1. 抽象类
2. 使用abstract定义抽象类

|  |
| --- |
| **public abstract class** Parent {  可以定义一系列的抽象方法 } |

注意：

1. 抽象类不能实例化、只能被继承
2. 继承抽象类的子类必需实现所有的抽象方法
3. 作用:实现方法重写
4. 抽象方法

//普通方法、构造方法、重载方法、重写方法

1. 什么是抽象方法：

使用**abstract定义的方法且没有方法体的方法称为抽象方法.**

**public abstract void** 方法名称();

示例:使用抽象类模拟电脑usb的工作原理

1. 定义抽象类

|  |
| --- |
| **public abstract class** Usb {  **public abstract void** work(); *//抽象方法* } |

1. 定义子类去继承抽象类

|  |
| --- |
| **public class** Mouse **extends** Usb {  @Override  **public void** work() {  System.***out***.println(**"光标正在移动......"**);  } }  **public class** Kboard **extends** Usb {  @Override  **public void** work() {  System.***out***.println(**"键盘正在输入数据....."**);  } } |

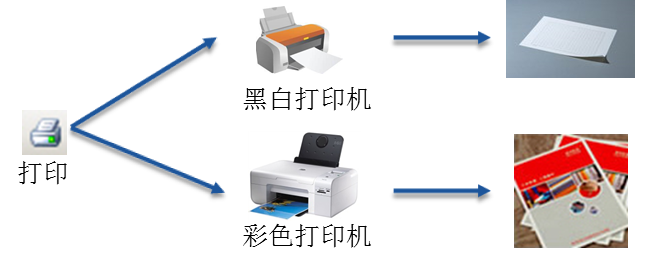
1. 测试抽象类

|  |
| --- |
| **public static void** main(String[] args) {  *//测试Usb的工作原理* Usb usb=**new** Kboard();  usb.work(); } |

1. final关键字
2. 使用final关键字定义的类为安全类|密封类:不能被继承
3. 使用final关键字定义的方法为最终方法:不能被重写
4. 使用final修饰的量|属性为常量，且必需初始值.

常量特点:不能被更改

1. 多态(需要借助方法重写)
2. 生活中多态：同一种操作，由于条件不同，产生的结果也不同



1. 程序中的多态:

同一个引用类型，使用不同的实例而执行不同的操作.

父类名 引用名=new 任何子类名();

1. 父类与子类之间的转换

3.1向上转型

将子类对象转换为父类引用

3.2向下转型:将父类引用对象转化为子类对象(装什么拿什么)

子类名 对象名=(子类名)父类引用名;

注意: *父类引用，无法访问子类的成员,重写方法除外*

1. 使用**instanceof判断某个对象是不是该类的对象**

**if**(对象 **instanceof** 类名) 是返回true,反之false