1 Object 对象

面向对象的核心思想:“找合适的对象，做适合的事情”。
合适的对象：

1. 自己描述类，自己创建对象。
2. sun 已经描述了好多常用的类，可以使用这些类创建对象。
   API（Application Program Interface）

sun 定义的那么多类的终极父类是 Object。Object 描述的是所有类的通用属性与方法。

1.1 toString 方法

```
public static void main(String[] args)
{
    Object o = new Object();
    System.out.println(o);  //java.lang.Object@de6ced
}
```

toString() 返回对象的描述信息 java.lang.Object@de6ced 类名@哈希码值的十六进制形式。
直接输入一个对象的时候，会调用对象的 toString 方法。
练习：自定义一个 Person 类，打印该对象的描述信息，要求描述信息为：姓名 — 年龄

```
class Person
{
    String name;
    int age;

    public Person(){

    }

    public Person(String name, int age){
      this.name = name;
      this.age = age;
    }
}

class  Demo1
{
    public static void main(String[] args)
    {
        Person  p = new Person();
        System.out.println(p);  //对象的描述信息：Person@c17164

    }
}
```

问题：调用 p 的 toString 方法时，打印出来的信息是类名+内存地址值。不符合要求。
根据我们之前学的继承，假如父类的指定的功能不能满足要求，那么子类可以复写父类的功能函数。那么该对象再调用 toString()方法时，则会调用子类复写的 toString 方法。

```
class Person
{
    String name;
    int age;

    public Person(){

    }

    public Person(String name, int age){
      this.name = name;
      this.age = age;
    }

    public String toString(){
        return this.name+"-"+this.age;
    }


}
```

编程习惯：开发者要对自定义的类重写 toString()，对对象做详细的说明

1.2 equals 方法

equals() 返回的是比较的结果 如果相等返回 true，否则 false，比较的是对象的内存地址值。

```
Object o = new Object();
Object o1 = new Object();
System.out.println(o.equals(o1)); //false
```

```
Object o = new Object();
Object o1 =o;
System.out.println(o.equals(o1)); //true
```

问题：比较两个人是否是同一个人，根据两个人的名字判断。

```
class Person
{
    String name;
    int age;

    public Person(){

    }

    public Person(String name, int age){
      this.name = name;
      this.age = age;
    }

    public String toString(){
        return this.name+"-"+this.age;
    }
}

class  Demo1
{
    public static void main(String[] args)
    {
        Person p = new Person("张三",1);
        Person p1 = new Person("张三",1);
        System.out.println(p.equals(p1)); //false
```

问题： 如果根据名字去作为判断两个人是否是同一个时，明显 p 与 p1 是同一个人，但是程序输入却不是同一个人。不符合我们现实生活的要求。

解决： 根据我们学的继承中的函数复写，如果父类的函数不能满足我们目前的要求，

那么就可以在子类把该功能复写，达到复合我们的要求。

```
class Person
{
    String name;
    int age;

    public Person(){

    }

    public Person(String name, int age){
      this.name = name;
      this.age = age;
    }

   public boolean equals( Object obj ){
      Person p = null;
      // 类型转换
      if( obj instanceof Person )
          p = (Person)obj;
      // 判断name属性  等两个对象的属性完全相等的时候返回true，否则返回false
      if ( this.name.equals(p.name) )
      {
          return true;
      }
      return false;
   }
```

编程习惯：开发者要对自定义的类重写 equals()，使得比较两个对象的时候比较对象的属性是否相等，而不是内存地址。

1.3 hashCode 方法

hashCode() 返回该对象的哈希码值： 采用操作系统底层实现的哈希算法。 同一个对象的哈希码值是唯一的。

java 规定如果两个对象 equals 返回 true，那么这两个对象的 hashCode 码必须一致。

```java
class Person
{
    String name;
    int age;

    public Person(){

    }

    public Person(String name, int age){
      this.name = name;
      this.age = age;
    }
```

如果重写了equals方法就重写hashCode方法

```java
    public int hashCode(){
       return this.name.hashCode() + this.age;
    }


    public boolean equals( Object obj ){
       Person p = null;
       // 类型转换
       if( obj instanceof Person )
           p = (Person)obj;
       // 判断name属性  等两个对象的属性完全相等的时候返回true，否则返回false
       if ( this.name.equals(p.name) )
       {
           return true;
       }
```

