**常用工具类作业**

1. **填空题**
2. 在Java中每个Java基本类型在java.lang包中都在一个相应的包装类，把基本类型数据转换为对象，其中包装类Integer是\_\_\_number\_\_\_\_\_\_\_的直接子类。
3. 包装类Integer的静态方法可以将字符串类型的数字”123”转换成基本整型变量n，其实现语句是：\_\_\_\_Integer.paseInt(“123”)\_\_\_\_\_\_\_。
4. 在Java中使用java.lang包中的\_\_\_StringBuffer\_\_\_\_\_\_\_\_\_类来创建一个字符串对象，它代表一个字符序列可变的字符串，可以通过相应的方法改变这个字符串对象的字符序列。
5. StringBuilder类是StringBuffer类的替代类，两者的共同点是都是可变长度字符串，其中线程安全的类是\_\_\_\_\_\_StringBuffer\_\_\_\_\_\_\_\_。
6. DateFormat类可以实现字符串和日期类型之间的格式转换，其中将日期类型转换为指定的字符串格式的方法名是\_\_\_\_\_\_\_.format()\_\_\_\_\_。
7. 使用Math.random( )返回带正号的 double值，该值大于等于0.0且小于1.0。使用该函数生成[30,60]之间的随机整数的语句是\_ (int)(Math.random()\*31+30)\_\_\_。
8. JDK1.5后提供了\_\_\_\_\_ Enum\_\_\_\_\_\_关键字，用以定义枚举类。枚举类是一种特殊的类，可以有自己的属性、方法和构造方法。
9. File对象调用方法 mkdirs 创建一个目录，不包括所有必需但不存在的父目录，当且仅当已创建目录时，返回true；否则返回false。
10. 将字符串”123”转换成基本数据类型。\_\_\_\_Integer.paseInt(“123”) \_\_\_\_\_\_\_\_\_\_\_.
11. String类的trim()方法作用是 忽略前导空白和尾部空白 。
12. "hamburger".substring(4, 8) 返回的结果是 urge 。
13. String s = "a"+”b”+”c” 创建 0 个对象。
14. System.currentTimeMillis()表示 返回以毫秒为单位的当前时间 。
15. **选择题**

|  |  |  |
| --- | --- | --- |
| **1.** | **以下选项中关于int和Integer的说法错误的是（ b d ）。（选择二项）** | |
|  |  |  |
|  | **A.** | int是基本数据类型，Integer是int的包装类，是引用数据类型 |
|  | **B.** | int的默认值是0，Integer的默认值也是0 |
|  | **C.** | Integer可以封装了属性和方法提供更多的功能 |
|  | **D.** | Integer i=5;该语句在JDK1.5之后可以正确执行，使用了自动拆箱功能 |

|  |  |  |
| --- | --- | --- |
| **2.** | **分析如下Java代码，该程序编译后的运行结果是（ d ）。（选择一项）** | |
|  | **public** **static** **void** main(String[ ] args) {  String str=**null**;  str.concat("abc");  str.concat("def");  System.*out*.println(str);  } | |
|  |  |  |
|  | **A** | null |
|  | **B.** | abcdef |
|  | **C.** | 编译错误 |
|  | **D.** | 运行时出现NullPointerException异常 |

|  |  |  |
| --- | --- | --- |
| **3.** | **以下关于String类的代码的执行结果是（ b ）。（选择一项）** | |
|  | **public** **class** Test2 {  **public** **static** **void** main(String args[]) {  String s1 = **new** String("bjsxt");  String s2 = **new** String("bjsxt");  **if** (s1 == s2) System.*out*.println("s1 == s2");  **if** (s1.equals(s2)) System.*out*.println("s1.equals(s2)");  }  } | |
|  |  |  |
|  | **A.** | s1 == s2 |
|  | **B.** | s1.equals(s2) |
|  | **C.** | s1 == s2  s1.equals(s2) |
|  | **D.** | 以上都不对 |

