**第3课 Java 面向对象特性**

|  |  |  |
| --- | --- | --- |
| **课 题** | Java 面向对象特性 | |
| **课 时** | 14课时（630 min）。 | |
| **教学目标** | **知识技能目标：**  1．掌握面向对象编程的三种特性：封装、继承、多态。  2．掌握接口的定义以及其实现的方法。  **思政育人目标：**  让学生通过学习Java 面向对象特性，编写简单的 Java 类，由浅至深，养成良好的编程习惯和风格。 | |
| **教学重难点** | **教学重点：**类的定义  **教学难点：**创建和使用对象 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第6节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第7节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第8节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第9节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第10节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第11节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第12节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第13节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第14节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**展示类的定义  **任务描述**  定义汽车类，分析汽车类的属性、构造器和方法。  /\* 源程序文件名：Ex0301.java\*/  class Automobile  {  String name;  String id;  String producer;  String type;  String color;  double price;  String owner;  public Automobile(String name,String id,String producer,String  type,String color,double price,String owner)  {  this.name = name;  this.id = id;  this.producer = producer;  this.type = type;  this.color = color;  this.price = price;  this.owner = owner;  System.out.println("Create a "+name);  }  public void drive()  {  System.out.println("The automobile is driving");  }  }  public class Ex0301  {  public static void main(String[] args)  {  System.out.println(" 我刚建立了一个类 ");  }  }  **任务目标**  1. 能够理解面向对象编程的思想。  2. 能够理解类和对象的概念。  3. 能够掌握使用 Java 语言定义类的规则与方法。  4. 能够掌握封装的概念，知道封装就是把数据和方法”保护”起来，树立总体国家安全观，能够对现实对象进行总结、抽象、加工，再描绘出它的模型。  **预备知识**  **一、面向对象程序设计方法概述**  面向对象程序设计方法是相对过去传统的结构化编程方法而言的，这种方法是首先对现实环境进行总结、抽象、加工，再描绘出它的模型，使用对象、类、继承、封装、消息等基本概念来进行程序设计。从现实世界中客观存在的事物（即对象）出发来构造  软件系统，并且在系统构造中尽可能运用人类的自然思维方式。然后用 Java 语言将其表述出。  **二、类的定义**  类是对一组具有共同特性的所有对象成员的抽象描述。就如同“汽车”这个概念一样，它能被看作是一个类，那么我们怎样来描述“汽车”呢？  这些属性是作为程序员的你，为了能在 Java 编程环境下编程而为汽车类所定义的。  除了以上的定义以外，汽车在现实生活中还能做些什么事呢？汽车能做的操作用 Java 的专业术语来表述就是方法。下面我们就假定汽车能做的操作只有行驶这一个，那么我们就可以定义一个方法 drive( )。  **（一）类的概念**  综上所述，我们说类是 Java 中的一种重要的复合数据类型，是组成 Java 程序的基本要素。它封装了一类对象的状态和方法，是这一类对象的原形型。一个类的实现包括两个部分：类声明和类体。  **（二）类的声明**  [public] [abstract|final] class className [extends superclassName]  [implements interfaceNameList]{……}  其中，修饰符 public、abstract、final 说明了类的属性，className 为类名，superclassName 为父类的名字，interfaceNameList 为类所实现的接口列表。  **（三）类的组成**  处于类内部的部分，我们称其为类的成员。组成的这些成员有：成员变量、成员方法、构造函数等。构造函数又叫构造器，成员变量还可叫作属性。  class className  {  [public|protected|private] [static] [final] [transient] [volatile]  type variableName;　　// 成员变量  [public] className([paramList]){statements;}　　// 构造函数  [public|protected|private] [static] [final|abstract] [native]  [synchronized] returnType methodName ([paramList]) [throws exceptionList]  { statements; }　　// 成员方法  }  **三、构造函数**  **（一）构造函数的概念、作用与格式**  构造函数是类中一种非常特殊的方法，这个方法在实例化对象时使用，在编程时主要用于为类中的成员赋值。我们先定义了“汽车”这个概念，这个概念在 Java 中是用类来表达的，现在准备根据这个概念来造一辆真正的汽车，那 Java 中怎样表达“建造”这件事呢？  Automobile auto1 = new Automobile();  这句代码所表达的意思就是依据概念来建造汽车这件事，那它做了什么呢？简单地说就是调用了相关的构造函数。  一般情况下，每个类都会有一个默认的无参的空构造函数，若程序员没编写相关的构造函数，系统会自动地去调用这个默认的无参的空构造函数。若需要自己写这个函数，那就必须遵循下面的规则：  （1）该方法与类同名。  （2）该方法没有返回值，不能使用 void 关键字。  （3）一般情况下其访问属性为 public。  （4）当编写了自己的构造函数后，其原来的默认构造函数就不存在。下面的代码就可以为任务 1 中的 Automobile 类添加一个构造函数：  public Automobile()  {  System.out.println("Create a new automobile");  }  **（二）构造函数的重载**  接着我们来思考，一个类是否只能有一个构造函数？答案当然是否定的。一个类中可以有多个构造函数，这种情况用于成员变量初始化时有多种不同要求时，称为构造函数的重载。  public Automobile()  {  System.out.println("Create a new automobile");  }  public Automobile(String name)  {  this.name = name;  System.out.println("Create a "+name);  }  上述情况中我们有了两个构造函数，那怎么知道 new 的时候调用的是哪个构造函数呢？秘密就在构造函数的变量表。  请在计算机上实验使用这两个构造函数。  **任务实施**  类：定义一个类，类名是 Automobile，其中 class Automobile 是类的声明，其余用{} 括起来的部分是类体。  属性：定义 6 个属性包括 String 类型的 name 属性，String 类型的 id 属性，String 类型的 producer 属性，String 类型的 type 属性，String 类型的 color 属性，double 类型的price 属性，String 类型的 owner 属性。  构造器：定义一个构造器 Automobile（…）{…}，构造器名称与类名一致，用来给 6个属性赋初值，并且输出一个字符串。  方法一个：drive( ){…}，没有参数，输出“The automobile is driving”。  **【学生】**思考、讨论。 | **展示类的定义，让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  简述构造函数的概念、作用与格式。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示创建和使用对象  **任务描述**  下列程序定义汽车类，分析汽车类实例化后的属性、构造器和方法的使用。  /\* 源程序文件名：Ex0302.java\*/  class Automobile  {  String name;  String ID;  String producer;  String color;  double price;  String owner;  public Automobile()  {  System.out.println("Create a new automobile");  }  public Automobile(String name)  {  this.name = name;  System.out.println("Create a "+name);  }  public void drive()  {  System.out.println("The automobile is driving");  }  }  public class Ex0302  {  public static void main(String[] args)  {  Automobile auto1 = new Automobile();  Automobile auto2 = new Automobile("Benz");  auto1.name = "BMW";  System.out.println(auto1.name);  System.out.println(auto2.name);  auto1.drive();  auto2.drive();  }  }  **任务目标**  1. 能够掌握实例化类的方法。  2. 会实例化类，并调用其中的属性和方法。  3. 能够理解把“理想”变为“现实”这一过程，如同推动“马克思主义中国化”，做一个“实干”人。  **预备知识**  **一、创建对象**  **（一）创建句柄**  为什么要创建对象？这可能是每一个刚接触面向对象编程的人都疑惑不解的问题，不要紧，我们现在就来解决它。首先来想象在家看电视的情景，我们使用遥控器来遥控电视换台或开关它，如果想在屋里边走动边控制电视机的频道时，你会怎么做？你会手里拿着遥控器，而不是搬着电视机跟着你走对吗？再回到语句上，下面这条语句就可以看作有两层含义：  Automobile auto1 = new Automobile();  将其分解以后就是这样，  Automobile auto1; // 首先申请一个遥控器  auto1 = new Automobile(); // 使用 auto1 这个遥控器来指挥电视  第一条语句的意思是：你想看 Automobile 牌子的电视了，所以你先在手里拿了一个Automobile 牌子的遥控器。用计算机语言来说就是你向计算机申请了一个句柄，它的名字叫 auto1，准备用它来控制对象。现在你只有遥控器，能看电视吗？当然不行，所以我们必须使用第二条语句。注意虽然不能看电视，但遥控器是能单独存在的，也就是说你可以向计算机申请一个句柄，但它并不与任何对象相关联。  第二条语句的意思是：你将使用名为 auto1 的遥控器与一台电视联系起来，以后这台电视必需使用这个遥控器来遥控，用计算机的语言来说就是你申请了句柄以后，将它与一个特定的对象关联了起来，以后使用这个对象时，就必须使用个句柄。这种情况也等同于你想叫你的一名同学帮你做一件事情时，你会使用他的名字（句柄），而这位同学就是那个特定的对象。  **（二）内存分配**  程序运行时，所有的数据都必须放在内存中，当然大家都知道了。内存中有三种存放数据的块我们需要了解：堆栈、堆、静态存储区。  （1）堆栈：这是一块有规律的内存空间，就像一个被分成很多相同大小的小格子的大柜子。通常用类似的图示来描述它。这种内存分配适合为长度一定的数据分配空间。  （2）堆：一种内存池。主要用于存放长度不固定的数据。我们已经知道对象的长度是不一样的，所以计算机使用它来为对象分配相应的空间。分配的大小是由类的结构来规定的，而分配在堆的空间就是对象。怎样来使用这个对象呢？——句柄。  **二、引用对象的属性**  观察下列代码：  public class Automobile  {  public String name;  String ID;  String producer;  String color;  double price;  String owner;  public void drive()  {  System.out.println("The automobile is driving");  }  }  上述语句中第 3～8 行的定义部分就是类的属性，我们也可以称它们为类的数据成员或者字段。怎样引用这些属性呢？还记得我们先必须制造电视并选一支指向它的遥控器吗？然后你就能使用遥控器来控制它的属性了。  public static void main(String args[])  {  Automobile auto = new Automobile();  auto.name = "Benz";  }  看见我们怎样遥控它的吗？  方法：句柄的名字（对象名）. 属性名。  **三、引用对象的方法**  观察下列代码：  public class Car  {  public String name;  String ID;  String producer;  String color;  double price;  String owner;  public void drive()  {  System.out.println("The automobile is driving");  }  }  上述代码中 public void drive( ){…} 部分就是类的方法，或者也可以叫作成员方法。  怎样调用它呢？方法：句柄 . 方法名 ( )，例如：  auto.drive();  **任务实施**  本程序对类 Automobile 进行了两次实例化，分别生成对象 auto1 和 auto2，实例化语句如下：  Automobile auto1 = new Automobile();  Automobile auto2 = new Automobile("Benz");  访问对象的属性的语句如下：  auto1.name = "BMW";　　// 给对象 auto1 的 name 属性赋值 "BMW"  System.out.println(auto1.name);　　// 输出对象 auto1 的 name 属性  System.out.println(auto2.name);　　// 输出对象 auto2 的 name 属性  访问对象的方法的语句如下：  auto1.drive();　　// 调用对象 auto1 的 drive() 方法  auto2.drive();　　// 调用对象 auto2 的 drive() 方法  **【学生】**思考、讨论。 | **通过教师讲解，了解创建和使用对象的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了创建和使用对象，让学生能够理解把“理想”变为“现实”这一过程，如同推动“马克思主义中国化”，做一个“实干”人。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  简述引用对象的方法。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示static 修饰符  **任务描述**  为上个任务程序添加一个计数器，用来记录创建了多少辆汽车。  **任务目标**  1. 能够理解什么是静态属性与方法。  2. 会使用 static 关键字。  3. 能够理解“非静态”是自己的，“静态”是大家的。  **预备知识**  静态（static）属性与方法又可称作类的属性与方法，顾名思义就是说这种变量与方法只属于类，不属于对象，所以调用它们时是不需要实例化的，那怎样做呢？  **一、定义方法**  怎样告诉计算机这个属性或者这个方法是静态的，是属于类的，不是属于对象的？其实很简单——使用 static 关键字。  静态属性的定义：  public static int number = 0;  静态方法的定义：  public static void count()  {  System.out.println(" 已制造了 "+number+" 辆汽车 ");  }  **二、调用方法**  静态属性：类名 . 属性名。  例如：Automobile.number。  静态方法：类名 . 方法名 ( )。  例如：Automobile.count( )。  另外还可以用对象名来调用，如对象名 . 属性名，对象名 . 方法名 ( )。  **任务实施**  /\* 源程序文件名：Ex0303.java\*/  import java.io.\*;  class Automobile  {  public static int number = 0;  String name;  String producer;  String color;  double price;  String owner;  public Automobile()  {  System.out.println("Create a new automobile");  number++;  }  public void drive()  {  System.out.println("The automobile is driving");  }  public static void count()  {  System.out.println(" 已制造了 "+number+" 辆汽车 ");  }  }  class Ex0303  {  public static void main(String[] args)  {  int i;  Automobile auto[] = new Automobile[10];  for(i = 0;i<10;i++)  {  auto[i] = new Automobile();  }  Automobile.count();  System.out.println(Automobile.  