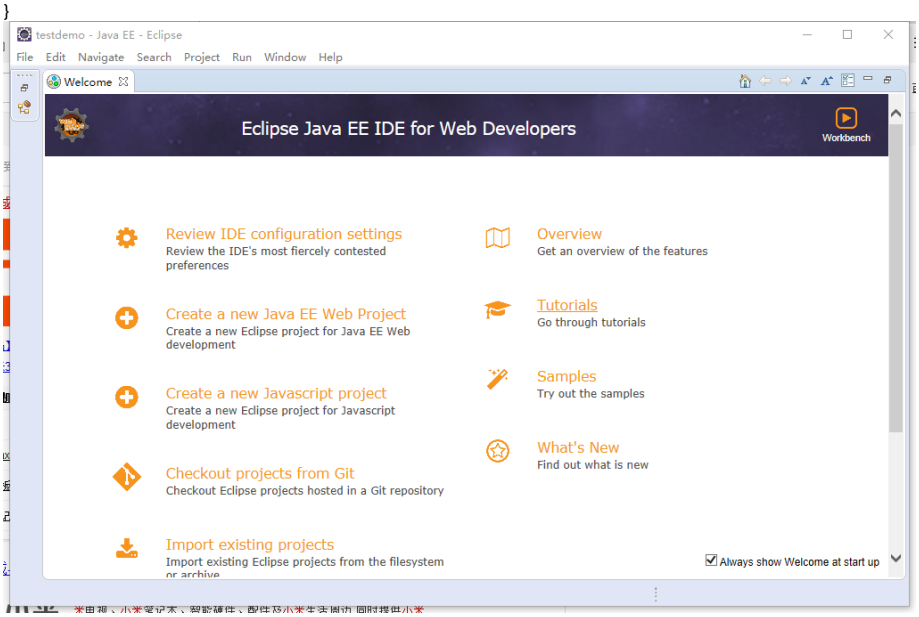
第一个Java程序

上上周学习Java第一个程序Hello World!，今天给电脑换了固态硬盘，重做完系统后，凭记忆登录www.oracle.com，下载了JAVA SE 最新版，安装完成后，查询教程配置了PATH，然后用记事本默了那个程序，直接编译通过，竟然没有出错，开心！

```
public class Hello{      //文件名和类名要一致
  public static void main(String args[]) {
    System.out.println("Hello World!");
  }
}
```




```
public class TestDemo{
public static void main(String args[]){
int x=9;
int y=5;
System.out.println(x/(double)y);
}
}
```

```
public class TestDemo{
public static void main(String args[])
{char c='a';
int num=c;
System.out.println(num);}
}
```

```
public class TestDemo{
public static void main(String args[]){
char c='A';
int num=c;
num=num+32;
c=(char)num;
System.out.println(c);
}
}
```

```
public class TestDemo{
public static void main(String args[]){
boolean flag=false;
if(!flag){
System.out.println(flag);}
}
}
```

```
public class TestDemo{
public static void main(String args[]){
String str="Hello";
str=str+"World";
str+="!!!";
System.out.println(str);
}
}
```

```
public class TestDemo{
public static void main(String args[]){
int numA=100;
double numB=99.0;
String str="加法运算"+numA+numB;
String str1="Hello\"World\Hello MLJS"
String str1="Hello \"World\" \n\t Hello MLJS";
System.out.println(str);
System.out.println(str1);
}
}
```

```
public class TestDemo{
public static void main (String args[]){
int numA=10;
int numB=20;
int max=numA>numB?numA:numB;
System.out.println(max);
}
}

public class TestDemo{
public static void main(String args[]){
int numA=9;
int numB=11;
int result=numA<<3;
System.out.println(numA&numB);
System.out.println(numA|numB);
System.out.println(result);
}
}

public class TestDemo{
public static void main(String args[]){

}
}

public class TestDemo{
public static void main(String args[]){
double score=119.1;
if(score<60.0){
    System.out.println("小白的成绩");
}else if(60<=score&&score<=90){
    System.out.println("矮油不错");
}else if(90<score&&score<=100){
    System.out.println("NB");
}else{
    System.out.println("你家的成绩这么怪异");
}
}
}

public class TestDemo{
public static void main(String args[]){
int ch=3;
switch(ch){
case 2:{
System.out.println("内容是2");break;
}case 3:{
System.out.println("内容是3");break;
}case 1:{
System.out.println("内容是1");break;
}default:{
System.out.println("没有匹配内容！");break;
}
}
}
