实验三 类与对象

实验目的

1. 了解面向对象的编程思想。
2. 学会如何抽象出一个类及类中的实例变量与方法。
3. 学会创建对象，理解类、对象及引用三者之间的关系。

实验预习

1. 将面向过程的编程思维方式转换成面向对象的编程思维方式

通过一个简单的题目来区分面向过程与面向对象编程思想的特点。题目要求为：在图形接口画出四方形、圆形与三角形。当用户点选图形时，图形需要以中心点为轴，进行 360°顺时针旋转，并依据形状的不同播放不同的 AIF 音效文件。

两种编程思想大致的解决方法如图 3.1 所示，仅从图片上看，面向过程的 C 语言程序相对简单得多，而面向对象的 Java 语言程序要复杂许多，但是试想如果题目要求有所改动：1）再加入一个六边形需要旋转，并且旋转的时播放 HIF 音效文件；2）其中三角形与六边形需要以一个顶点为轴，进行 360° 逆时针旋转。通过对 Java 语言的深入学习，要满足以上增加的题目要求时，你会发现面向对象（Java 语言）的优势所在。

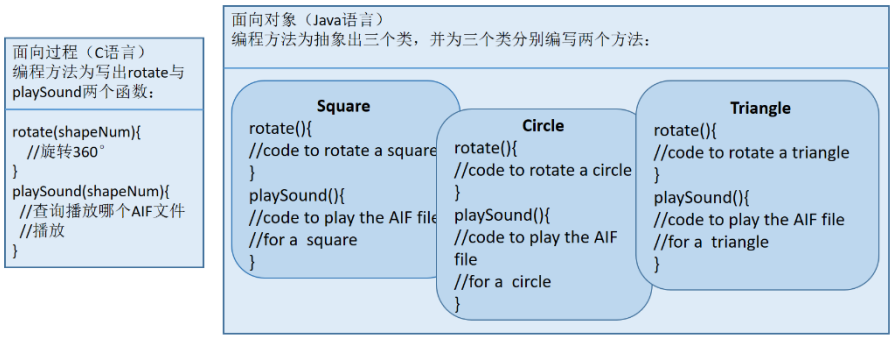

图 3.1 面向过程与面向对象解决方案的比较

2. 类、对象与引用

1. 设计类

在编程时，我们如何设计类呢？我们需要把具有相同特征的众多对象归纳成一个类，在设计类时，你可以将对象们已知的事物设计为“实例变量”，将它们可以执行的动作设计为“方法”。例如一个通讯簿如果被设计成一个类，那么它当中的每一个联系人的所有信息则是一个具体的对象，这些对象都有共同已知的属性（实例变量）包括：姓名、电话号码、住址等；那么你会对这些联系人执行可以执行的动作（方法）包括（方法）：取得电话号码、修改住址、删除联系人等。图 3.2 中 Song 和 Dog，已经设计出 “实例变量”和“方法”，请你试着完成 Television 类的设计。

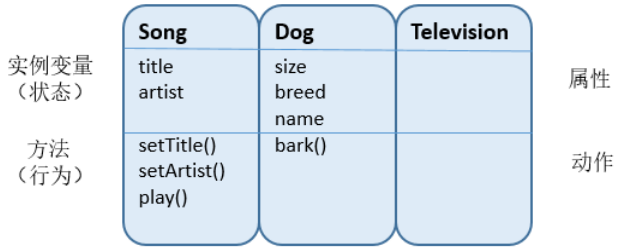

图 3.2 设计 Television 类

2. 定义引用类型的变量

对象的声明、创建与赋值如图 3.3 所示分为三个步骤。1）声明一个引用变量，“Dog myDog”这部分要求 Java 虚拟机分配空间给引用变量，并将此变量命名为 myDog，此变量将永远被固定为 Dog 类型。2）创建对象，“new Dog()”这部分要求 Java 虚拟机分配堆空间给新建立的 Dog 对象。3）连接对象和引用，“=”将新建立的 Dog 对象的内存地址赋值给 myDog 这个引用变量。