number);  }  }  **【学生】**思考、讨论。 | **教师通过static 修饰符展示，让学生了解static 修饰符的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了static 修饰符，能够理解“非静态”是自己的，“静态”是大家的。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **调用方法有哪些？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示类的封装  **任务描述**  分析下列程序是如何控制用户对成员属性与成员方法的访问权限。  /\* 源程序文件名：Ex0304.java\*/  import java.io.\*;  class Tom  {  public float weight;　　//weight 被修饰为 public 的 float 型变量  private float height;　　//height 被修饰为 private 的 float 型变量  public float f1(float a, float b){……}　　// 方法 f1 是 public 方法  private float f2(float a, float b){……}　　// 方法 f2 是 private 方法  ……  }  class Jerry  {  void g()  {  Tom cat = new Tom();  cat.weight = 23f; // 合法  cat.height = 1.2; // 非法  cat.f1(3,4); // 合法  cat.f2(1,2); // 非法  }  }  public class Ex0304  {  public static void main(String[] args)  {  System.out.println(" 这是关于访问修饰符的实验 ");  }  }  **任务目标**  1. 能够正确理解面向对象编程中类封装的思想。  2. 能够掌握 public、protected、default、private 关键字的意义。  3. 能够掌握上述三个关键字的访问区域。  4. 会保护自己的数据，我的“地盘“我做主，树立总体国家安全观。  **预备知识**  **一、封装**  什么是封装？现在我们又回到前面看电视这个例子上去。试着回忆你看电视时是怎样操作的。对了，电视机或者遥控器上有按钮，若想要调整电视音量的大小，我们会在电视机或者遥控器上找到一个标有音量的按钮按下它则可，不需要知道这个按钮具体是怎样工作的，到底有哪些电子零件起了作用，这就是封装。在程序中我们只需要提供给用户方法的接口，而不需要告诉他们这个方法里具体的内容是怎样，它到底是怎样进行工作的。  **二、修饰符**  **（一）public 公共的**  public 修饰的成员变量和方法被称为“公共变量”和“公共方法”。例如：  class Tom  {  public float weight; //weight 被修饰为 public 的 float 型变量  public float f(float a, float b){……} // 方法 f 是 public 方法  ……  }  当我们在另外一个类中用类 Tom 创建了一个对象后，该对象能访问自己的 public属性和 public 方法。例如：  class Jerry  {  void g()  {  Tom cat = new Tom();  cat.weight = 23f; // 合法  cat.f(3,4); // 合法  }  }  public 修饰的方法和属性可以被所有类的对象使用。  **（二）protected 受保护的**  用 protected 修饰的成员变量或方法可以被该类自身、同一个包中的其他类以及在其他包中的该类的子类访问，例如：  class cat  {  protected void eat(){……}  }  public class tom extends cat  {  public static void main(String args[])  {  tom t = new tom();  t.eat(); // 合法  }  }  **（三）default 默认的**  不用修饰符的成员变量和方法被称为默认变量和默认方法。例如：  class Tom  {  float weight; // 变量 weight 是默认的 float 型变量  float f(float a,float b){……} // 方法 f 是默认的方法  ……  }  当我们在另外一个类中用类 Tom 创建了一个对象后，如果这个类与类 Tom 在同一个包中，那么该对象能访问自己的默认变量和默认方法。假如类 Jerry 与 Tom 在同一个包中，例如：  class Jerry  {  void g()  {  Tom cat = new Tom();  cat.weight = 23f; // 合法  cat.f(3,4); // 合法  }  }  需要注意的是：  （1）如果你在源文件中用 import 语句引入了另外一个包中的类，并用该类创建了一个对象，那么该类的这个对象将不能访问自己的默认变量和默认方法；  （2）如果一个类被修饰为 public 的，例如：  public class A{……}  那么你可以在任何另外一个类中，使用该类创建对象。如果一个类不加任何修饰，例如：  class A{……}  那么你在另外一个类中使用这个类创建对象时，要保证它们是在同一个包中。  **（四）private 私有的**  用 private 修饰的成员变量和方法被称为私有变量和私有方法。例如：  class Tom  {  private float weight; //weight 被修饰为私有的 float 型变量  private float f(float a, float b){……} // 方法 f 是私有方法  ……  }  当我们在另外一个类中用类 Tom 创建了一个对象后，该对象不能访问自己的私有变量和私有方法。例如：  class Jerry  {  void g()  {  Tom cat = new Tom();  cat.weight = 23f; // 非法  cat.f(3f,4f); // 非法  ……  }  }  private 修饰的成员变量和方法只能被同一类中的其他成员访问，不能被其他类的成员访问，也不能被其子类成员访问。但是若访问这个方法的对象位于包外或者不是该类的继承者，则访问就是非法的。  **（五）访问限定符小结**  变量和方法可以处于四个访问级别的一个：public、protected、default 或 private。类可以是 public 或 default。  变量、方法或类有缺省访问性，如果它没有显式受保护修饰符作为它的声明的一部分的话。这种访问性意味着访问可以来自任何方法，当然这些方法只能在作为目标的同一个包中的成员类当中。以修饰符 protected 标记的变量或方法实际上比以缺省访问控制标记的更易访问。一个 protected 方法或变量可以从类当中的任何方法进行访问，这个类可以是同一个包中的成员，也可以是从任何子类中的任何方法进行访问。当它适合于一个类的子类但不是不相关的类时，就可以使用这种受保护访问来访问成员。  **任务实施**  在本例中，我们学习了关于 Java 编程语言中访问修饰符的使用。当用一个类创建了一个对象之后，该对象可以通过“.”运算符访问自己的变量和方法。但访问自己的变量和方法有一定的限制，通过修饰符 private 和 public 来说明访问权限。关于各种访问修饰符的特点和具体用法，请大家参考前面的学习。  **【学生】**思考、讨论。 | **教师通过类的封装展示，让学生了解类的封装的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了类的封装，会保护自己的数据，我的“地盘“我做主，树立总体国家安全观。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述访问限定符小结。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示继承  **任务描述**  编写程序：在定义汽车类的基础上，定义出其子类公共汽车类。  **任务目标**  1. 能够正确理解继承的含义与作用。  2. 能够在 Java 中使用类的继承。  3. 能够使用 Java 中继承的属性和方法。  4. 能够理解继承不仅是“抄袭”，还要有“创新”，做一个善于“学习”，勇于“创新”的人。  **预备知识**  在 Java 中，继承是一种由已有的类创建新类的机制。利用继承机制，我们可以先创建一个共有属性的一般类，根据该一般类再创建具有特殊属性的新类。新类继承一般类的状态和行为，并根据需要增加它自己的新的状态和行为。由继承而得到的类称为子类，被继承的类称为父类（超类或基类）。Java 不支持多重继承（子类只能有一个父类）。  **一、创建子类**  在类的声明中，通过使用关键字 extends 来创建一个类的子类，格式如下：  class Bus extends Automobile{……}  把 Bus 类声明为 Automobile 类的子类，Automobile 类是 Bus 类的父类。  **二、子类的继承性**  子类继承了其父类中除 private 属性外的成员变量和成员方法作为自己的成员变量和成员方法。注意，如果一个类的声明中没有使用 extends 关键字，则这个类被系统默认为是 Object 的子类。Object 是包 java.lang 中的类，是所有类的父类。  **三、特殊的父类**  我们知道类也受访问限定符的控制，对于不是定义为 public 的类，由于只能在包中被访问，因此这种类型也只能被包中的子类继承。包外的子类不可继承。当一个类被关键字 final 所修饰时，它表示这是这个类是最终类，不能再被继承，所以如果你写的类不想被其他类继承，就可以给它加上 final 的修饰符。  **四、this 和 super 关键字**  1. this  在 Java 中，this 是非常重要的关键字，它可以被看作一个指针，用来指向调用方法或成员变量的对象本身，因此针对静态的方法或变量，是没有 this 指针指向它的。观察下面的代码：  class Test  {  int x;  int y;  public Test(int x, int y)  {  this.x = x;  this.y = y;  }  public void f(Object o){……}  public static void main(String[] args)  {  Test t = new Test(3,4);  t.f(t);  }  }  在这个例子中，使用了构造函数给成员变量赋值，但形式参数的名称与成员变量的名字是一样的。为了以示区别，我们在调用成员变量使用了关键字 this，那这个指针指向谁呢？简而言之，谁调用就指向谁。因此 this 指针在例子中就指向了对象 t，所以我们是给 t 对象中 x，y 赋值。this 的另一种用法：this( ) 方法可以调用同一个类中不同的构造器。  2. super  在 Java 中，super 关键字有两种作用。  （1）用在子类中引用其父类的构造函数。其格式为：  super([ 实参列表 ]);  其中的 [ 实参列表 ] 对应于父类的构造函数列表 , 例如：父类：  public class Animal  {  String name = " 动物 ";  public void eat(String food)  {  System.out.println(" 饿了 , 吃东西 ");  }  }  子类：  public class Cat extends Animal  {  public Cat()  {  super(" 猫 ");  }  }  （2）用来调用父类的成员变量和成员方法。  super 可以用作前缀来引用父类中的成员。例如，上例中 Animal 类与 Cat 类中都有eat( ) 这个方法，当想在子类中引用父类的 eat 方法时，以示区别，必须使用 super.eat( )来引用父类中的 eat( ) 方法。  子类：  public class Cat extends Animal  {  String name = " 猫 ";  public void eat()  {  System.out.println(" 喂我老鼠 ");  }  public void test()  {  System.out.println(name); // 输出自己的 name, 猫  System.out.println(super.name); // 输出父类的 name, 动物  eat(); // 调用自己的 eat() 方法 , 输出 " 喂我老鼠 "  super.eat(); // 调用父类的 eat() 方法 , 输出 " 饿了 , 吃东西 "  }  }  **任务实施**  /\* 源程序文件名：Ex0305.java\*/  class Automobile  {  protected String name;  protected String ID;  protected String producer;  protected String color;  protected int price;  protected String owner;  public Automobile()  {  System.out.println("Create a new automobile");  }  public Automobile(String name, String ID, String producer, String color,  int price, String owner)  {  this.name = name;  this.ID = ID;  this.producer = producer;  this.color = color;  this.price = price;  this.owner = owner;  }  public void drive()  {  System.out.println("The automobile is driving");  }  }  class Bus extends Automobile  {  private String type; // 公共汽车的类型  private int number; // 多少路公共汽车  private int money;  public Bus(String name, String ID, String producer, String color, int  price, String owner, String type, int number, int money)  {  super(name, ID, producer, color, price, owner);  this.type = type;  this.number = number;  }  public int sell(int people)  {  this.money = 1\*people;  return this.money;  }  }  public class Ex0305  {  public static void main(String[] args)  {  Bus b = new Bus(" 公交车 ", "3P550", " 中国公交 ", " 绿色 ", 100000,  " 中国公交 ", " 双层柴油 ", 76, 100);  b.drive();  System.out.println(" 共卖了 "+b.sell(10)+ " 钱 ");  }  }  **【学生】**思考、讨论。 | **教师通过继承展示，让学生了解继承的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了继承，能够理解继承不仅是“抄袭”，还要有“创新”，做一个善于“学习”，勇于“创新”的人。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述this 和 super 关键字。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示抽象类  **任务描述**  定义一个抽象类及成员方法，再定义一个该抽象类的子类来实现抽象类中方法。  **任务目标**  1. 能够理解抽象类的含义与作用。  2. 能够使用抽象类。  3. 具备“无规矩不成方圆”的意识，做一个既“循规蹈矩”又“勇于创新”的人。  **预备知识**  **一、抽象类的作用**  有时在开发一些规模较大的程序系统时，会为其建立一些用户自定义的库。在自定义的库开发中，要创建一个体现某些基本行为的类，并为该类声明方法，即告诉其他程序员这个方法的名称、参数表，但并不在这里实现这个方法，取而代之，在子类中实现该方法。在子类中实现时必须按照父类中的声明方法来声明，并实现它，这就对方法的命名及参数等作了规定，对程序进行了规范。  **二、定义格式**  （1）抽象类的定义格式：  ［public］abstract class 类名 [extends 父类名 ][implements 接口名 ]{}  （2）抽象方法的定义的格式：  ［访问修饰符］abstract < 方法返回类型 > 方法名 ([ 形式参数表 ])  例如，考虑一个 Drawing 类。该类包含用于各种绘图设备的方法，但这些必须以独立平台的方法实现。它不可能去访问机器的录像硬件而且还必须是独立于平台的。其意图是绘图类定义哪种方法应该存在，但实际上，由特殊的从属于平台的子类去实现这个行为。  正如 Drawing 类这样的类，它声明方法的存在而不是实现，以及带有对已知行为的方法的实现，这样的类通常被称作抽象类。通过用关键字 abstract 进行标记声明一个抽象类。被声明但没有实现的方法（即这些方法没有程序体），也必须标记为抽象方法。  public abstract class Drawing  {  public abstract void drawDot(int x, int y);  public void drawLine(int x1, int y1, int x2, int y2){}  }  我们不能创建 abstract 类的实例，即不能通过 new 的方法来创建抽象类的对象。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。注意在Java 中是没有抽象构造函数与抽象静态方法的。  抽象方法是用 abstract 修饰的方法。只有方法的定义，没有方法的实现。  **三、关于子类**  所有继承于 abstract 类的子类，必须实现它们父类中的所有抽象方法，否则这些子类也是抽象类。  