**每日作业卷答案**

**就业班JavaSE第1天**

传智播客.黑马程序员

**基础题**

### 练习一：Object类equals方法

1. 简述String类中的equals方法与Object类中的equals方法的不同点。

答：String类中的equals方法是用来判断两个对象的内容是否相同，而Object 类中的equals方法是用来判断两个对象是否是同一个对象，所谓同一个对象指的是内存中的同一块存储空间。

### 练习二：Object类的toString方法

1. 不运行代码，直接说出打印结果，并解释原因。

**public class** ToStringTest{  
 **static int** i = 1;  
 **public static void** main(String args[]){  
 System.**out**.println(**"love "** + **new** ToStringTest());//love java  
 ToStringTest a = **new** ToStringTest();  
 a.i++;  
 System.**out**.println(**"me "** + a.i);//me 2  
 }  
 **public** String toString(){  
 System.**out**.print(**"I "**);//I  
 **return "java "**;  
 }  
}

运行结果：I love java me 2

原因：当执行代码的时候，首先加载静态变量，然后执行main方法，由于main方法内部第一行代码为输出语句，里面new了此类对象，当执行此行代码时会先创建了本类的对象，由于此类重写了toString方法，会先执行toString方法的打印输出，然后返回“java ”，再执行main方法第一行打印输出。在Java中“System.out.println(类对象名);”实际输出的是该对象的toString()方法返回的字符串，即括号中的内容等价于类对象名.toString(),toString方法的好处是在碰到println方法的时候会被自动调用，不用显示的写出来。

### 练习三：Object类equals方法

1. 看下列程序，不运行说结果，写出答案后，并在IntelliJ IDEA中运行看看自己给的答案与运行结果是否正确，并分析原因。

(1)

String s1 = **new** String(**"abc"**);  
 String s2 = **"abc"**;  
 System.**out**.println(s1 == s2); //false  
 System.**out**.println(s1.equals(s2)); //true

(2)

String s1 = **"abc"**;  
 String s2 = **"abc"**;  
 System.**out**.println(s1 == s2); //true  
 System.**out**.println(s1.equals(s2)); //true

(3)

String s1 = **"a"** + **"b"** + **"c"**;  
 String s2 = **"abc"**;  
 System.**out**.println(s1 == s2); //true  
 System.**out**.println(s1.equals(s2)); //true

(4)

String s1 = **"ab"**;  
 String s2 = **"abc"**;  
 String s3 = s1 + **"c"**;  
 System.**out**.println(s3 == s2); //false  
 System.**out**.println(s3.equals(s2)); //true

### 练习四：StringBuilder类与String类的区别

1. 简述StringBuilder类与String类的区别。

答：String类的对象内容不可改变，所以每当进行字符串拼接时，总是会在内存中创建一个新的对象，所以经常改变内容的字符串最好不要用String，因为每次生成对象都会对系统性能产生影响。

StringBuilder又称为可变字符序列，是JDK5.0中新增加的一个类，它是一个类似于String的字符串缓冲区，通过某些方法调用可以改变该序列的长度和内容。即它是一个容器，容器中可以装很多字符串，并且能够对其中的字符串进行各种操作。它的内部拥有一个数组用来存放字符串内容，进行字符串拼接时，直接在数组中加入新内容，StringBuilder会自动维护数组的扩容。

### 练习五：Date类的使用

1. 请用代码实现:获取当前的日期,并把这个日期转换为指定格式的字符串,如2088-08-08 08:08:08。

**public class** DateTest {  
 **public static void** main(String[] args) {  
 //获取当前日期对象 now;  
 Date now = **new** Date();  
 //创建SimpleDateFormat对象 df,并制定日期格式  
 SimpleDateFormat df = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);  
 //调用df的format(Date date) 方法,传入now; 接收返回的字符串  
 String datestr = df.format(now);  
 //打印这个字符串  
 System.**out**.println(datestr);  
 }  
}