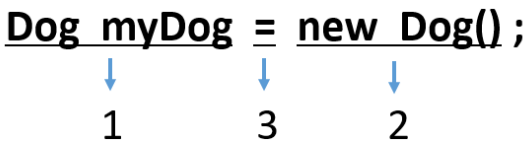

图 3.3 对象的声明、创建与赋值

我们建立了一个新的Dog对象，它的引用变量名称为myDog，我们可以通过myDog.size、myDog.breed 和 myDog.name 来表示名为“myDog”的这只狗的大小、品种和名字；通过myDog.bark()来表示它的叫声。

3. 命名规则

可以根据以下简单的规则来为类、方法或变量命名。

1） 名称必须以字母、下划线（-）或 $ 符号开头，不能用数字开头。

2） 除了第一个字符之外，后面可以使用数字。

3） 在满足上面两个规则外，还要避免使用 Java 的保留字。

表 3.1 保留字一览表

boolean	byte	char	double	float	int	long	short	public	private
protected	abstract	final	native	static	strictfp	synchronized	transient	volatile	if
else	do	while	switch	case	default	for	break	continue	assert
class	extends	implements	import	instanceof	interface	new	package	super	this
catch	finally	try	throw	throws	return	void	const	goto	enum

4. 对象与引用

1） 对象与引用之间的对应关系

假设声明两个 Book 的引用变量并创建两个 Book 对象，然后将 Book 对象赋值给引用变量，语句如下：

```
Book b = new Book();
Book c = new Book();
```

这时引用变量的数量为 2，对象的数量为 2，如图 3.4 中 1 所示。

假设再声明一个 Book 的引用变量 d，但不创建新的 Book 对象，而将变量 c 的值赋给变量 d，那么 c 与 d 引用到同一对象，语句如下：

```
Book d = c;
```

这时引用变量的数量为 3，对象的数量为 2，如图 3.4 中 2 所示。

假设把变量 c 的值赋给变量 b，那么 b 与 c 引用到同一对象，语句如下：

```
b = c;
```

这时引用变量的数量为 3，对象的数量为 1，被抛弃的对象数量为 1，如图 3.4 中 3 所示。

假设将 null 赋值给变量 c，那么这代表 c 不再引用任何事物，语句如下：

```
c = null;
```

这时作用中的引用变量数量为 2，null 引用数量为 1，可存取的对象数量为 1，被抛弃的对象数量为 1，如图 3.4 中 4 所示。

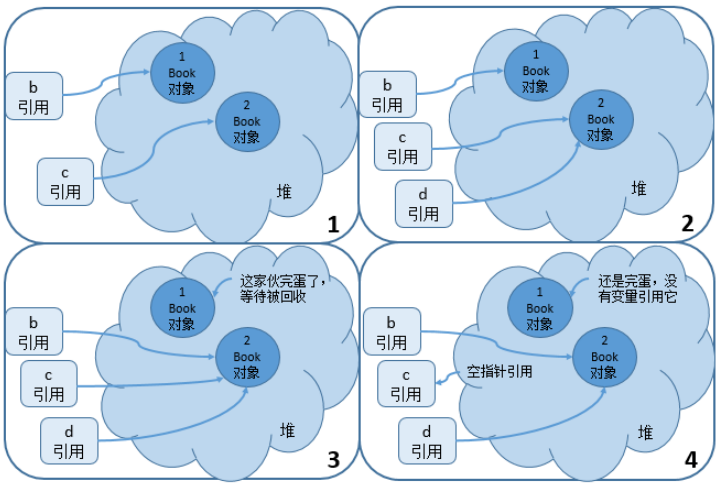

图 3.4 引用变量与对象

2） 数组也是对象

基本数据类型数组的声明、创建及赋值，以 int 型数组为例：

```
int [] nums;  //声明一个 int 型数组
nums = new int[4];  //创建一个 int 型数组对象
nums[0] = 1;
nums[1] = 2;
nums[2] = 3;
nums[3] = 4;  //为数组对象中的每一个元素赋值（int 型）
```