public class MachineDrawing extends Drawing  {  public void drawDot(int machx, int machy){} // 在此添加代码 , 实现方法  }  Drawing d = new MachineDrawing( ); 一句声明为抽象类对象，但实例化时实例的是抽象类的子类，这是多态特性的体现。这在后面会讲到。  **任务实施**  /\* 源程序文 Ex0306\*/  abstract class Drawing // 定义抽象类 Drawing  {  public abstract void drawDot(int x, int y);  public void drawLine(int x1, int y1, int x2, int y2)  {  System.out.print("this is in the abstract class Drawing");  }  }  class MachineDrawing extends Drawing  {  public void drawDot (int x, int y) // 在此添加代码 , 实现方法  {  System.out.println("Drawing d dot"+x+" "+y);  }  }  public class Ex0306  {  public static void main(String [] args)  {  Drawing d = new MachineDrawing();  d.drawDot(1,2);  d.drawLine(1,2,3,4);  }  }  **【学生】**思考、讨论。 | **教师通过抽象类展示，让学生了解抽象类的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了抽象类，具备“无规矩不成方圆”的意识，做一个既“循规蹈矩”又“勇于创新”的人。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述抽象类的作用。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示多态  **任务描述**  在编写程序时，有时我们会希望一种方法有相同的名字，它会针对不同的情况做出不同的反应，那么怎样才能写这么聪明的程序呢？请看以下例子：  /\* 源程序文件名 Ex0307\*/  class A  {  void callMe()  {  System.out.println("Inside A's callMe() method");  }  }  class B extends A  {  void callMe()  {  System.out.println("Inside B's callMe() Method");  }  void overLoad(String s)  {  System.out.println(s);  }  void overLoad()  {  System.out.println("This is a overLoad test");  }  }  public class Ex0307  {  public static void main(String args[])  {  A a = new B();  a.callMe();  B b = new B();  b.overLoad("Hello");  b.overLoad();  }  }  **任务目标**  1. 能够掌握 Java 实现多态的方法。  2. 能够理解方法的覆盖，掌握覆盖的使用。  3. 能够理解方法的重载，掌握重载的使用。  4. 做一个“见风使舵，善于察言观色”的人。  **预备知识**  **一、多态**  对象的多态性是指在父类中定义的属性或方法被子类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或方法在父类及其各个子类中具有不同的语义。例如：“几何图形”中的“绘图”方法，“椭圆”和“多边形”都是“几何图形”的子类，其“绘图”方法功能不同。  在 Java 语言中，多态性体现在两个方面：由方法重载实现的静态多态性（编译时多态）和方法覆盖实现的动态多态性（运行时多态）。  **（一）编译时多态**  在编译阶段，具体调用哪个被重载的方法，编译器会根据参数的不同来静态确定调用相应的方法。  **（二）运行时多态**  由于子类继承了父类所有的属性（私有的除外），所以子类对象可以作为父类对象使用。程序中凡是使用父类对象的地方，都可以用子类对象来代替。一个对象可以通过引用子类的实例来调用子类的方法。  **二、方法覆盖**  方法覆盖，其实就是运行时多态。子类中的方法与它父类中的方法有相同的方法名、参数列表和返回值类型，称子类中的方法覆盖（override）父类中的方法。从子类中调用覆盖方法时，它总是引用子类定义的方法，而父类中定义的方法将被隐藏。  方法覆盖时应遵循的原则：  （1）发生在父类和子类的同名方法之间；  （2）两个方法的返回值类型必须相同；  （3）两个方法的参数类型、参数个数、参数顺序必须相同；  （4）子类方法的权限必须不小于父类方法的权限；  （5）子类方法只能抛出父类方法声明抛出的异常或异常子类（后续会讲解）。  覆盖方法的调用原则：Java 运行时系统根据调用该方法的实例，来决定调用哪个方法。对子类的一个实例，如果子类覆盖了父类的方法，则运行时系统调用子类的方法；  如果子类继承了父类的方法（未覆盖），则运行时系统调用父类的方法。注意，如果不希望子类覆盖父类的某个方法，则将父类的该方法用 final 修饰。  按照下面的类图 3−7 设计类并测试运行。注意其中的继承关系和方法之间的覆盖。下面这段代码实现了父类 Shape 和一个子类 Rectangle，它的运行过程和结果请大家自行分析。其他两个子类的实现由读者自行完成。  提示：三角形面积的计算，设 s=（a+b+c）/2，则面积 =Math.sqrt（s\*（s-a）\*（s-b）\*  （s-c））    class Shape  {  double area;  double per;  String color;  double getArea()  {  return 0;  }  double getPer()  {  return 0;  }  String getColor()  {  return "";  }  void showAll()  {  System.out.println(getArea());  System.out.println(getPer());  System.out.println(getColor());  }  }  public class Rectangle extends Shape  {  int a;  int b;  double getArea()  {  return a\*b;  }  double getPer()  {  return 2\*(a+b);  }  String getColor()  {  return "red";  }  public static void main(String args[])  {  Rectangle rt = new Rectangle();  rt.a = 4;  rt.b = 5;  rt.showAll();  }  }  **三、方法重载**  方法重载是指同一个类中多个方法享有相同的名字，但是这些方法的参数必须不同（或者是参数的个数不同，或者是参数类型不同）。返回类型不能用来区分重载的方法。参数类型的区分度一定要足够，例如不能是同一简单类型的参数，如 int 与 long。编译器会根据参数的个数和类型来决定当前所使用的方法。例如：  import java.io.\*;  class MethodOverloading  {  void receive(int i)  {  System.out.println("Receive one int data");  System.out.println("i = "+i);  }  void receive(int x, int y)  {  System.out.println("Receive two int datas");  System.out.println("x = "+x+" y = "+y);  }  }  public class MethodOverloadingTest  {  public static void main(String args[])  {  MethodOverloading mo = new MethodOverloading();  mo.receive(1);  mo.receive(2,3);  }  }  **任务实施**  在本任务中，创建了两个类 A 与 B，其中类 B 是类 A 的子类，在类 B 中创建一个方法 callMe( ) 来覆盖类 A 中的 callMe( ) 方法，这是方法的覆盖。同时又在类 B 中另写了一个方法 overLoad（String），它与另一个同名方法 overload( ) 的区别在于它们的参数表不同，这就是方法的重载。  **【学生】**思考、讨论。 | **教师通过多态展示，让学生了解多态的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了多态，让学生做一个“见风使舵，善于察言观色”的人。