### 练习六：DateFormat类方法的使用

1. 使用SimpleDateFormat类,把2018-03-04转换为2018年03月04日。

**public class** DateFormatTest {  
 **public static void** main(String[] args) **throws** ParseException {  
 //创建SimpleDateFormat对象df1,指定日期模式为yyyy-MM-dd  
 SimpleDateFormat df1 = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  
 //调用df1的parse(String str)方法传入2018-03-04,得到对应日期类型  
 Date date = df1.parse(**"2018-03-04"**);  
 //创建日期格式化对象df2,在获取格式化对象时可以指定风格  
 DateFormat df2 = **new** SimpleDateFormat(**"yyyy年MM月dd日"**);  
 //调用df2的format(Date date) 传入刚才转换的日期  
 String str = df2.format(date);  
 System.**out**.println(str);  
 }  
}

### 练习七：Calendar类方法的使用

1. 用程序判断2018年2月14日是星期几。

**public class** CalendarTest01 {  
 **public static void** main(String[] args) {  
 //创建Calendar对象  
 Calendar c = Calendar.getInstance();  
 //将给定的日历字段设置到Calendar对象中  
 c.set(Calendar.**YEAR**, 2018);  
 c.set(Calendar.**MONTH**, 1);  
 c.set(Calendar.**DATE**, 14);  
 //设置年  
 **int** year = c.get(Calendar.**YEAR**);  
 //设置月  
 **int** month = c.get(Calendar.**MONTH**)+1;  
 //设置日  
 **int** date = c.get(Calendar.**DATE**);  
 //设置星期  
 **char** week = getWeek(c.get(Calendar.**DAY\_OF\_WEEK**));  
 //输出结果  
 System.**out**.println(year+**"年"**+month+**"月"**+date+**"日是星期"**+week);  
 }  
 //定义方法，获取星期汉字  
 **public static char** getWeek(**int** a){  
 **char**[] c = {**' '**,**'日'**,**'一'**,**'二'**,**'三'**,**'四'**,**'五'**,**'六'**};  
 **return** c[a];  
 }  
}

## 扩展题

### 练习八：System类arraycopy方法的使用

1. 现有一个字符数组{'i','t','c','a','s','a'}，请使用System类中的arraycopy()方法在控制台输出“itcast”。（提示：将[1]号数组元素复制到最后位置并覆盖原有元素。）

**public class** ArraycopyTest {  
 **public static void** main(String[] args) {  
 **char**[] cha ={**'i'**,**'t'**,**'c'**,**'a'**,**'s'**,**'a'**};  
 //将cha数组中第2个元素，复制到目标数组最后一位数组上  
 System.arraycopy(cha, 1, cha, 5, 1);  
 //遍历目标数组，在控制台输出字符串  
 **for** (**int** i = 0; i < cha.**length**; i++) {  
 System.**out**.print(cha[i]);  
 }

}  
}

### 练习九：StringBuilder类的使用

1. 请使用代码实现

分别使用String的 += 和StringBuilder的append方法对字符串做100000次拼接，计算String拼接100000次花费时间与StringBuilder拼接100000次所花费时间并打印。

**public class** StringBuilder01 {  
 **public static void** main(String[] args) {  
 //使用System的currentTimeMillis()方法获取当前操作系统的毫秒值,作用程序执行的开始时间,使用start变量接收  
 **long** start = System.currentTimeMillis();  
 //需要测试执行性能的代码  
 //testString(); //消耗时间: 29295毫秒  
 testStringBuilder();//消耗时间:6毫秒  
 //使用System的currentTimeMillis()方法获取当前操作系统的毫秒值,作用程序执行的结束时间,使用end变量接收  
 **long** end = System.currentTimeMillis();  
 //计算代码执行花费的时间 end - start,输出代码执行消耗的时间  
 System.**out**.println(**"所花费的时间为:"**+(end-start));  
 }  
  