|  |  |  |
| --- | --- | --- |
| **4.** | **以下关于StringBuffer类的代码的执行结果是（ d ）。（选择一项）** | |
|  | **public** **class** **TestStringBuffer {**  **public** **static** **void** main(String args[]) {  StringBuffer a = **new** StringBuffer("A");  StringBuffer b = **new** StringBuffer("B");  *mb\_operate*(a, b);  System.*out*.println(a + "." + b);  }  **static** **void** mb\_operate(StringBuffer x, StringBuffer y) {  x.append(y);  y = x;  }  } | |
|  |  |  |
|  | **A.** | A.B |
|  | **B.** | A.A |
|  | **C.** | AB.AB |
|  | **D.** | AB.B |

|  |  |  |
| --- | --- | --- |
| **5.** | **给定如下Java代码，编译运行的结果是（ c ）。（选择一项）** | |
|  | **public** **static** **void** main(String []args){  String s1= **new** String("pb\_java\_OOP\_T5");  String s2 = s1.substring(s1.lastIndexOf("\_"));  System.*out*.println("s2="+s2);  } | |
|  |  |  |
|  | **A** | s2=\_java\_OOP\_T5 |
|  | **B.** | s2=\_OOP\_T5 |
|  | **C.** | s2=\_T5 |
|  | **D.** | 编译出错 |

|  |  |  |
| --- | --- | --- |
| **6.** | **对于语句String s="my name is kitty"，以下选项中可以从其中截取”kitty”的是（ ab ）（选择二项）** | |
|  |  |  |
|  | **A** | s.substring(11,16) |
|  | **B.** | s.substring(11) |
|  | **C.** | s.substring(12,17) |
|  | **D.** | s.substring(12,16) |

|  |  |  |
| --- | --- | --- |
| **7.** | **分析下面的Java程序段，编译运行后的输出结果是（d ）。（选择一项）** | |
|  | **public** **class** Test {  **public** **void** changeString(StringBuffer sb) {  sb.append("stringbuffer2");  }  **public** **static** **void** main(String[] args) {  Test a = **new** Test();  StringBuffer sb = **new** StringBuffer("stringbuffer1");  a.changeString(sb);  System.*out*.println("sb = " + sb);  }  } | |
|  |  |  |
|  | **A** | sb = stringbuffer2stringbuffer1 |
|  | **B.** | sb = stringbuffer1 |
|  | **C.** | sb = stringbuffer2 |
|  | **D.** | sb = stringbuffer1stringbuffer2 |

|  |  |  |
| --- | --- | --- |
| **8.** | **给定如下Java代码，编译运行的结果是（ a ）。（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {  StringBuffer sbf = **new** StringBuffer("java");  StringBuffer sbf1 = sbf.append(",C#");  String sbf2 = sbf + ",C#";  System.*out*.print(sbf.equals(sbf1));  System.*out*.println(sbf2.equals(sbf));  } | |
|  |  |  |
|  | **A** | true false |
|  | **B.** | true true |
|  | **C.** | false false |
|  | **D.** | false true |

|  |  |  |
| --- | --- | --- |
| **9.** | **分析下面的Java程序，编译运行后的输出结果是（ b ）。（选择一项）** | |
|  | **public** **class** Example {  String str = **new** String("good");  **char**[] ch = { 'a', 'b', 'c' };  **public** **static** **void** main(String args[]) {  Example ex = **new** Example( );  ex.change(ex.str, ex.ch);  System.*out*.print(ex.str + "and");  System.*out*.print(ex.ch);  }  **public** **void** change(String str, **char** ch[]) {  str = "test ok";  ch[0] = 'g';  }  } | |
|  |  |  |
|  | **A** | goodandabc |
|  | **B.** | goodandgbc |
|  | **C.** | test okandabc |
|  | **D.** | test okandgbc |