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **有如下程序：**  **public class Overload**  **{**  **void test()**  **{**  **System.out.println("No parameters");**  **}**  **void test(int a,int b)**  **{**  **System.out.println("a and b: " + a + " " + b);**  **}**  **void test(double a)**  **{**  **System.out.println("Inside test(double) a: " + a);**  **}**  **}**  **public class OverloadDemo**  **{**  **public static void main(String args[])**  **{**  **Overload ob = new Overload();**  **int i = 88;**  **ob.test();**  **ob.test(10,20);**  **ob.test(i);**  **ob.test(123.2);**  **}**  **}**  **其中，ob.test（i）和 ob.test（123.2）分别调用哪个方法？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示接口  **任务描述**  定义售票接口，并在 Bus 类中实现。  **任务目标**  1. 能够掌握接口的定义及接口的使用方法。  2. 能够理解接口和抽象类的区别。  3. 理解接口和抽象类的本质，对不按“规定“写的代码“零容忍“。  **预备知识**  当编程任务较大时，会由多人对程序进行编写，那怎样来规范程序文件中的格式呢？如同一种功能的方法由多人编写时，可能各人会有各人的声明方式，这样的话不仅浪费时间与资源，还会对程序的运行造成不小的麻烦。而且，在 Java 中继承是单继承，如果你的类想继承多个类时又怎么办呢？怎样解决这些问题呢？  Java 不支持多继承性，即一个类只能有一个父类。单继承性使得 Java 简单，易于管理程序。为了克服单继承的缺点，Java 使用了接口，一个类可以实现多个接口。接口是抽象类的变体。接口中的所有方法都是抽象的，没有一个有程序体。接口只可以定义static final 类型的成员变量。  **一、接口的定义**  接口是一种标准，一些规范。满足相同接口所制定出来的标准和规范的产品之间可以相互沟通，便于程序之间互相访问。  **二、定义格式**  使用关键字 interface 来定义一个接口。接口的定义和类的定义很相似，分为接口的声明和接口体。  [public] interface 接口名 [extends 父辈接口 ]  {  常量定义 ;  方法声明 ;  }  例如：  public interface IInterTest1  {  public static final int typeone = 1;  public static final int typetwo = 2;  public int getTest1();  }  public interface IInterTest2  {  public int getTest2();  }  **三、接口实现**  接口只定义了一些常量和抽象方法，因此接口需要由具体的类予以实现。  格式：  class 类名［extends 父类名］implements 接口列表名  例如：  public class Test implements IInterTest1,IInterTest2  {  public int getTest1()  {  System.out.println("This is test one");  }  public int getTest2()  {  System.out.println("This is test two");  }  }  如果实现接口的类定义为非抽象类，就必须实现接口中声明的所有抽象方法。如果有一个接口中的抽象方法没有被实现，则实现接口的类应声明为 abstract 类（抽象类）。  如果想知道一个类是否实现了某接口，可以使用 instanceof 运算符来确定。instanceof 严格来说是 Java 中的一个双目运算符，用法为：  boolean result = obj instanceof Class  其中，obj 为一个对象，Class 表示一个类或者一个接口。当 obj 为 Class 的对象，或者是其直接或间接子类，或者是其接口的实现类，结果 result 都返回 true，否则返回 false。  **任务实施**  /\* 源程序文 Ex0308\*/  public interface ISellTicket  {  public static final double price = 1; // 成人普通票单价  public static final double halfPrice = 0.5; // 儿童票半价  public double sell(int adult ,int children);  }  class BusImpl implements ISellTicket  {  public double sell(int adult, int children)  {  final double money;  money = adult\*price+children\*halfPrice;  return money;  }  }  public class Ex0308  {  public static void main(String[]args)  {  BusImpl b = new BusImpl();  double money;  money = b.sell(3,2);  System.out.println(" 共需要 "+money+" 块钱 ");  }  }  在本任务中，我们建立了一个接口，并使用类 BusImpl 实现了接口，以及接口中的方法，并完成相应功能。实现接口的关键字是 implements。  **【学生】**思考、讨论。 | **教师通过接口展示，让学生了解接口的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了接口，理解接口和抽象类的本质，对不按“规定“写的代码“零容忍“。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述接口的定义。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示包（一）  **任务描述**  将自己定义的类打包，并在另一个类定义时引入。  **任务目标**  1. 能够理解包的概念。  2. 能够掌握 Java 打包与引用包的方法。  3. 打包便于类的管理，在今后的学习和工作中学会归纳管理“知识和事务“。  **预备知识**  当编程任务较大时，会使用较多的类文件与接口，或者有一些功能方法需要被重复使用。这时，就需要对程序文件进行打包与整理。  **一、包的定义**  **（一）包的作用**  包是一种分组与组织的机制，用户可以将相关的接口、类等文件组织在包中，以方便 Java 程序的设计。同时包还使编译单元具有唯一命名的机制，程序员把各编译单元组织在包中可确保包中的编译单元与其他包中的编译单元不产生命名冲突。因为包中的类名都由包名来限定，在程序中引用其他包中的类时，必须通过“import”关键字，将指定包中的类载入。  **（二）定义格式**  package 包名 1[. 包名 2][. 包名 3][… …包名 n]需要注意的是：  （1）该语句必须位于程序的第一行，之前只允许出现注释或空行；  （2）包是可以嵌套的，其间用“.”分隔；  （3）包名用合法标识符表示。  **（三）包的定义**  程序在编译时 Java 解释器的工作过程如下：首先，它找到环境变量 CLASSPATH，它的值将作为寻找包中类文件的根；其次，通过这个根，它会去寻找对应的包名，并将每个“.”翻译为路径中的斜杠，从而产生路径，找到包或类文件的路径。例如：  package com.ada.util;  public class Vector  {  public Vector()  {  System.out.println("com.ada.util.Vector");  }  }  “package com.ada.util；”语句中的符号“.”代表了目录分隔符，程序在编译时创建  了三个文件夹，在当前文件夹下创建的第一个文件夹是 com，第二个是 com 下的子文件  夹 ada，第三个是 ada 下的文件夹 util，当前文件编译后生成的所有字节码文件（.class 文  件）就存放在 util 这个文件夹中。  注意：包就是文件夹。  **二、包的引用**  **（一）作用**  每个包定义好，就成为一个可重复使用的程序单元，它具有一定的功能。当我们要  使用它时就必须要使用引用语句来将其引入到自己的程序中来。  **（二）格式**  import 包名 1[. 包名 2][. 包名 3][… …包名 n].\*;  import 包名 1[. 包名 2][. 包名 3][… …包名 n]. 类名 ;  其中，第一种格式是将包中所有的文件引入到程序中，第二种格式是将包中指定的  类引入到程序中，引入时仍按照打包时的方法确认路径。  **三、常见的包**  java.lang——包含一些 Java 语言的核心类，如 String、Math、Integer、System 和  Thread，提供常用功能。  java.awt——包含了构成抽象窗口工具集（abstract window toolkits）的多个类，这些  类被用来构建和管理应用程序的图形用户界面（GUI）。  