 //写一个静态方法: testString(),在该方法中  
 **public static void** testString(){  
 //定义一个字符串 str,内容随意  
 String str = **"hello"**;  
 //写一个循环100000次for循环,在循环中写上  
 **for**(**int** i = 0; i<100000;i++){  
 //str +=”随机内容” ; 这里不要写str += (str+”xxx”) 这样会导致堆内存溢出错误.  
 str += **"world"**;  
 }  
 }  
 //写一个静态方法:testStringBuilder(),在方法中  
 **public static void** testStringBuilder(){  
 //创建一个StringBuilder对象sb,初始内容与testString()中的字符串相同  
 StringBuilder sb = **new** StringBuilder(**"hello"**);  
 //写一个循环100000次for循环,在循环中写上  
 **for**(**int** i = 0; i<100000;i++){  
 //调用sb.append()方法,传入的内容与testString()方法中+=后面的内容一样  
 sb.append(**"world"**);  
 }  
 //循环结束调用sb.toString()方法转换为字符串  
 String newStr = sb.toString();  
 }  
}

### 练习十：StringBuilder类的使用（reverse()方法）

1. 分析以下需求，并用代码实现：

(1)定义数字字符串数组{"010","3223","666","7890987","123123"}；

(2)判断该数字字符串数组中的数字字符串是否是对称(第一个数字和最后一个数字相等，第二个数字和倒数第二个数字是相等的，依次类推)的，并逐个输出；

(3)如：010 是对称的，3223 是对称的，123123 不是对称的；

(4)最终打印该数组中对称字符串的个数。

**注：判断对称可用reverse(),将此字符序列用其反转形式取代。**

**public class** StringBuilderTest02 {  
 **public static void** main(String[] args) **throws** IOException {  
 //定义数字字符串数组  
 String[] str = {**"010"**,**"3223"**,**"666"**,**"7890987"**,**"123123"**};  
 SBTest(str);  
 }  
  
 **public static void** SBTest(String[] str) {  
 **int** count = 0;  
 //遍历定义的字符串数组  
 **for** (String string : str) {  
 //创建StringBuilder对象  
 StringBuilder sb = **new** StringBuilder(string);  
 //调用reverse()方法，将遍历的数字进行反转，然后用equals()方法对比是否与原数字相同  
 **if** (sb.reverse().toString().equals(string)) {  
 count++;  
 System.**out**.println(string + **"是对称的"**);  
 }  
 }  
 System.**out**.println(**"总数为"** + count);  
 }  
}

### 练习十一：StringBuilder类的使用

1. 分析以下需求，并用代码实现：

(1)打印由7，8，9三个数组成的三位数，要求该三位数中任意两位数字不能相同；

(2)打印格式最后的三位数字以空格分隔，如789 798 879 897 978 987。

**注：要求使用StringBuilder来完成**

**public class** StringBuilderTest03 {  
 **public static void** main(String[] args) **throws** IOException, Exception {  
 SBTest();  
 }  
  
 **public static void** SBTest() {  
 //定义由7、8、9组成的字符串  
 String s = **"789"**;  
 //创建StringBuilder对象  
 StringBuilder sb = **new** StringBuilder();  
 //采用嵌套for循环，遍历字符串  
 **for** (**int** i = 0; i < 3; i++) {  
 //遍历字符串s，把字符依次添加到StringBuilder内，组成一个元素  
 **for** (**int** j = 0; j < s.length(); j++) {  
 sb.append(s.charAt(j));  
 }  
 //当StringBuilder内元素为一个和两个时，需要加空格来隔开元素  
 **if** (i != 2) {  
 sb.append(**" "**);  
 }  
 //把字符串s的第一个元素切割，添加到字符串末位，组成新的字符串  
 s = s.substring(1).concat(s.substring(0, 1));  
 }  
 // 把StringBuilder内元素反转，组成新的数字  
 s = sb.toString() + **" "** + sb.reverse().toString();  
 System.**out**.println(s);  
 }  
}