2 String 类

String 类描述的是文本字符串序列。 留言 QQ 写日志。

创建 String 类的对象的两种方式：

1. ""直接赋值法
2. new 关键字法

2.1 字符串对象的比较

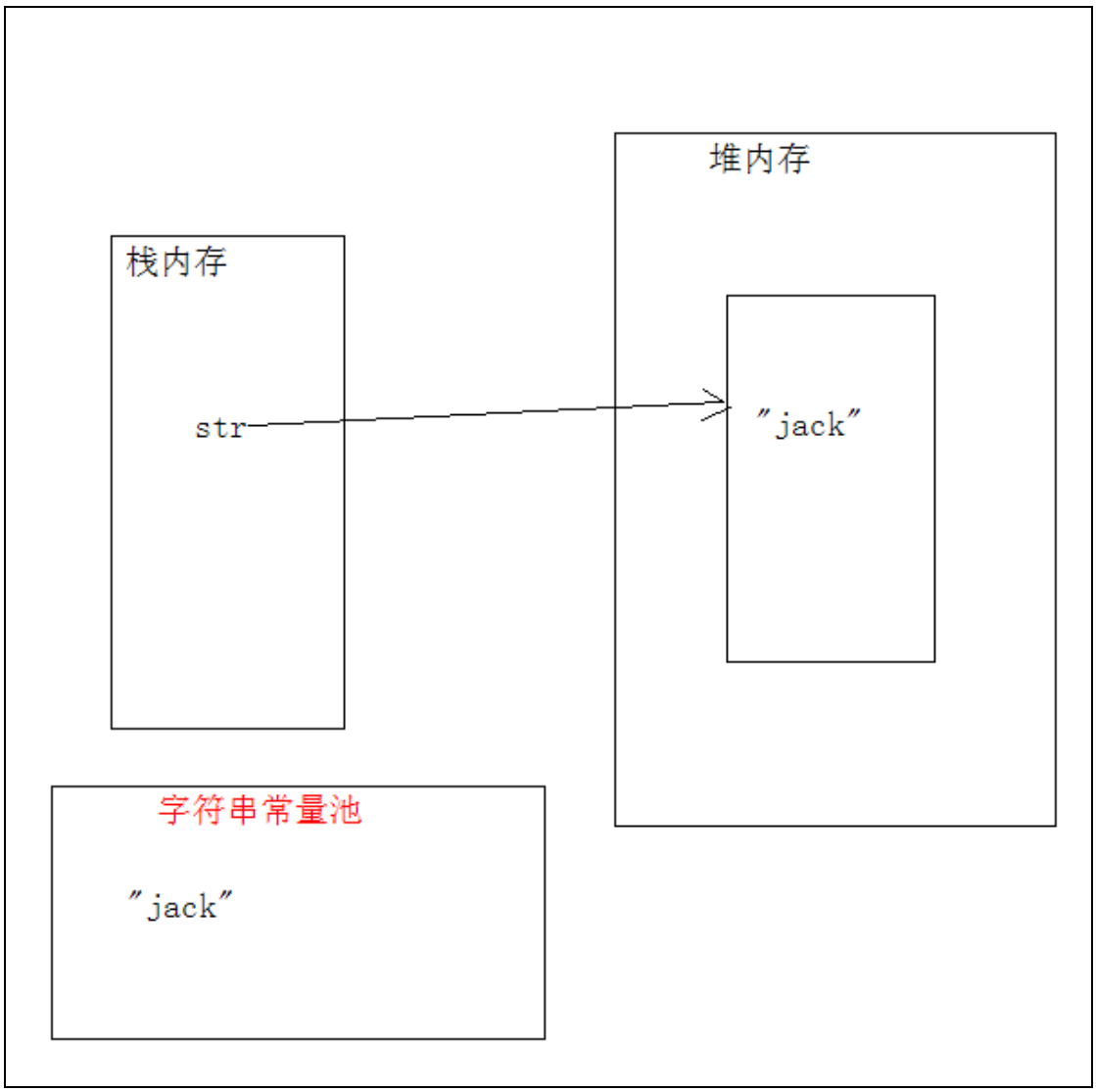
```java
class Demo2
{
    public static void main(String[] args)
    {
        String str1 = "jack";
        String str2 = "jack";
        String str3 = new String("jack");
        String str4 = new String("jack");

        System.out.println( str1 == str2 );    // true
        System.out.println( str1 == str3 );    // false
        System.out.println( str4 == str3 );
    }
}
```