java.applet——包含 applet 运行所需的一些类。  java.net——包含执行与网络相关的操作的类。  java.io——包含能提供多种输入 / 输出功能的类。  java.util——包含一些实用工具类，如定义系统特性、使用与日期日历相关的函数。  **【学生】**思考、讨论。 | **教师通过包（一）展示，让学生了解包的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了包（一），打包便于类的管理，在今后的学习和工作中学会归纳管理“知识和事务“。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **包的作用有哪些？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示包（二）  **四、常用的类**  **（一）Math 类**  Math 类来自于 java.lang 包，是数学操作类，提供了一系列用于数学运算的静态方法，包括三角函数、绝对值、幂等。Math 类中有两个静态常量 PI 和 E，分别代表数学中的常量 π 和 e。由于 Math 类比较简单，因此初学者可以通过查看 API 文档来学习Math 类的具体用法。Math 类的常用静态方法如表 3−3 所示。    **（二）Random 类**  Random 类来自于 java.util 包，它可以在指定的范围内随机产生数字。Random 类中  包括了两个构造方法，具体如表 3−4 所示。    表中列举了 Random 类的两个构造方法，其中通过第一个构造方法创建的实例对象  每次使用的种子是随机的，所以每个对象所产生的随机数是不同的。  **任务实施**  /\* 源程序文件名：Ex0309.java\*/  import java.lang.Math;  public class Ex0309  {  public static void main(String[] args)  {  System.out.println(Math.PI);  }  }  **【学生】**思考、讨论。 | **教师通过包（二）展示，让学生了解包（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了包（二），让学生建立了一个名为 util 的包，并在另一个类中引用了这个包中的Math 类，完成了一个简单的打包与引用包的过程。。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述Math 类。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示知识拓展（一）  一、ArrayList 类  ArrayList 类是一个可以动态修改的数组，与普通数组的区别就是它没有固定大小的限制，可以添加或删除元素。ArrayList 继承了 AbstractList 类，并实现了 List 接口。  ArrayList 类位于 java.util 包中，使用前需要引入它，语法格式如下：  import java.util.ArrayList; // 引入 ArrayList 类  ArrayList<E> objectName = new ArrayList<>(); // 初始化  E: 泛型数据类型，用于设置 objectName 的数据类型，只能为引用数据类型。  objectName: 对象名。  ArrayList 是一个数组队列，提供了相关的添加、删除、修改、遍历等功能。  **（一）添加元素**  ArrayList 类提供了很多有用的方法，添加元素到 ArrayList 可以使用 add( ) 方法 :  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  System.out.println(sites);  }  }  输出结果：[Google, Runoob, Taobao, Weibo]。  **（二）修改元素**  如果要修改 ArrayList 中的元素可以使用 set( ) 方法：  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  sites.set(2, "Wiki"); // 第一个参数为索引位置 , 第二个为要修改的值  System.out.println(sites);  }  }  输出结果：[Google, Runoob, Wiki, Weibo]。  **（三）删除元素**  如果要删除 ArrayList 中的元素可以使用 remove( ) 方法：  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  sites.remove(3); // 删除第四个元素  System.out.println(sites);  }  }  输出结果：[Google, Runoob, Taobao]。  **（四）计算大小**  如果要计算 ArrayList 中的元素数量可以使用 size( ) 方法：  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  System.out.println(sites.size());  }  }  输出结果：4。  **（五）迭代数组列表**  我们可以使用 for 来迭代数组列表中的元素：  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  for (int i = 0; i < sites.size(); i++)  {  System.out.println(sites.get(i));  }  }  }  输出结果：Google  Runoob  Taobao  Weibo  我们也可以使用 for −each 来迭代数组列表中的元素：  import java.util.ArrayList;  public class RunoobTest  public static void main(String[] args)  {  ArrayList<String> sites = new ArrayList<String>();  sites.add("Google");  sites.add("Runoob");  sites.add("Taobao");  sites.add("Weibo");  for (String i : sites)  {  System.out.println(i);  }  }  }  输出结果同上。  **（六）其他的引用类型**  ArrayList 中的元素实际上是对象，在以上实例中，数组列表元素都是字符串 String 类型。如果我们要存储其他类型，而 <E> 只能为引用数据类型，这时我们就需要使用到基本类型的包装类。  以下实例使用 ArrayList 存储数字（使用 Integer 类型）:  import java.util.ArrayList;  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<Integer> myNumbers = new ArrayList<Integer>();  myNumbers.add(10);  myNumbers.add(15);  myNumbers.add(20);  myNumbers.add(25);  for (int i : myNumbers)  {  System.out.println(i);  }  }  }  输出结果：10  15  20  25  **【学生】**思考、讨论。 | **教师通过知识拓展（一）展示，让学生了解议论文阅读与写作的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了知识拓展（一），如果我们要存储其他类型，而 <E> 只能为引用数据类型，这时我们就需要使用到基本类型的包装类。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **ArrayList 类有哪些？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示知识拓展（二）  **二、ArrayList 排序**  Collections 类也是一个非常有用的类，位于 java.util 包中，提供的 sort( ) 方法可以对字符或数字列表进行排序。  以下实例对数字进行排序：  import java.util.ArrayList;  import java.util.Collections; // 引入 Collections 类  public class RunoobTest  {  public static void main(String[] args)  {  ArrayList<Integer> myNumbers = new ArrayList<Integer>();  myNumbers.add(33);  myNumbers.add(15);  myNumbers.add(20);  myNumbers.add(34);  myNumbers.add(8);  myNumbers.add(12);  Collections.sort(myNumbers); // 数字排序  for (int i : myNumbers)  {  System.out.println(i);  }  }  }  输出结果：8  12  15  20  33  34  **【学生】**思考、讨论。 | **教师通过知识拓展（二）展示，让学生了解知识拓展（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了知识拓展（二），让学生了解ArrayList 排序的更多知识。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **ArrayList 排序应该如何做？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示思考与练习  一、简答题  （1）解释什么是类？怎样定义类？怎样实例化类？（举例说明）  （2）举例说明类方法和实例方法以及类变量和实例变量的区别。  （3）子类将继承父类的哪些成员变量和方法？子类在什么情况下隐藏父类的成员变  量和方法？在子类中是否允许有一个方法和父类的方法名字相同，而类型不同？说明你  的理由。  （4）下列程序有什么错误？  public class TakeCare  {  int a = 90;  static float b = 10.98f;  public static void main(String args[])  {  float c = a+b;  System.out.println(" = "+c);  }  }  （5）使用接口有哪些注意事项？编写一个类实现两个接口的程序。  （6）什么是多态？多态有什么作用？编写一图形类作为父类，类中含有成员方法  drawing( )，定义圆类与方形类继承于图形类，实现 drawing( ) 方法在不同类中的多态  特性。  （7）上机调试下面的程序：  class Student  {  String book, pen;  int 学号 , 年级 ;  Student(int number, int grade )  {  学号 = number;  年级 = grade;  }  void goTeachingRoom()  {  System.out.println(" 我带着 "+book+" 和 "+pen+" 来到了教室 , 准备听课 ");  }  }  class College  {  public static void main(String args[])  {  Student zxl = new Student(9901,2);  zxl.book = " 英语书 ";  zxl.pen = " 钢笔 ";  zxl.goTeachingRoom();  }  }  二、选择题  （1）在继承中，关于构造方法的说明，下列说法正确的是（　　）。  A. 子类无条件地继承父类的无参构造方法  B. 子类可以引用父类中的有参构造方法，使用 super 关键字  C. 如果子类没有构造方法，则父类无参构造方法作为自己的构造方法  D. 如果子类有无参构造方法，而父类的无参构造方法则被覆盖  （2）类包括两方面的内容，它们是（　　）。  A. 变量和常量 B. 函数和方法 C. 属性和方法 D. 构造器和方法  （3）Java 中将类成员的访问权限设置为最大，需要关键字（　　）。  A. public B. protected C. 不加修饰 D. private  （4）为（　　）处选择正确的答案。  ( ) abc  {  double pi = 3.14;  public double area(double r);  }  A. class B. public class C. interface D. implements  （5）用 static 修饰的类的变量称为类变量或（　　）变量。  A. 最终 B. 抽象 C. 静态 D. 常量  （6）Java 中 this 关键字的正确说法是（　　）。  A. this 关键字是在对象内部指代自身的应用  B. this 关键字可以在类中的任何位置使用  C. this 关键字和类关联，而不是和特定的对象关联  D. 同一个类的不同对象共用一个 this  （7）Java 中，（　　）类是所有类的父类。  A. Object B. Threadable C. Exception D. Subject  （8）类与对象的关系是（　　）。  A. 类是对象的抽象 B. 类是对象的具体实例  C. 对象是类的抽象 D. 对象是类的子类  （9）下列说法正确的是（　　）。  A. 不需要定义类，就能创建对象  B. 对象中必须有属性和方法  C. 属性可以是简单变量，也可以是一个对象  D. 属性必须是简单变量  （10）下列关于 final 的说法错误的是（　　）。  A. final 修饰的变量即为常量 B. final 修饰的抽象类能被继承  C. final 修饰的类不能被继承 D. final 修饰的方法不允许被子类覆盖  （11）关于 private 修饰符的方法，下列说法正确的是（　　）。  A. 可以被子类使用 B. 在同一个包中的类都可以使用  C. 只能被本类中的方法访问 D. 不能被重载  （12）设有类 x 具有 static 方法 a( )，则访问该方法的语句为（　　）。  A. a.x B. a.x( )  C. 实例化以后才能访问 D. x.a( )  （13）Java 中，如果类 C 是类 B 的子类，类 B 是类 A 的子类，下面描述正确的是  （　　）。  A. C 不仅继承了 B 中的成员，同样也继承了 A 中的成员  B. C 只继承了 B 中的成员  C. C 只继承了 A 中的成员  D. C 不能继承 A 或 B 中的成员  （14）子类必须通过（　　）关键字调用父类有参数的构造器。  A. this B. applet C. super D. father  （15）Java 语言中，下列关于类继承的描述，正确的是（　　）。  A. 一个类可以继承多个父类 B. 一个类可以有多个子类  C. 子类可以使用父类的所有方法 D. 子类一定比父类有更多的成员方法  （16）设类 B 和类 C 都不是抽象类，且类 B 是类 C 的父类。下列声明对象 x1 的语  句中不正确的是（　　）。  A. B x1=new B( ) B. B x1=new C( )  C. C x1=new C( ) D. C x1=new B( )  （17）覆盖与重载的关系是（　　）。  A. 覆盖只有发生在父类与子类之间，而重载可以发生在同一个类中  B. 覆盖方法可以不同名，而重载方法必须同名  C. final 修饰的方法可以被覆盖，但不能被重载  D. 覆盖与重载是同一回事  （18）为了区分重载的不同方法，要求（　　）。  A. 参数名必须不同 B. 返回值类型必须不同  C. 形式参数列表必须不同 D. 选项 A、B、C 都正确  （19）在 Java 中通常有两种途径实现多态：方法的重载和（　　）。  A. 覆盖 B. 继承 C. 实现 D. 抽象  （20）在继承时要实现方法的覆盖，需满足（　　）。  A. 方法的参数列表要相同  B. 方法名相同  C. 子类中的同名方法的权限要大于等于父类中的同名方法的权限  D. A、B、C 同时成立  （21）关于接口的说法，下列错误的是（　　）。  A. 接口是一种特殊的抽象类  B. 接口中只有方法声明和常量的定义，或者什么也没有  C. 把方法的定义和方法的实现从对象的层次中分开  D. 实现了多继承  **【学生】**思考、讨论。 | **教师通过思考与练习展示，让学生了解思考与练习的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了思考与练习，让学生能够更加巩固所学知识。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **Java 语言中引入包时，使用的关键字是（　　）。**  **A. import B. include C. implements D. package** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示实训任务  请在分析题目要求后，完成下列练习。  ①银行账户。  创建一个类 Account，它代表了银行账户。你来决定应该做什么样的账户，需要存储什么样的数据，以及将用什么样的方法。将其打包在名为 bank 的包中。  ②账目类型。  修改练习①“银行账户”中的实验，分析需求划分 Account 的不同子类，并创建它  们。创建 CheckingAccount 方法，它能检查出账户是否超出范围。  ③在线账目服务。  创建一个简单的应用程序 Teller.java，它使用上述两题的结果来提供一个在线账目  开户服务。  ④修改银行账户问题。  定义的接口 Personal 只包含两个方法 deposit 和 withdraw。运用 Personal 接口来定义一套不同的账户类型，重新定义类 Account。它必须能处理个人账户，进一步分成支票和存款两个账户。设计并开发提供保护的方法。例如，如果一个客户有一个存款和支票账户，须确保支票账户受存款账户保护。  **【学生】**思考、讨论。 | **教师通过实训任务展示，让学生了解实训任务的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了实训任务，增强学生的实操性。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **请在分析题目要求后，完成下列练习。**  **①创建一个名为 Factorial 的类，它包含一个计算传入参数的阶乘的方法。**  **②创建一个测试类，在其中使用上面创建的对象及方法来实际计算用户要求的阶乘的值。** | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 为了激发学生的合作探究精神，采用了小组比赛、组间互评的评价机制，引入了竞争小组成员也就多了一份责任感。 | |