如果数组的类型不是基本数据类型，而是例如 Dog 类，我们将做如下的声明、创建及赋值：

```
Dog [] pets; //声明一个 Dog 型数组
pets = new Dog[4];  /*创建一个 Dog 型数组对象，此时我们有了四个 Dog 型的引用变量，那么这 4 个引用型变量还需要为它们分别创建对象*/
pets[0] = new Dog();
pets[1] = new Dog();
pets[2] = new Dog();
pets[3] = new Dog();  //为引用型变量创建对象
```

两种数组类型的区别可参见图 3.5 所示。

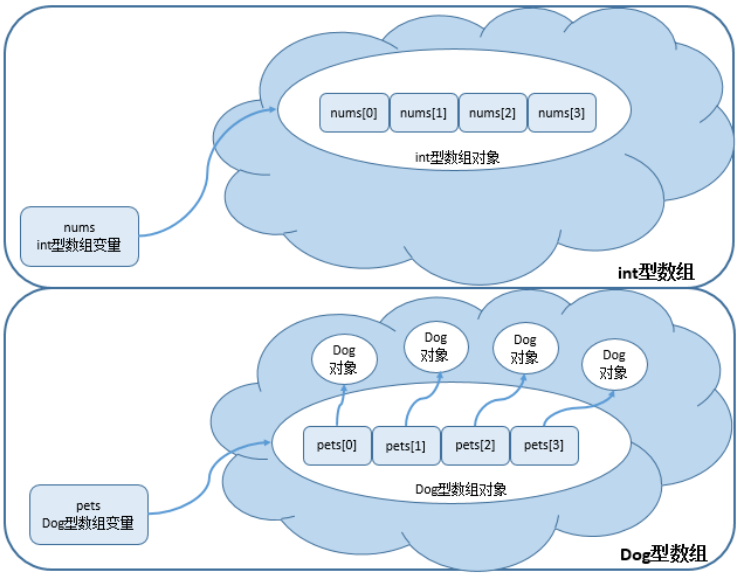

图 3.5 int 型与 Dog 型数组的区别

5. 对象的状态与行为

对象的状态在程序中通过实例变量体现，对象的行为则是通过方法体现。每个具体对象的实例变量取值不同，状态也就不同；每个对象都带有相同的方法，但是方法可以根据实例变量的值来表现不同的行为。

你可以传值给方法，也可以从方法中取返回值。方法运用形式参数，调用的一方会传入实际参数。形参与实参的类型一定要相同，用 Dog 对象举个例子：

```
Dog d = new Dog();
d.bark(3);
void bark(int numOfBarks){
  while(numOfBarks > 0){
    System.out.println(“ruff”);
    numOfBarks = numOfBarks-1;
  }
}
```

用 void 修饰的方法没有返回值，有返回值的方法类型一定要与返回值的类型相同。

6. 实例变量与局部变量

声明在类中的变量称为实例变量，表示对象已知的状态。实例变量永远都会有默认值，如果你没有明确的赋值给它，也没有调用 setter（本页后段封装部分详细说明），实例变量还是会有值。integers 的默认值为 0；float points 的默认值为 0.0；booleans 的默认值为 flalse；references 的默认值为 null。

在方法中也是可以声明变量的，这类变量称为局部变量，局部变量没有默认值，因此使用前必须进行初始化，如果没有初始化局部变量，则编译器会显示错误。

7. 构造方法

“Dog myDog = new Dog();”中的“new Dog()”看起来像是在调用 Dog()方法，实际上我们是在调用 Dog 的构造方法。构造方法与方法的不同点在于，它没有返回类型，并且与类同名。它是在创建对象的时候执行的代码，即这个构造方法内的程序代码会在你初始化一个对象的时候执行。例如我们如果想在创建一个 Dog 对象时，让它叫一声表示存在，程序可以这样写：

```
class Dog{
```

```
  public Dog(){
    System.out.println("ruff");
  }
}
public class UseADog{
  public static void main(String[] args){
    Dog myDog = new Dog();
  }
}
```