}//switch不能判断布尔判断式，只能判断内容。jdk1.7开始支持String判断了，识别大小写。


public class TestDemo{
public static void main(String args[]){
int current=1; int sum=0;
while(current<=100){
sum+=current;
current++;
}
System.out.println(sum);
}
}


//do{} while();以后建议不要使用
public class TestDemo{
public static void main(String args[]){
int current=101; int sum=0;
do{
sum+=current;
current++;
}
while(current<=100);
System.out.println(sum);
}
}

//for(循环初始化条件；循环判断；循环条件变更)
//如果不知道循环次数，但是知道循环结束条件的时候用while
//如果知道循环的次数，使用for循环。
//循环可以嵌套循环
public class TestDemo{
public static void main(String args[]){
```

```
int max=9;
for(int x=1;x<10;x++){
System.out.print("开始抽第"+x+"根烟:");
for(int y=1;y<=20;y++){
System.out.print("嘬、");
}System.out.println();
}
System.out.println("离完蛋不远了，还差一根烟，嘬完就没得嘬了！");
}
}

//单行注释符号是//而不是\\  下列程序是乘法口诀表
public class TestDemo{
  public static void main(String args[]){
    for(int x=1;x<10;x++){
      for(int y=1;y<=x;y++){
        System.out.print(x+"*"+y+"="+(x*y)+"\t");//制表符是\t不是/t
   }System.out.println();
  }
 }
}

//循环语句控制continue和break,需和判断语句配合使用
public class TestDemo{
public static void main(String args[]){
for(int x=0;x<=10;x++){
if(x==3){break;}
System.out.println("x="+x);}
}
}

//面向对象就是一种组件化的设计思想

class Book{
String title;
double price;
public void getinfo(){
System.out.println("图书名称："+title+",图书价格："+price);}
}
public class TestDemo{
public  static void main(String args[]){
Book bk=new Book();
//bk.title="Java开发";
//bk.price=89.9;
bk.getinfo();
}
}
```

第7章 课时25：深入分析类与对象（构造方法与匿名对象）

```
class Book{
public Book(){//构造方法。
System.out.println("***********");
}
public void print(){//普通方法
System.out.println("HELLO");}
}
public class Demo{
public static void main(String args[]){
Book book=null;//声明对象
book=new Book();//实例化对象，所有构造方法都在对象使用关键字new实例化时候调用，只能调用一次。
System.out.println(book);//输出第一次实例化后对象的堆内存地址
book.print();//普通方法可以多次调用
book.print();//普通方法可以多次调用
book.print();//普通方法可以多次调用
book=new Book();//又重新实例化了一次对象，堆内存里面的地址已经变化，也就是说这已不是之前那个对象了
System.out.println(book);//验证上一句话

}//构造方法是在实例化新对象（new）的时候只调用一次
}//普通方法是在实例化对象产生之后可以随意调用多次
//总结：1、构造方法的定义要求：方法名称与类名称相同，且无返回值声明。
//2、构造方法是在类对象使用关键字new实例化对象时候被默认调用的，不管你代码如何改变，只要是有了关键字new，就一定需要构造方法；
//3、一个类中至少保留一个构造方法，如果没有明确定义构造方法，那么会自动生成一个无参的什么都不做的构造方法。
//4、构造方法的核心功能是在类对象实例化的时候为类中的属性初始化；
//5、构造方法重载时只考虑参数的类型和个数即可。
```

