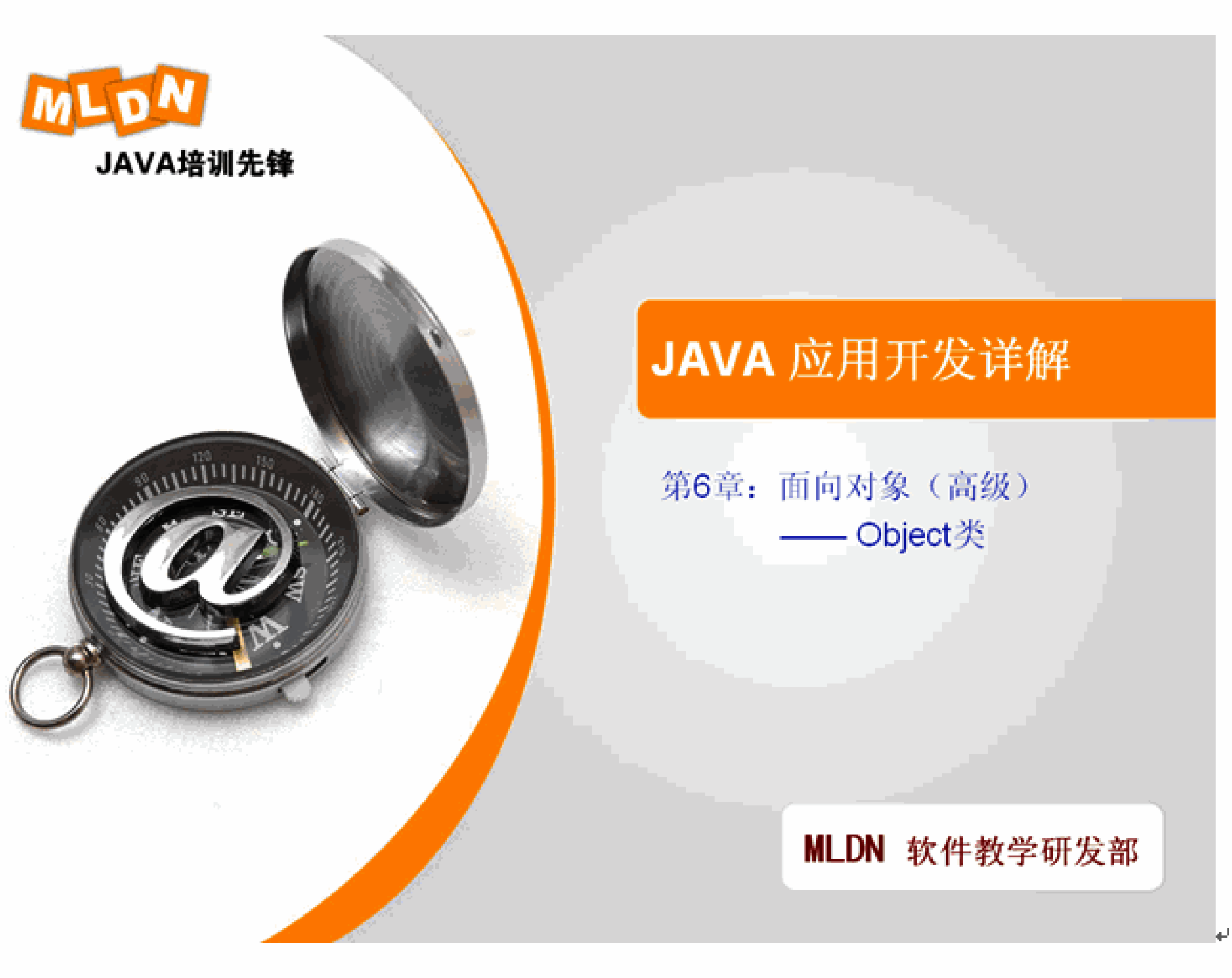
1、课程名称：Object 类


JAVA培训先锋

JAVA 应用开发详解

第6章：面向对象（高级）
—— Object类

MLDN 软件教学研发部

2、本课程预计讲解的知识点

本章目标

- 掌握Object类的作用
- 掌握Object类中toString()、equals()方法的作用
- 掌握Object接收引用数据类型的操作

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

3、具体内容

在 Java 中一切的类都是以继承的关系存在的，如果现在假设定义了一个 Person 类。

```
class Person{}
```

如果一个类在定义时没有明确的指名继承那个类，则默认继承 Object 类，也就是说以上的类实际上的定义格式：

```
class Person extends Object{}
```

Object类

- 在Java中所有的类都有一个公共的父类就是Object类，一个类只要没有明显的继承一个类，则肯定是Object类的子类。如下两种代码表示的含义都是一样的：
  - class Person **extends Object**{}
  - class Person{}

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

Object类中的主要方法

No.	方法名称	类型	描述
1	public Object()	构造	构造方法
2	public boolean equals(Object obj)	普通	对象比较
3	public int hashCode()	普通	取得Hash码
4	public String toString()	普通	对象打印时调用

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

```
class Demo{      // 定义 Demo 类，实际上就是继承了 Object 类
};
public class ObjectDemo01{
	public static void main(String args[]){
		Demo d = new Demo() ;     // 实例化 Demo 对象
		System.out.println("不加 toString()输出："+d) ;
		System.out.println("加上 toString()输出："+d.toString()) ;
	}
};
```