这个程序的运行结果会在屏幕上打印出：ruff。

注意：当一个 Java 源文件中有不止一个类的时候，只有能有一个 public 修饰的类，并且文件名要以 public 类的名字命名。

8. 封装

我们可以通过“myDog.size＝45;”，对 myDog 这个对象的大小进行赋值，并且无法阻止“myDog.size＝-6;”这样的赋值情况发生。这时我们通过封装实例变量来阻止不良数据。目前我们使用的封装做法为使用 private（私有）这个存取修饰符来隐藏数据，使用 public（公有）这个存取修饰符通过 setter 与 getter 两个方法来实现存取数据。程序可写成如下形式：

```
private int size;
public void setsize(int s){
  if(s > 20){
    size = s;
  }
}
public int getsize(){
  return size;
}
```

3. 概念理解练习

1. 名词填写Ⅰ

仔细分析下面的描述，将类、方法、对象、实例变量四个名词按照它们各自的特点填入空格中，有的空格可能会填入不只一个名词。

我是由.java 文件编译出来的_____类_____

我的实例变量值可以与其他兄弟姐妹不同_____

我的功能类似模板_____

我喜欢执行工作_____

我带有很多方法_____

我代表“状态”_____

我拥有很多行为_____

我呆在对象中_____

我生于堆上_____

我被用来创建对象实例_____

我的状态可以改变_____

我会声明方法_____

我可以在运行期变化 _____

2. 名词填写Ⅱ

仔细分析下面的描述，将 instance variable、argument、return、getter、setter、encapsulation、public、private、pass by value 和 method 这些组件按照它们各自的特点填入空格中，有的空格可能会填入不只一个组件。

一个类可以带有很多个_____

一个方法只能带有一个_____

可以被隐含的转换_____

我喜欢 private 的实例变量_____

其实就是制作一个拷贝_____

应该只有 setter 才能更新_____

方法可以带很多个_____

根据定义返回_____

不应该以实例变量来运用_____

可以有许多个参数_____

被定义成采用一个参数_____

帮忙创建封装 _____

总是单飞_____

基础练习

1. 补全程序练习

依据注释内容，补全程序代码，按要求完成程序功能。提示：每行注释需要补全的代码可能不只一行。

1. 填写程序，要求输入圆的半径，得出圆的周长和面积。

```
class Circle{
  //定义圆的半径、π 的值
  //编写计算周长的方法
  //编写计算面积的方法
}
public class TestCircle{
  public static void main(String[] args){
    //定义一个圆的对象、所需变量
    //输入圆的半径
    //输出圆的周长、面积
  }
}
```

2. 填写程序，要求通过判断狗的大小，输出不同的叫声。

```
class Dog{
  //利用封装的方法，定义 size 变量，使得 size 变量的值始终大于 10
  /*编写 bark 方法，size 值 > 60 的时候输出“Woof！”；size 值 > 40 的时候输出“Ruff！”；其他取值时输出“Yip！”。*/
}
```

```
public class DogTestDrive{
  public static void main(String[] args){
    //创建至少三个 Dog 对象
    //为每个对象设定或输入 size 的大小
    //输出每个 Dog 对象的 size 大小
    //调用 bark 方法输出不同 size 大小的叫声
  }
}
```

2. 编写程序练习

1. 编写程序，统计班级同学的期末考试成绩，按总成绩高低进行排名。要求统计人数为 10 人，统计 6 门课程考试成绩，从键盘输入每个同学的姓名、学号及各门考试成绩，计算总成绩并排序后，输出每人的排名及总成绩。（程序中至少包含 3 个类）

2. 有长、短半径分别为 4m、3m，高为 10m 的椭圆形柱子六根。现需要为柱子顶部及四周粉刷油漆，油漆每 5 升一桶，每 5 升可粉刷面积 70m^2。编写程序计算出完成 6 根柱子的一次粉刷需油漆多少桶，结果精确到小数点后一位。（程序中至少包含 3 个类）