第7章 课时26：深入分析类与对象（综合实战：简单Java类）

```
//类名称必须存在有意义；
//类中所有属性必须用private封装，封装后的属性必须提供有setter，getter;
//类之中可以提供有任意多个构造方法，但必须保留有一个无参构造方法；
//类之中不允许出现任何的输出语句，所有信息输出必须交给被调用的处输出，
//类之中需要提供有一个取得对象完整信息的方法，暂定为：getInfo（），返回String数据；
//第一个代码模型，老师说整个Java有10来个，能掌握核心命脉。
class Emp{//定义一个有意义的类
    private int empno;
    private String ename;
    private String job;
    private double sal;
    private double comm;
    public Emp(){}//必须有一个无参构造方法
    public Emp(int eno,String ena,String j,double s,double c){//构造方法重载
```

```
    empno=eno;
    ename=ena;
    job=j;
    sal=s;
    comm=c;
    }
    public void setEmpno(int e){
    empno=e;
    }
    public void setEname(String e){
    ename=e;
    }
    public void setJob(String j){
    job=j;
    }
    public void setSal(double s){
    sal=s;
    }
    public void setComm(double c){
    comm=c;
    }
    public int getEmpno(){
    return empno;
    }
    public String getEname(){
    return ename;
    }
    public String getJob(){
    return job;
    }
    public double getSal(){
    return sal;
    }
    public double getComm(){
    return comm;
    }
    public String getInfo(){//需要一个取得对象完整信息的方法
    return"雇员编号"+empno+"\n"+
          "雇员姓名"+ename+"\n"+
          "雇员职位"+job+"\n"+
          "基本工资"+sal+"\n"+
          "佣　金"+comm;
    }
}
public class TestDemo{
    public static void main(String args[]){
   //编写测试程序
    Emp e=new Emp(7369,"SMITH","CLERK",800.00,1.0);
    e.setEname("ALLEN");
    System.out.println(e.getInfo());//输出雇员所有信息
    System.out.println("姓名"+e.getEname());//只取得姓名，用getter方法。
    }
}
//所有类之中提供的setter、getter方法可能某些操作不会使用到，但是依然必须提供，
//所有的setter方法除了具备有设置属性的内容之外，也具备有修改属性内容的功能。
//简单Java类为日后进行整个项目开发的千分之一的组成部分，也是最为重要的组成部分，所以必须做到信手拈来。
```

第8章 课时27：数组的定义与使用（基本概念）

```
//声明数组：数据类型 数据名称[]=null;
//开辟数组：数据名称 =new 数据类型[];
//声明并开辟数组：数据类型 数据名称[]=new 数据类型[数据长度]
public class Array{
    public static void main(String args[]){
    int data []=new int[3];
    data[0]=10;
    data[1]=20;
    data[2]=30;
    for(int x=0;x<data.length;x++){
    System.out.println(data[x]);
}
}
}
//数据既然是引用类型，就一定会发生引用传递，引用传递的本质是同一块堆内存空间可以被不同的栈访问。
public class Array{
public static void main(String args[]){
int data[]=new int[3];
data[0]=10;
data[1]=20;
data[2]=30;
int temp[]=data;
temp[0]=99;
for (int x=0;x<data.length;x++){
System.out.println(temp[x]);
}
}
}

//数组的静态初始化 int data[]=new int []{1,2,3,4,5}
public class Arry{
public static void main(String args[]){
```

```
  int data[]=new int[]{1,2,3,4,5};
for (int x=0;x<data.length;x++){
System.out.println(data[x]);}
}
}
```

第8章 课时28：数组的定义与使用（二维数组）

```
public class Array{
public static void main(String args[]){
int data[][]=new int[][]{
{1,2,3},
{4,5,6},
{7,8,9}
};
//外层循环是控制数组的行内容
for(int x=0;x<data.length;x++){
  for(int y=0;y<data[x].length;y++){
  System.out.print(data[x][y]+"\t");
  }
System.out.println();
}
}
}
```