|  |  |  |
| --- | --- | --- |
| **10.** | **以下程序片段中可以正常编译的是（ c ）。（选择一项）** | |
|  |  |  |
|  | **A** | String s = "Gone with the wind";  String k = s+t;  String t = "good"; |
|  | **B.** | String s = "Gone with the wind";  String t;  t = s[3]+"one"; |
|  | **C.** | String s = "Gone with the wind";  String stanfard = s.toUpperCase(); |
|  | **D.** | String s = "home directory";  String t = s – "directory"; |

|  |  |  |
| --- | --- | --- |
| **11.** | **File类中的（ b ）方法可以用来判断文件或目录是否存在。（选择一项）** | |
|  |  |  |
|  | **A** | exist() |
|  | **B.** | exists() |
|  | **C.** | fileExist() |
|  | **D.** | fileExists() |
| **12.** | **在Java中，以下File类的方法中（ c ）用来判断是否是目录。（选择一项）** | |
|  |  |  |
|  | **A** | isFile( ) |
|  | **B.** | getFile( ) |
|  | **C.** | isDirectory( ) |
|  | **D.** | getPath( ) |

|  |  |  |
| --- | --- | --- |
| **13.** | **分析下面的Java程序，编译运行后的输出结果是（ b ）。（选择一项）** | |
|  | **public** **class** Example {  String str = **new** String("good");  **char**[] ch = { 'a', 'b', 'c' };  **public** **static** **void** main(String args[]) {  Example ex = **new** Example( );  ex.change(ex.str, ex.ch);  System.*out*.print(ex.str + "and");  System.*out*.print(ex.ch);  }  **public** **void** change(String str, **char** ch[]) {  str = "test ok";  ch[0] = 'g';  }  } | |
|  |  |  |
|  | **A** | goodandabc |
|  | **B.** | goodandgbc |
|  | **C.** | test okandabc |
|  | **D.** | test okandgbc |

|  |  |  |
| --- | --- | --- |
| **14.** | **分析下面代码的结果（ a）。（选择一项）** | |
|  | **public** **static** **void** main(String args[]) {  String s = "abc";  String ss = "abc";  String s3 = "abc" + "def"; // 此处编译器做了优化!  String s4 = "abcdef";  String s5 = ss + "def";  String s2 = **new** String("abc");  System.***out***.println(s == ss);  System.***out***.println(s3 == s4);  System.***out***.println(s4 == s5);  System.***out***.println(s4.equals(s5));  } | |
|  |  |  |
|  | **A** | true true false true |
|  | **B.** | true true true false |
|  | **C.** | true false true true |
|  | **D.** | false true false true |

1. **判断题**
2. 方法Integer.parseInt()的作用是将一个整数转变成String。（ f ）
3. JK1.5后提供了自动装箱和自动拆箱功能，从而可以实现基本数据类型和对应包装类之间的自动转换，简化了操作。（ f ）
4. 执行语句String str="abcedf"; int len=str.length; 后，能够得到字符串的长度是6。（ t ）
5. 运算符“==”用于比较引用时，如果两个引用指向内存同一个对象，则返回true。（ t ）
6. java.sql.Date类和java.util.Date类的关系是前者是后者的父类，其中前者没有提供无参数构造方法，而后者可以提供无参数构造方法来获取当前时间。（ f ）
7. 求x的y次方，其表达式为：Math.pow(x,y)。（ t）
8. 一个File对象可以代表一个文件或目录，它可以获取文件和目录属性，也可以访问文件内容。( t )
9. 在使用File类中的delete( )方法时，删除时可能文件不存在，所以我们最好先判断一下是否存在，不然会出现NullPointerException异常。（ t ）
10. Date d = new Date()表示的是当前时间。（ t）
11. 递归可以完全使用迭代来代替。（ f ）
12. **简答题**
13. 自动装箱和自动拆箱