String Str = “jack”这个语句会先检查字符串常量池是否存放这个”jack1”这个字符串对象，如

果没有存在，那么就会在字符串常量池中创建这个字符串对象，如果存在直接返回该字符串的内存地址值。

String str3 = new String(“jack”) 该语句会创建两个对象,首先会先检查字符串常量池中存不存在 jack 这个字符串对象，如果不存在就会创建，如果存在就返回内存地址值。创建了出来之后，new String 这个语句就会在堆内存中开辟一个字符串对象。总共两个对象。



2.2 获取方法

int length()　获取字符串的长度
char charAt(int index) 获取特定位置的字符 (角标越界)
int indexOf(String str) 获取特定字符的位置(overload)
int lastIndexOf(int ch) 获取最后一个字符的位置

```
String str = "hello world";
System.out.println( "length() : "+str.length());       // 11
System.out.println( "indexOf() : "+str.indexOf("l")); // 2
System.out.println( "lastIndexOf() : "+str.lastIndexOf("l"));// 9
System.out.println( "indexOf() : "+str.indexOf("p")); // -1
System.out.println( "charrAt() : "+str.charAt(1));     // e
System.out.println( "charrAt() : "+str.charAt(12));
// java.lang.StringIndexOutOfBoundsException
```

2.3 判断方法

boolean endsWith(String str) 是否以指定字符结束
boolean isEmpty()是否长度为 0 如：“” null V1.6
boolean contains(CharSequences) 是否包含指定序列 应用：搜索
boolean equals(Object anObject) 是否相等
boolean equalsIgnoreCase(String anotherString) 忽略大小写是否相等

```
String str = " ";
System.out.println( "length() : "+str.length());
System.out.println( "isEmpty() : "+str.isEmpty());
// 长度为0返回true否则false  1.6新特性

str = "hello java world";
System.out.println( "contains() : "+str.contains("java"));  // true
System.out.println( "abc".equals("abc") );  // true
//  重写了，比较的是String对象的存储字符内容
System.out.println( new String("abc").equals(new String("abc")) );

System.out.println( new String("abc").equals(new String("Abc")) );   //  false
System.out.println( new String("abc").equalsIgnoreCase(new String("Abc")) ); //
str = "Demo1.java";
System.out.println( "e3ndsWith() : " + str.endsWith(".java") );
```

2.4 转换方法

String(char[] value) 将字符数组转换为字符串
String(char[] value, int offset, int count)
Static String valueOf(char[] data)
static String valueOf(char[] data, int offset, int count)
char[] toCharArray() 将字符串转换为字符数组

```
String str = new String( new char[]{'h','e','l','l','o'} );
System.out.println( str );
char [] chs = str.toCharArray();
for ( int i = 0; i < chs.length ; i++ )
{
        System.out.println( chs[i] );
}

byte[] bs = new byte[]{97,98,99};
String str = new String(bs);
System.out.println( str );

byte [] bytes = str.getBytes();
for ( int i = 0; i < bytes.length ; i++ )
{
     System.out.println(  bytes[i] );
}
```

2.5 其他方法

String replace(char oldChar, char newChar) 替换
String[] split(String regex) 切割
String substring(int beginIndex)
String substring(int beginIndex, int endIndex)截取字串
String toUpperCase() 转大写
String toLowerCase() 转小写
String trim() 去除空格

2.6 练习

1. 去除字符串两边空格的函数。

```
public class Demo1 {
// 定义一个祛除字符串两边空格的函数
public static String trim( String str ){

   // 0、定义求字串需要的起始索引变量
   int start = 0;
   int end = str.length()-1;
   // 1. for循环遍历字符串对象的每一个字符
   for (int i = 0; i<str.length() ; i++ )
   {
       if ( str.charAt(i) == ' ' )
       {
           start++;
       }else{

           break;
       }
   }
   System.out.println( start );
   for (; end<str.length() && end >= 0;  )
   {
       if ( str.charAt(end) == ' ' )
       {
           end--;
       }else{
           break;
       }
   }
   System.out.println( end );
   // 2. 求子串
   if( start < end ){

     return str.substring( start , (end+1) );
   }else{

     return "_";
   }
```