第8章 课时29：数组的定义与使用（数组与方法的引用操作）

```
//排序，当你困了又睡不着的时候，就赶快滚起来学习，一般来说会更困，所以坚持到特别困就去睡觉，保证你睡的又快又好。如果不困，那么问题解决了。好好学习吧。
public class Array{
  public static void main(String args[]){
    int data[]=new int[]{2,1,0,9,5,13,7,6,8};
    print (data);
    sort(data);
    print (data);
  }

  public static void sort(int arr[]){
    for(int x=0;x<arr.length;x++){
      for(int y=0;y<arr.length-1;y++){
        if(arr[y]>arr[y+1]){
        int t=arr[y];
        arr[y]=arr[y+1];
        arr[y+1]=t;
        }
      }
    }
  }

  public static void print(int temp[]){
    for(int x=0;x<temp.length;x++){
    System.out.print(temp[x]+",");
    }
  System.out.println();
  }

}
```

```
public class Array{
  public static void main(String args[]){
    int data[]=new int[]{1,2,3,4,5,6,7,8,9};
    print (data);
    reverse(data);
    print(data);
  }
  public static void reverse(int arr[]){
    int len=arr.length/2;//转置的次数
    int head=0;//头部索引
    int tail=arr.length-1;//尾部索引
    for(int x=0;x<len;x++){
    int temp=arr[head];
    arr[head]=arr[tail];
    arr[tail]=temp;
    head++;
    tail--;
    }
  }



  public static void print(int temp[]){
    for(int x=0;x<temp.length;x++){
    System.out.print(temp[x]+",");
    }
  System.out.println();
  }

}
```

```
public class Array{
```

```
  public static void main(String args[]){
    int data[][]=new int[][]{{1,2,3},{4,5,6},{7,8,9}};
    print(data);
    reverse(data);
    print(data);
  }
  public static void reverse(int arr[][]){
    for(int x=0;x<arr.length;x++){
      for(int y=0;y<arr[x].length;y++){
        if(x!=y){
          int temp=arr[x][y];
          arr[x][y]=arr[y][x];
          arr[y][x]=temp;
        }
      }
    }
  }

  public static void print(int temp[][]){
    for(int x=0;x<temp.length;x++){
      for(int y=0;y<temp[x].length;y++){
        System.out.print(temp[x][y]+"、");
      }
    System.out.println();
    }
  }

}
```

第8章 课时30：数组的定义与使用（数组相关操作方法）

```
\\数组拷贝
public class Array{
  public static void main(String args[]){
    int dataA[]=new int[]{1,2,3,4,5,6,7,8};
    int dataB[]=new int[]{11,22,33,44,55,66,77,88};
  System.arraycopy(dataA,4,dataB,2,3);
  print(dataB);
  }
  public static void print(int temp[]){
    for(int x=0;x<temp.length;x++){
    System.out.print(temp[x]+"、");
    }
  System.out.println();
  }
}
```

```
\\Java排序
public class Array{
  public static void main(String args[]){
    int data[]=new int[]{6,3,9,23,2,45};
    java.util.Arrays.sort(data);
    print(data);
  }
  public static void print(int temp[]){
    for(int x=0;x<temp.length;x++){
      System.out.print(temp[x]+"、");
    }
    System.out.println();
  }
}
```

第8章 课时31：数组的定义与使用（对象数组）

数组是引用类型，而类也同样是引用类型，所以如果是对象数组的话表示一个引用类型里面嵌套了其他的引用类型。在之前使用的数组都属于基本数据类型的数组，但是所有的引用数据类型也同样可以定义数组。这样的数组叫对象数组。
如果要想定义对象数组（以类为例），可以采用如下的形式完成：
 动态初始化：
 1、声明并开辟对象数组：类名称 对象数组名称[]=new 类名称[长度]
 2、分步完成：
  |-声明对象数组：类名称 对象数组名称[]=null;
  |-开辟对象数组：对象数组名称=new 类名称[长度];
举例：Book books []=new Book[3];

第9章 课时32：String类对象的两种实例化方式

```
String str="Hello World";//直接赋值
String str=new String("Hello World");//构造方法实例化
```

第9章 课时33：字符串比较

```
//==是进行数值判断，用于字符串比较时，比较的是字符串内存地址数值的大小。
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb=new String("hello");
    String strc=strb;//引用传递
    System.out.println(stra==strb);
    System.out.println(stra==strc);
    System.out.println(strb==strc);
  }
```

```
}


//字符串比较用equal()例如stra.equal(strb)
//比较字符串内容的话，可以使用String类里面定义的方法。public boolean equals(String str)
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb=new String("hello");
    String strc=strb;//引用传递
    System.out.println(stra.equals(strb));
    System.out.println(stra.equals(strc));
    System.out.println(strb.equals(strc));
  }
}
```