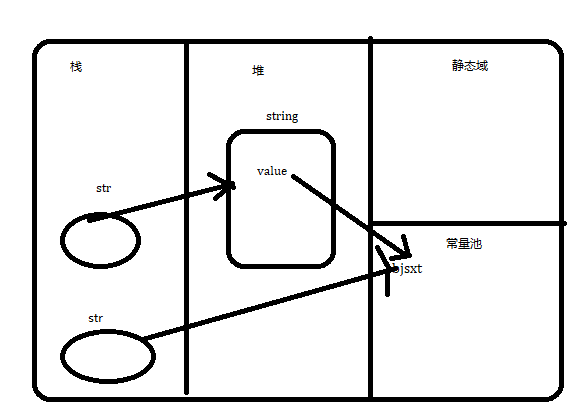
JDK1.5提供了自动装箱（autoboxing）和自动拆箱（autounboxing）功能, 从而实现了包装类和基本数据类型之间的自动转换

1. String、StringBuffer、StringBuilder区别与联系。

String:长度不可变，是final类修饰的，不可继承

Stringbuffer：长度可变，线程安全，效率低

Stringbuilder：长度可变，线程不安全，效率高

1. String str=”bjsxt”;和String str= new String(“bjsxt”);的区别
2. java.sql.Date和java.util.Date的联系和区别

java.sql.date是java.util.date是子类，java.util.Date 就是Java的日期对象，而java.sql.Date 是针对SQL语句使用的，只包含日期而没有时间部分

1. 为什么要使用包装类，包装类的作用。

包装类使基本数据类型的数据变成了类，可以调用类的方法，有了类的特点

1. File类的方法mkdir跟mkdirs，有什么区别?
2. 简述枚举的使用。
3. 递归算法的优点是什么?缺点是什么?
4. **编码题**
5. 验证键盘输入的用户名不能为空，长度大于6，不能有数字。

提示：使用字符串String类的相关方法完成

**package** com.uek.test.homework;

**import** java.util.Scanner;

**import** com.sun.beans.editors.IntegerEditor;

**public** **class** **Test01** {

/\*

\* 1. 验证键盘输入的用户名不能为空，长度大于6，不能有数字。

\* 提示：使用字符串String类的相关方法完成

\*/

**public** **static** **void** **main**(**String**[] args) {

**Scanner** sc = **new** Scanner(**System**.***in***);

**System**.***out***.println("输入用户名");

**String** name = sc.next();

**if**(name.length()<6) {

**System**.***out***.println("用户名不能小于6");

**return**;

}

**if**("".equals(name)) {

**System**.***out***.println("用户名不能为空");

**return**;

}

**for** (**int** i = 0; i <name.length(); i++) {

**char** c = name.charAt(i);

**if**(c <='9' && c >='0') {

**System**.***out***.println("不能有数字");

**return**;

}

}

**System**.***out***.println("用户名符合要求");

}

}

1. 接收从键盘输入的字符串格式的年龄，分数和入学时间，转换为整数、浮点数、日期类型，并在控制台输出。

提示：使用包装类Integer、Double和日期转换类DateFormat实现

1. 根据交通信号灯颜色决定汽车停车、行驶和慢行

提示：使用枚举实现

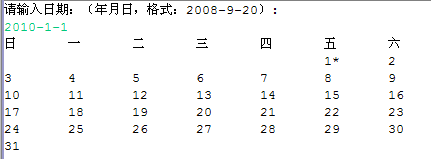
1. 以树状结构输出计算机某个指定文件夹下的所有的文件和子文件夹名称。

提示：使用File的方法，并结合递归实现

1. 将1990年3月3日通过Calendar来表示，并得出这天是该年的第几天?将该日期增加35天，是哪一天?使用代码来说明。
2. **可选题**
3. 生成10个[10,23)之间的随机整数

提示：分别使用Math.random()和Random类的nextDouble()或nextInt()实现

1. 手动定义一个枚举，表示十二个月的英文月份。
2. 打印某个月份的可视化日历



提示：使用DateFormat、Calendar类实现功能