2、具体内容

Object类的主要特点是可以解决参数的统一问题，也就是说使用Object类可以接收所有的数据类型。

■Object类简介

在Java之中只有一个类是不存在有继承关系的，这个类就是Object，也就是说所有的类默认的情况下都是Object的子类，以下两种类的定义效果完全相同：

class Person{} //一个类	class Person extends Object{} //一个类

在Object类设计的时候考虑到了所有继承的问题，所以该类提供有无参构造方法，这样所有的类在定义时即便不知道Object类的存在也不会出现构造方法调用失败的语法错误。

那么既然Object类时所有类的父类，那么这种情况下就可以使用Object类接收所有的子类对象。

范例：观察Object类接收所有子类对象

```
class Person {} // 一个类

public class JavaDemo {
	public static void main(String args[]) {
		Object obj = new Person() ;  // 向上转型
		if (obj instanceof Person) {
			Person per = (Person) obj ;
			System.out.println("Person对象向下转型执行完毕。") ;
		}
	}
}
```

如果一个程序的方法要求可以接收所有类对象的时候就可以利用Object实现处理。但是还有一点需要注意，在Java设计过程之中对于所有的引用数据类型实际上都可以使用Object类进行接收，包括数组也可以。

范例：使用Object类接收数组

```
public class JavaDemo {
	public static void main(String args[]) {
		Object obj = new int [] {1,2,3} ;	// 向上转型
		if (obj instanceof int[]) { // 是否为整型数组
			int data [] = (int []) obj ; // 向下转型
			for (int temp : data) {
				System.out.print(temp + "、") ;
			}
		}
	}
}
```

Object是一个万能的数据类型，它更加适合于进行程序的标准设计。

■获取对象信息：toString()

Object虽然是一个类，但是这个类本身也是提供有一些处理方法的，在Object类之中提供有一个“toString()”方法，该方法可以获取一个对象的完整信息：public String toString()。

范例：观察默认的toString()使用

```
class Person {

}
public class JavaDemo {
	public static void main(String args[]) {
		Person per = new Person();
		System.out.println(per);
		System.out.println(per.toString());//Object类继承而来
	}
}
```

可以发现在之前进行对象直接输出的时候所调用的方法就是toString()方法，所以这个方法调用与不调用的效果是一样的，所以在以后的开发之中对象信息的获得可以直接覆写此方法。

范例：覆写toString()方法

```
class Person {
	private String name ;
	private int age ;
	public Person(String name,int age) {
		this.name = name ;
		this.age = age ;
	}
	//Override
	public String toString() {
		return "姓名：" + this.name + "、年龄：" + this.age ;
	}
```

```
}
public class JavaDemo {
	public static void main(String args[]) {
		Person per = new Person("张三",20) ;
		System.out.println(per) ;
	}
}
```

以后在编写简单Java类的过程之中只需要覆写toString()方法即可；

■对象比较：equals()

Object类之中另外一个比较重要的方法就是在于对象比较的处理之上，所谓的对象比较主要的功能时比较两个对象的内容是否完全相同，假如说现在有两个Person对象，要想确认这两个对象是否一致，但是两个对象本身会有不同的 内存地址数值，所以此时的比较应该是通过内容的比较完成的。

范例：对象比较的基础实现

```
class Person {
	private String name ;
	private int age ;
	public Person(String name,int age) {
		this.name = name ;
		this.age = age ;
	}
	public String toString() {
		return "姓名：" + this.name + "、年龄：" + this.age ;
	}
	public String getName() {
		return this.name ;
	}
	public int getAge() {
		return this.age ;
	}
}
public class JavaDemo {
	public static void main(String args[]) {
		Person perA = new Person("张三",20) ;
		Person perB = new Person("张三",20) ;
		if (perA.getName().equals(perB.getName()) &&
			perA.getAge() == perB.getAge()) {
			System.out.println("是同一个对象。") ;
		} else {
			System.out.println("不是同一个对象。") ;
		}
	}
}
```

此时的确实现了对象比较的功能，但是这个功能比较麻烦：

• 由于需要进行对象比较的时候要将每一个属性都进行相等判断，所以在外部要调用大量的getter()方法；

• 对象比较应该是一个类内部所具备的功能，而不应该在外部定义；

Object类作为所有类的父类提供了对象比较操作的支持，对于对象比较的操作实现可以使用equals()方法完成；

• 对象比较：public boolean equals(Object obj)，可以接收所有类型；

```
public boolean equals(Object obj) {
    return (this == obj);
 }
```

也就是说对于实际的使用者而言，如果想要正确判断处理，那么就必须要在子类中覆写此方法，并且进行属性判断。

范例：观察Object类中的equals()方法覆写

```
class Person {
    private String name ;
    private int age ;
    public Person(String name,int age) {
        this.name = name ;
        this.age = age ;
    }
    public String toString() {
        return "姓名：" + this.name + "、年龄：" + this.age ;
    }
    // equals()方法这个时候会有两个对象：当前对象this、传入的Object
    public boolean equals(Object obj) {
        if (!(obj instanceof Person)) {
            return false ;
        }
        if (obj == null) {      // 不关心null的比较
            return false ;
        }
        if (this == obj) {      // 同一个地址
            return true ;
        }
        Person per = (Person) obj ; // 目的是为了获取类中的属性
        return this.name.equals(per.name) &&  this.age == per.age ;
        /*与return 一致
        if (perA.getName().equals(perB.getName()) &&
            perA.getAge() == perB.getAge()) {
            System.out.println("是同一个对象。") ;
        } else {
            System.out.println("不是同一个对象。") ;
        }
        */
    }
}
public class JavaDemo {
    public static void main(String args[]) {
```

```
        Person perA = new Person("张三",20) ;
        Person perB = new Person("张三",20) ;
        System.out.println(perA.equals(perB)) ;
    }
}
```

String类作为Object的子类，那么这个类里面实际上已经覆写了equals()方法。