第9章 课时34：String常量为匿名对象

java中什么是匿名对象？
普通声明一个对象是这样的
A a = new A();
那么这个时候a就是类A的一个对象，这个对象名字就是a
再来看下面一个例子:
method(A a);
整理method是一个方法，他需要传递一个对象来作为参数，那么这个时候有2种方法：
方法1：
 A a =new A();
  method (a);
方法2：
 method (new A());
方法2中new A()就是一个匿名对象，他没有名字。这样可以理解了吧。

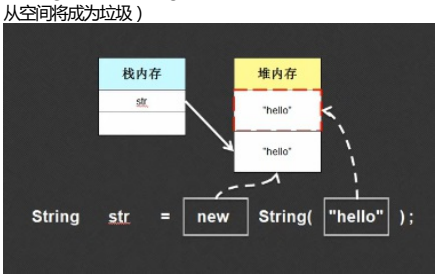
```
public class StringDemo{
  public static void main(String args[]){
  String str="Hello";//给匿名对象起了一个名字str（"Hello"是一个匿名对象）。但str不是对象名称，声明对象并实例化StringDemo strd=new StringDemo();
  System.out.println("Hello".equals(str));
  }
}
public class StringDemo{
  public static void main(String args[]){
  String input="Hello";
  if(input.equals("Hello")){//字符串是否相等 ，用input调用了equals()方法
    System.out.println("Hello World!");
  }
  }
}

public class StringDemo{
  public static void main(String args[]){
  String input=null;
  if("Hello".equals(input)){//如果要判断输入的内容是否某一字符串，请一定要将字符串写在前面。equals()可以回避空指向异常。
    System.out.println("Hello World!");
  }
  }
}
```

第9章 课时35：两种实例化方式的区别

```
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb="hello";
    String strc="hello";
    System.out.println(stra==strb);
    System.out.println(stra==strc);
    System.out.println(strb==strc);
  }
}
//String类直接进行赋值时，值会存入对象池，下次如果再进行String类的直接赋值且相等时，直接引用该栈内存。


//String str=new String("hello");如果使用的是构造方法的方式进行String类对象实例化的时候，程序默认从右往左执行，那么最终的操作形式就变成了开辟2块堆内存空间（其中有一块堆内从空间将成为垃圾）
```




```
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb=new String ("hello");
    System.out.println(stra==strb);
  }
}
```

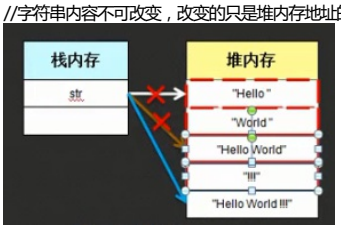
如果采用构造方法定义String类，其值不会入对象池，可以采用String类的一个方法：手工入池。public String intern;

```
public class StringDemo{
  public static void main(String args[]){
    String stra=new String ("hello").intern();//使用了构造方法定义了新的内存空间，而后入池。
    String strb="hello";
    System.out.println(stra==strb);
  }
}
```

第9章 课时36：字符串内容不可改变

```
public class StringDemo{
  public static void main(String args[]){
    String str="Hello ";
    str=str+"World";
    str+="!!!";
    System.out.println(str);
  }
}
```

//字符串内容不可改变，改变的只是堆内存地址的指向，且每一次改变都会产生垃圾空间。如下图




```
public class StringDemo{
  public static void main(String args[]){
    String str="";
    for(int x=0;x<1000;x++){
      str+=x;//任何类型和String类型相加时都会变为String类型，所以str会变成一长串数字字符且产生999块垃圾空间。
    }
    System.out.println(str);
  }
}
```