2. 获取上传文件名 "D:\\20120512\\day12\\Demo1.java"。

```
public static String getFileName2( String path ){
      return path.substring( path.lastIndexOf("\\") + 1 );
    }
}
```

3. 将字符串对象中存储的字符反序。

```
// 将字符串对象中存储的字符反序
    public static String reaverseString( String src ){

       // 1. 将字符串转换为字符数组
       char chs[] = src.toCharArray();
       // 2. 循环交换
       for ( int start = 0 , end = chs.length - 1; start < end ; start++,end-- )
       {
           // 3. 数据交换
           char temp = chs[end];
           chs[end] = chs[start];
           chs[start] = temp;
       }
      // 4. 将字符数组转换为字符串
       return new String( chs );
    }
```

4. 求一个子串在整串中出现的次数

```
 public static int getCount( String src , String tag ){
      // 0. 定义索引变量和统计个数的变量
      int index = 0;
      int count = 0;
      // 1. 写循环判断
      while ( ( index = src.indexOf(tag) ) != -1 )   // jackjava
      {
       // 2. 求字串
         System.out.println( src );
         src = src.substring( index + tag.length() );   // index 4 + 4 =
8
         System.out.print( src.length() + " : " + index + " :  " +
       tag.length() );
              // 3. 累加
         count++;
           }
           return count;
        }
```

3 StringBuffer

```
public static void main(String[] args) {
    String str = "";
    for ( int i = 0; i < 100 ; i++ )
    {
        str+=i;
    }
    System.out.println( str );
}
```

StringBuffer ：由于 String 是不可变的，所以导致 String 对象泛滥，在频繁改变字符串对象的应用中，需要使用可变的字符串缓冲区类。

特点：

1. 默认缓冲区的容量是 16。
2. StringBuffer ： 线程安全的所有的缓冲区操作方法都是同步的。效率很低。

3.1 添加方法

StringBuffer("jack")	在创建对象的时候赋值
append()	在缓冲区的尾部添加新的文本对象
insert()	在指定的下标位置添加新的文本对象

```
StringBuffer sb = new StringBuffer("jack");
sb.append(true);
sb.append('a');
sb.append(97).append(34.0).append(new char[]{'o','o'});   // 链式编程
System.out.println( sb.toString() );          // 输出缓冲区的中文本数据
sb = new StringBuffer("jack");
sb.insert( 2, "java" );                      // jajavack
System.out.println( sb.toString() );
```

3.2 查看

toString() 返回这个容器的字符串

indexOf(String str) 返回第一次出现的指定子字符串在该字符串中的索引。

substring(int start) 从开始的位置开始截取字符串

```java
public static void main(String[] args) {
    StringBuffer sb = new StringBuffer("jackc");
    System.out.println( sb.indexOf("c") );    // 2
    System.out.println( sb.lastIndexOf("c") );
}
```

3.3 修改(U)

replace(int start int endString str)　使用给定 String 中的字符替换此序列的子字符串中的字符。该子字符串从指定的 start 处开始，一直到索引 end - 1 处的字符
setCharAt(int index char ch)　指定索引位置替换一个字符

```java
StringBuffer sb = new StringBuffer("helloworld");
System.out.println( sb.replace( 4 , 7 , "java" ) );  // helljavarld  不包含结束索引下标
sb.setCharAt( 8, 'Q' );
System.out.println( sb );  //  // helljavaQld
```

3.4 删除(D)

```
删除(D)
delete (int start, int end)        start <=  char < end
    清空缓冲区:  delete(0, sb.length())
deleteCharAt(int index)
```

```java
StringBuffer sb = new StringBuffer("helloworld");
System.out.println( sb.delete( 2, 5 ) ); // heworld
//System.out.println( sb.delete( 0, sb.length() ) );
System.out.println( sb.deleteCharAt(2) ); // heorld
```

3.5 反序

reverse() 把字符串反序输出。

```java
String str = "helloworld";
StringBuffer sb = new StringBuffer(str);
System.out.println( sb.reverse() );
```