从以上代码的运行结果可以发现，加与不加 toString()完成的功能都是一样的，也就是证明了对象在打印的时候一定会调用 toString()方法，是默认调用的。

那么此时就可以利用此操作完成信息的输出。

```
class Person{
	private String name ;	// 定义 name 属性
	private int age ;		// 定义 age 属性
```

```
	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 class ObjectDemo02{
	public static void main(String args[]){
		Person per = new Person("李兴华",30) ; // 实例化 Person
		System.out.println("对象信息：" + per) ;
	}
};
```

就是打印对象信息。

equals()方法

- equals()方法的主要功能是进行对象的比较操作。
- String本身也是Object类的子类，所以已经覆写了此方法

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

equals 方法的主要功能是完成两个对象的比较的。

对象的比较操作在之前已经讲解过了，下面直接将之前的代码修改，使用标准的 equals()完成。

```
class Person{
	private String name ;	// 定义 name 属性
	private int age ;		// 定义 age 属性
	public Person(String name,int age){
		this.name = name ;
		this.age = age ;
	}
	public boolean equals(Object obj){
		if(this==obj){		// 地址相等
			return true ;	// 肯定是同一个对象
		}
		if(!(obj instanceof Person)){ // 不是 Person 对象
			return false ;
		}
		Person per = (Person) obj ;  // 向下转型
		if(per.name.equals(this.name)&&per.age == this.age){
			return true ;	// 依次比较内容
		}else{
			return false ;
		}
	}
	public String toString(){
		return "姓名：" + this.name + "；年龄：" + this.age ;
	}
};
public class ObjectDemo03{
	public static void main(String args[]){
		Person per1 = new Person("李兴华",30) ;	// 实例化 Person
		Person per2 = new Person("李兴华",30) ;	// 实例化 Person
		System.out.println(per1.equals(per2)?"是同一个人！" : "不是同一个人！") ;
		System.out.println(per1.equals("hello")?"是同一个人！" : "不是同一个人！") ;

	}
};
```

Object 类是所有类的父类，则所有类的对象都可以使用 Object 接收，但是 Object 不光可以接收对象，还可以接收任意的引用数据类型。

使用 Object 接收接口实例

```
interface A{
	public String getInfo();
}
class B implements A{
	public String getInfo(){	// 覆写方法
		return "Hello World!!!" ;
	}
};
public class ObjectDemo04{
	public static void main(String args[]){
		A a = new B() ;  // 向上转型，为接口实例化
		Object obj = a ;  // 使用 Object 接收，向上转型
		A x = (A)obj ;   // 向下转型
		System.out.println(x.getInfo()) ;
	}
};
```

数组实际上也可以使用 Object 类进行接收

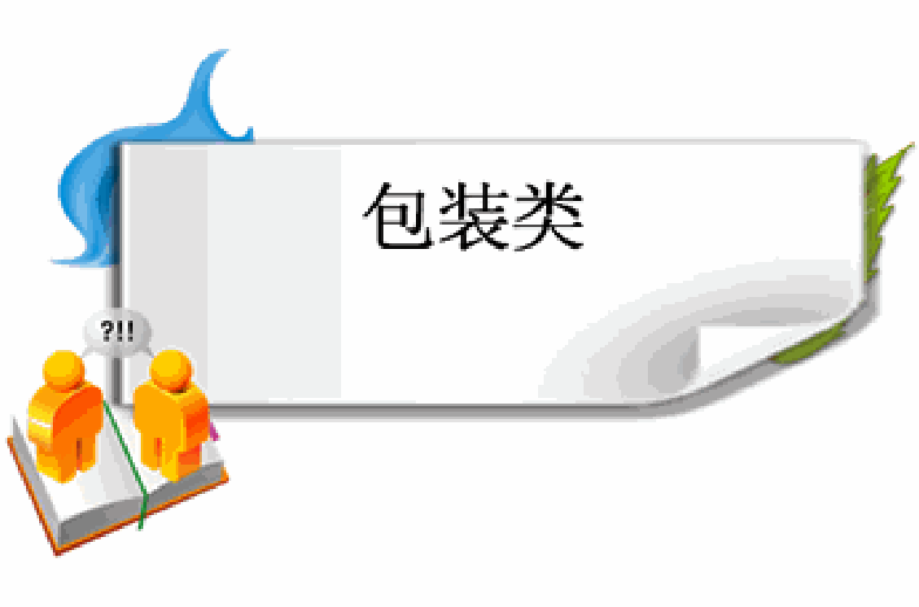
```
public class ObjectDemo05{
	public static void main(String args[]){
		int temp[] = {1,3,5,7,9} ;	// 定义数组
		Object obj = temp ;	// 使用 Object 接收数组
		print(obj) ;
	}
	public static void print(Object o){
		if(o instanceof int[]){		// 判断是否是整型数组
			int x[] = (int[])o ;
			for(int i=0;i<x.length;i++){
				System.out.print(x[i] + "\t") ;
			}
		}
	}
};
```

4、总结

1、 Object 类是所有类的父类，只要是引用数据类型都可以使用 Object 进行接收
2、 对象在进行向下转型之前一定要先发生向上转型，要使用 instanceof 关键字判断
3、 toString()：对象打印时调用
4、 equals()：对象比较时调用
5、 String 类也是 Object 类的子类

5、预习任务

下一章内容



E-MAIL：mldnqa@163.com www.MLDNJAVA.cn