第10章 课时37：String类常用方法（字符与字符串）

No	方法名称	类型	描述
1	String(char[] value)	构造	将字符数组变为String类对象
2	String(char[] value, int offset, int count)	构造	将部分字符数组变为String
3	public char charAt(int index)	普通	返回指定索引对应的字符信息
4	public char toCharArray()	普通	将字符串以字符数组的形式返回

```
public class StringDemo{
  public static void main(String args[]){
    String str="hello";
    char c=str.charAt(0);//取出指定索引的字符
    System.out.println(c);
  }
}
```

```
//字符数组与字符串的转换
public class StringDemo{
  public static void main(String args[]){
    String str="hello";
    char[] data=str.toCharArray();
    for(int x=0;x<data.length;x++){
      System.out.println(data[x]+"、");
    }
  }
}
```

```
public class StringDemo{
  public static void main(String args[]){
    String str="hello";
    char[] data=str.toCharArray();
    for(int x=0;x<data.length;x++){
    data[x]-=32;
    }
    System.out.println(new String(data));
    System.out.println(new String(data,1,2));
  }
}
```

```
//判断一个字符串是否有数字组成
public class StringDemo{
  public static void main(String args[]){
    String str="1231234a123";
    if(isNumber(str)){
      System.out.println("字符串由数字组成！");
```

```
    }else{
      System.out.println("字符串由非数字组成！");
    }
  }
  public static boolean isNumber(String temp){
    char[] data=temp.toCharArray();
    for(int x=0;x<data.length;x++){
      if(data[x]>'9'||data[x]<'0'){
        return false;
      }
    }
    return true;
  }
}
```

第10章 课时38：String类常用方法（字符与字符串）

```
public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    byte[] data=str.getBytes();
    for(int x=0;x<data.length;x++){
    data[x]-=32;//将小写字母变为大写形式；
    }
    System.out.println(new String(data));
    System.out.println(new String(data,5,5));
  }
}
```

No	方法名称	类型	描述
5	public String(byte[] bytes)	构造	将全部字节数组变为字符串
6	public String(byte[] bytes,int offset,int length)	构造	将部分字节数组变为字符串
7	public byte[] getBytes()	普通	将字符串变为字节数组
8	public byte[] getBytes(String charsetName)throws UnsupportedEncodingException	普通	进行编码转换

第10章 课时39：String类常用方法（字符串比较）

No	方法名称	类型	描述
9	public boolean equals(String anObject)	普通	进行相等判断，它区分大小写
10	equalsIgnoreCase(String anotherString)	普通	进行相等判断，不区分大小写
11	compareTo(String anotherString)	普通	判断两个字符的大小（按照字符编码比较），此方法的返回值有如下三种结果： 1、=0，表示要比较的两个字符串内容相等； 2、>0，表示大于的结果 3、<0，表示小鱼的结果

```
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb="heLLo";
    System.out.println(stra.equals(strb));
    System.out.println(stra.equalsIgnoreCase(strb));
    System.out.println(stra.compareTo(strb));
  }
}
```

第10章 课时40：String类常用方法（字符串查找）

No	方法名称	类型	描述
12	public boolean contains(String s)	普通	判断指定的内容是否存在
13	public int indexOf(String str)	普通	由前向后查找指定字符串的位置，如果查找到了则返回（第一个字母）位置的索引，如果找不到返回-1
14	public int indexOf(String str, int fromIndex)	普通	由指定位置从前向后查找指定字符串的位置，找不到返回-1
15	public int lastIndexOf(String str)	普通	由后向前查找指定字符串的位置，找不到返回-1
16	public int lastIndexOf(String str, int fromIndex)	普通	从指定位置由后向前查找，找不到返回-1
17	public boolean startsWith(String prefix)	普通	判断是否以指定的字符串开头，如果是返回true，否返回false

