤：



eg9.3.txt

在这个例子中，Person类定义了两个属性（name和age）和一个构造器，以及一个introduce方法。Main类中的main方法创建了一个Person对象，并调用了该对象的introduce方法。这展示了面向对象的基本概念，包括封装、继承、多态和抽象。

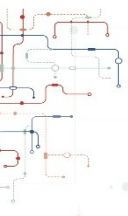


9.4 问题分解描述与类的初步设计

- 概述

问题分解描述是将复杂的问题逐步分解为多个子问题，并为每个子问题设计相应的类。





9.4 问题分解描述与类的初步设计

- 详细说明：

分解问题：将大问题分解为可以单独解决的小问题，每个小问题对应一个或多个类。

初步设计：设计类的属性和方法，明确类之间的关系和交互。

9.4 问题分解描述与类的初步设计

- 生活举例：

问题分解：像写一篇长文，将其分成几个段落，每个段落负责一个主题。

以一个实例来阐述：



eg9.4.txt

这个简化的代码示例展示了如何使用Java中的枚举、类继承和多态等概念来设计一个游戏物品系统。它定义了一个基类GameItem以及几个子类，分别代表不同类型的游戏物品，并且提供了一个使用示例来创建并展示这些物品的信息。这样的设计方式有利于代码的扩展和维护。



9.5 类的详细设计与实现

- 以一个例子来阐述



eg9.5.txt

这个类Patient包含病患的基本信息，并提供了对应的getter和setter方法来访问和修改这些信息。构造函数用于在创建对象时初始化病患信息。在真实的医疗场景中，病患类可能会包含更多复杂的信息和方法，例如病情描述、处方、检查结果等。



9.5.1 类的详细设计

- 概述

在详细设计阶段，为每个类定义具体的属性、方法，以及类之间的关系，确保设计符合需求和设计原则。

举个例子：



eg9.5.1.txt

这个类MedicalService用于描述基本的医疗服务信息，包括服务ID、名称、类型、描述和价格。构造函数用于在创建对象时初始化这些属性。getter和setter方法被省略以保持代码简洁，实际应用中需要根据需要实现。





9.5.1 类的详细设计

- 详细说明：

属性设计：为类定义所需的属性，并决定其可见性（private、protected、public）。

方法设计：定义方法的参数、返回值和逻辑，确保方法职责单一，易于维护。

关系设计：确定类之间的继承、聚合、关联关系。

9.5.1 类的详细设计

- 生活举例：

类设计：像绘制建筑设计图纸，细化每个房间的布局和功能。

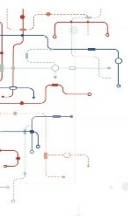


9.5.2 类的规格说明与编码实现

- 概述：

类的规格说明详细描述了类的功能、属性和方法的定义，是编码实现的依据。





9.5.2 类的规格说明与编码实现

- 详细说明：

规格说明：包括类的描述、属性列表、方法列表、关系描述等，作为实现和测试的依据。

编码实现：根据规格说明书，编写类的代码实现，并进行单元测试确保其功能正确。

9.5.2 类的规格说明与编码实现

- 生活举例：

编码实现：像根据设计图纸实际建造房屋，确保每个部分都按照设计要求完成。



- 代码举例：`eg.9.5.2.txt`

- 这个代码示例展示了如何定义一个基础的医疗服务类，并创建一个继承自该基类的检查服务子类。它还展示了如何在main方法中实例化这些类并调用它们的方法，以便理解如何使用这些类和方法。这个例子简单但有效地展示了面向对象编程的基本概念。





本章重点

面向对象程序的架构

功能驱动的设计方法

功能驱动的系统架构设计方法

功能驱动类的设计方法

面向对象程序设计的过程

问题分解描述与类的初步设计

类的详细设计与实现



本章作业

作业一：面向对象程序设计实践

目的：加深理解类与对象的概念，以及面向对象程序的架构和功能驱动的设计方法。

要求：

- 1 设计一个简单的图书管理系统，包含图书（Book）类。图书类应包含属性如书名、作者、出版日期和ISBN号，以及相应的getter和setter方法。
- 2 在图书类的基础上，设计并实现一个图书馆（Library）类，图书馆类应包含添加图书、删除图书、显示所有图书等功能。
- 3 使用面向对象的思想，结合功能驱动的设计方法，考虑图书馆系统的扩展性，比如未来可能需要添加借阅、归还图书等功能，如何在当前设计中预留接口。



本章作业

作业二：功能驱动的系统架构与类设计

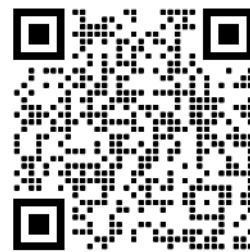
目的：理解功能驱动的系统架构设计方法，并应用于类的详细设计与实现中。

要求：

- 1 选择一个你熟悉或感兴趣的系统（如学生信息管理系统、在线购物系统等），首先进行功能分解描述，列出该系统需要实现的主要功能。
- 2 根据功能分解结果，进行类的初步设计。识别出系统中的主要实体（如学生、商品、订单等），并为每个实体设计相应的类。
- 3 对每个类进行详细设计，包括类的属性、方法（包括构造方法、getter和setter方法、业务逻辑方法等）以及类之间的关系（如继承、组合、关联等）。
- 4 思考并描述如何将这类组织起来形成一个完整的系统架构，以及各个部分如何协同工作以实现系统功能。

信创智能医疗系统研发课程体系

河南中医药大学信息技术学院（智能医疗行业学院）



河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

河南中医药大学医疗健康信息工程技术研究所