4 StringBuilder

StringBuilder 是 JDK1.5 之后提出的，线程不安全，但是效率要高。用法与 StringBuffer 类似。

5 System

System 可以获取系统的属性。

```
// 获取系统属性
Properties ps = System.getProperties();
// 输出系统属性
ps.list( System.out );

// 获取操作系统名称
String os_name =  System.getProperty("os.name");
System.out.println(os_name);     // Windows XP

// 检测操作系统的是否支持该软件
if("Windows XP".equals(os_name))
{
    System.out.println("继续安装");
}else{
    System.out.println("系统不兼容.....");
}
// 获取path环境变量值
System.out.println( System.getenv("path") );

//static long currentTimeMillis()     返回以毫秒为单位的当前时间。
//static void exit(int status)        退出JVM 0为正常退出
```

6 Runtime

Runtime 类主要描述的是应用程序运行的环境。

```
exit()              退出虚拟机
long freeMemory()   获取可用内存数
gc()                调用垃圾回收器程序，但是调用该方法的时候不会马上就运行GC。
long maxMemory()    获取JVM最大内存容量
long totalMemory()获取总内存
Process exec(String command)    启动一个字符串命令的进程
```

```
// 获取应用运行环境的对象
Runtime run = Runtime.getRuntime();
// 获取可用的内存数
System.out.println( run.freeMemory() );
// 获取JVM试图使用的内存的总容量
System.out.println( run.maxMemory() );
// 获取JVM只能使用的总容量
System.out.println( run.totalMemory() );

Process notepad = run.exec("notepad Demo8.java" );
Process qq =  run.exec("C:\\Program Files\\Tencent\\QQ\\Bin\\QQ.exe");

Thread.sleep(1000*10);

notepad.destroy();
```

7 Date

Date 类封装的是系统的当前时间.。但是 Date 已经过时了，sun 推荐使用 Calendar 类。

Calendar: 该类是一个日历的类，封装了年月日时分秒时区。

```
Date date = new Date();
Calendar calendar = Calendar.getInstance();
// 获取年，月，日，时，分，秒
int year = calendar.get(Calendar.YEAR);
int month = calendar.get(Calendar.MONTH)+1;
int day = calendar.get(Calendar.DAY_OF_MONTH);
int dayofweek = calendar.get(Calendar.DAY_OF_WEEK);

int hour = calendar.get(Calendar.HOUR_OF_DAY);
int minute = calendar.get(Calendar.MINUTE);
int second = calendar.get(Calendar.SECOND);

System.out.println( date );
System.out.println( year );
System.out.println( month );
System.out.println( day );
System.out.println( dayofweek );   // 1 星期天  2 一  3 二 4 三  5 四  6 五  7 六
System.out.println( hour );
System.out.println( minute );
System.out.println( second );
```

日期格式化类：SimpleDateFormat

```
SimpleDateFormat sm = new SimpleDateFormat("yyyy年MM月dd日 E a hh时mm分ss秒");
System.out.println( sm.format( new Date() ) );
```

8 Math

Math：类封装了很多数学的功能。

```
static double ceil(double a) : 返回大于等于指定小数的最小整数
static double floor(double a): 返回小于等于指定小数的最大整数
static long round(double a)  : 四舍五入
static double random()       : 返回大于等于0.0 小于1.0的小数   1.0<= x < 11.0
```

```
System.out.println( Math.PI );
System.out.println( Math.ceil(12.3) ); // 13.0
System.out.println( Math.ceil(12.5) ); // 13.0
System.out.println( Math.ceil(-12.5) );// -12.0

System.out.println( Math.floor(-15.1) );// -16.0
System.out.println( Math.floor(15.1) ); // 15.0

System.out.println( Math.round(15.1) ); // 15
System.out.println( Math.round(15.5) ); // 16

System.out.println( Math.random() );
```

练习：生成一个随机码

```
// 生成一个校验码
Random ran = new Random();
char chs[] = new char[]{'a','b','c','F','H','3','6','中','过','你','好','@'};
StringBuilder rel = new StringBuilder("");
for (int i = 0; i < 4 ; i++ )
{
    rel.append( chs[ran.nextInt(chs.length)] );
}
System.out.println( "校验码是 :  "+rel.toString() );
```