18	public boolean startsWith(String prefix, int toffset)	普通	从指定位置开始，判断是否以指定的字符串开头，如果是返回true，否返回false
19	public boolean endsWith(String suffix)	普通	判断是否以指定的字符串结尾

```
public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    System.out.println(str.indexOf("world"));
    System.out.println(str.indexOf("l"));
    System.out.println(str.indexOf("l",5));
    System.out.println(str.lastIndexOf("l"));
    System.out.println(str.lastIndexOf("xxxx"));
  }
}
```

```
//自从Java 1.5有了contains后，在整个Java里面contains已经成了查询的代名词。

public class StringDemo{
  public static void main(String args[]){

    String stra="helloworld";
    if(stra.indexOf("world")!=-1){
      System.out.println("可以查询到数据");
    }
    String strb="helloworld";
    if(strb.contains("world")){
      System.out.println("可以查询到数据");
    }
  }
}


public class StringDemo{
  public static void main(String args[]){

    String str="##@@hello**";
    System.out.println(str.startsWith("##"));
    System.out.println(str.startsWith("@@",2));
    System.out.println(str.endsWith("**"));
  }
}
```

第10章 课时41：String类常用方法（字符串替换）

No	方法名称	类型	描述
20	public **String** replaceAll(String regex, String replacement)	普通	用新的内容替换掉全部旧的内容
21	public **String** replaceFirst(String regex, String replacement)	普通	替换首个满足条件的内容

```
//正则，regex
public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    String resultA=str.replaceAll("l","_");
    String resultB=str.replaceFirst("l","_");
    System.out.println(resultA);
    System.out.println(resultB);
  }
}
```

第10章 课时42：String类常用方法（字符串截取）

No	方法名称	类型	描述
22	public String substring(int beginIndex)	普通	从指定索引截取到结尾
23	public String substring(int beginIndex, int endIndex) //1和2重载，substring第二个s没有大写	普通	截取部分子字符串的数据

```
public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    String resultA=str.substring(5);
    String resultB=str.substring(0,5);
    System.out.println(resultA);
    System.out.println(resultB);
  }
}
```

第10章 课时43：String类常用方法（字符串拆分）

No	方法名称	类型	描述
24	public String[] split(String regex)	普通	按照指定的字符串进行全部拆分
25	public String[] split(String regex, int limit)	普通	按照指定的字符串进行部分拆分，最后的数组的长度就是由limit决定的，即：前面拆，后面不拆。

```
public class StringDemo{
  public static void main(String args[]){
    String str="张三：20|李四：21|王五：22|";
    String result []=str.split("\\|");
    for(int x=0;x<result.length;x++){
      String temp[]=result[x].split("：");//  “：” 也要注意中英文一致
      System.out.println("姓名:"+temp[0]+",年龄:"+temp[1]);
    }
  }
}
```

```
public class StringDemo{
  public static void main(String args[]){
    String str="hello world nihao mdln";
    String result[]=str.split(" ",3);//" "和""是不一样的
    for(int x=0;x<result.length;x++){
      System.out.println(result[x]);
    }
  }
}
```

```
public class StringDemo{
  public static void main(String args[]){
    String str="192.168.1.2";
    String result[]=str.split("\\.");//如果不加\\就拆不开，因为regex
    for(int x=0;x<result.length;x++){
      System.out.println(result[x]);
    }
  }
}//如果是一些敏感字符（正则标记）严格来讲是拆分不了的，需要用转译字符\\
```

第10章 课时44：String类常用方法（其他操作）

No	方法名称	类型	描述
26	public String concat（Sting str）	普通	字符串连接，和"+"类似
27	public String toLowerCase()	普通	转小写
28	public String toUpperCase()	普通	转大写
29	public String trim()	普通	去掉字符串左右两边的空格，中间的空格保留
30	public int length()	普通	取得字符串长度
31	public String intern()	普通	数据入池
32	public boolean isEmpty()	普通	判断是否空字符串（不是null，而是""，长度0）

```
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    String strb="hello "+"world";
    String strc="hello world";
    System.out.println(stra==strc);
    System.out.println(strb==strc);
  }
}
```

```
public class StringDemo{
  public static void main(String args[]){
    String stra="hello";
    //String strb=stra+"world";//和下一句效果相同
    String strb=stra.concat("world");
    String strc="hello world";
    System.out.println(stra==strc);
    System.out.println(strb==strc);
  }
}
```

```
 public class StringDemo{
  public static void main(String args[]){
    String str="(*(*Hello(*(*";
    System.out.println(str.toLowerCase());
    System.out.println(str.toUpperCase());//所有的非字母数据不会进行任何的转换操作
  }
}
```

```
public class StringDemo{
  public static void main(String args[]){
```

```
    String str="    hello    world    ";
    System.out.println("【"+str+"】");
    System.out.println("【"+str.trim()+"】");//去掉字符串两端的空格
  }
}

public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    System.out.println(str.length());//和数组对象的区别是多了()
  }
}

public class StringDemo{
  public static void main(String args[]){
    String str="helloworld";
    System.out.println(str.isEmpty());
    System.out.println("".isEmpty());
  }
}

public class StringDemo{
  public static void main(String args[]){
    String str="hEllO";
    System.out.println(initcap(str));
  }
  public static String initcap(String temp){
    return temp.substring(0,1).toUpperCase()+temp.substring(1).toLowerCase();
  }
}
```

No	方法名称	类型	描述
1	String(char[] value)	构造	将字符数组变为String类对象
2	String(char[] value, int offset, int count)	构造	将部分字符数组变为String
3	public char charAt(int index)	普通	返回指定索引对应的字符信息
4	public char toCharArray()	普通	将字符串以字符数组的形式返回
5	public String(byte[] bytes)	构造	将全部字节数组变为字符串
6	public String(byte[] bytes,int offset,int length)	构造	将部分字节数组变为字符串
7	public byte[] getBytes()	普通	将字符串变为字节数组
8	public byte[] getBytes(String charsetName)throws UnsupportedEncodingException	普通	进行编码转换
9	public boolean equals(String anObject)	普通	进行相等判断，它区分大小写
10	equalsIgnoreCase(String anotherString)	普通	进行相等判断，不区分大小写
11	compareTo(String anotherString)	普通	判断两个字符的大小（按照字符编码比较），此方法的返回值有如下三种结果： 1、=0，表示要比较的两个字符串内容相等； 2、>0，表示大于的结果 3、<0，表示小鱼的结果
12	public boolean contains(String s)	普通	判断指定的内容是否存在
13	public int indexOf(String str)	普通	由前向后查找指定字符串的位置，如果查找到了则返回（第一个字母）位置的索引，如果找不到返回-1
14	public int indexOf(String str, int fromIndex)	普通	由指定位置从前向后查找指定字符串的位置，找不到返回-1
15	public int lastIndexOf(String str)	普通	由后向前查找指定字符串的位置，找不到返回-1
16	public int lastIndexOf(String str, int fromIndex)	普通	从指定位置由后向前查找，找不到返回-1
17	public boolean startsWith(String prefix)	普通	判断是否以指定的字符串开头，如果是返回true，否返回false
18	public boolean startsWith(String prefix,int toffset)	普通	从指定位置开始，判断是否以指定的字符串开头，如果是返回true，否返回false
19	public boolean endsWith(String suffix)	普通	判断是否以指定的字符串结尾
20	public String replaceAll(String regex, String replacement)	普通	用新的内容替换掉全部旧的内容
21	public String replaceFirst(String regex, String replacement)	普通	替换首个满足条件的内容
22	public String substring(int beginIndex)	普通	从指定索引截取到结尾
23	public String substring(int beginIndex, int endIndex)//1和2重载，substring第二个s没有大写	普通	截取部分子字符串的数据
24	public String[] split(String regex)	普通	按照指定的字符串进行全部拆分
25	public String[] split(String regex, int limit)	普通	按照指定的字符串进行部分拆分，最后的数组的长度就是由limit决定的，即：前面拆，后面不拆。

26	public String concat (Sting str)	普通	字符串连接，和"+"类似
27	public String toLowerCase()	普通	转小写
28	public String toUpperCase()	普通	转大写
29	public String trim()	普通	去掉字符串左右两边的空格，中间的空格保留
30	public int length()	普通	取得字符串长度
31	public String intern()	普通	数据入池
32	public boolean isEmpty()	普通	判断是否空字符串（不是null，而是""）,长度0

第11章 课时45：this关键字（调用属性）

//在以后的开发中，只要是访问类中的属性前面必须加上"this."