### Java第一阶段—DAY14-JAVA案例

1. 练习String类的相关API。

public class Demo01String {  
  
 public static void main(String[] args) {  
 String name =new String("haley") ;  
 System.out.println(name.length()); *//长度为5* System.out.println(name.equals("Haley")); *//false* System.out.println(name.equalsIgnoreCase("Haley")); *//true* System.out.println(name.toLowerCase()); *//haley* System.out.println(name.toUpperCase()); *//HALEY* System.out.println(name.concat("helay")); *//hello helay* System.out.println("hello "+"helay"); *//hello helay* String word ="hello , welcome to bailiban! ";  
 *// 返回字符97第一次出现的位置* System.out.println(word.indexOf(97)); *//20  
 // 返回字符串ba第一次出现的位置* System.out.println(word.indexOf("ba")); *//19  
 // 返回字符97最后一次出现的位置* System.out.println(word.lastIndexOf(97)); *//25  
 // 返回字符串ba最后一次出现的位置* System.out.println(word.lastIndexOf("ba")); *//24* }  
}

**public class** Demo02String {  
  
 @Test  
 **public void** testString01() {  
 String word =**"hello , welcome to bailiban! "**;  
 System.out.println(word.substring(8)); *//welcome to bailiban!* System.out.println(word.substring(8,15)); *//welcome* }  
  
 @Test  
 **public void** testString02() {  
 String word =**" hello "**;  
 *//去掉两端的空白字符* System.out.println(word.trim()); *//hello* }  
  
 @Test  
 **public void** testString03() {  
 String word =**"hello , welcome to bailiban! "**;  
 *//hello子字符串替换为byebye* System.out.println(word.replace(**"hello"**,**"byebye"**)); *//byebye , welcome to bailiban!* }  
  
 @Test  
 **public void** testString04() {  
 String word =**"hello , welcome to bailiban! "**;  
 *//返回索引位置为4的内容* System.out.println(word.charAt(4)); *//o* }  
  
 @Test  
 **public void** testString05() {  
 String word =**"welcome to bailiban! "**;  
 *//以空格字符将字符串分割为数组* String[] attr = word.split(**" "**);  
 System.out.println(Arrays.toString(attr)); *//[welcome, to, bailiban!]* }  
  
}

1. 测试StringBuilder类的相关API。

public class Demo03StringBuilder {  
  
 @Test  
 public void testStringBuilder01() {  
 StringBuilder sb = new StringBuilder();  
 System.out.println(sb);*// 空白* StringBuilder sb2 = new StringBuilder("bailiban");  
 System.out.println(sb2);*// bailiban* }  
  
 @Test  
 public void testStringBuilder02() {  
 StringBuilder sb = new StringBuilder();  
 sb = sb.append("hello ").append(" welcome ").append(" to ").append(" bailiban ");  
 System.out.println(sb);*// hello welcome to bailiban* }  
  
 @Test  
 public void testStringBuilder03() {  
 StringBuilder sb = new StringBuilder("bailiban");  
 System.out.println(sb.reverse());*// nabiliab* }  
}

1. 练习Integer类的常用API。

public class Demo04Integer {  
  
 @Test  
 public void testInteger01(){  
 *//将基本类型的1自动装箱为包装类Integer* Integer i = 1 ;  
 System.out.println(i);*//1* }  
  
 @Test  
 public void testInteger02(){  
 *//定义包装类对象i* Integer i = new Integer(1) ;  
 *//将包装类拆箱成基本类型int* int i2 = i ;  
 System.out.println(i2);*// 1* }  
  
 @Test  
 public void testInteger03(){  
 *// 将字符串类型转成对应的基本类型* int i = Integer.parseInt("99");  
 System.out.println(i);  
 }  
  
 @Test  
 public void testInteger04(){  
 Integer it = new Integer(100);  
 *// 将包装类转成基本类型* int i = it.intValue();  
 System.out.println(i);  
 }  
  
 @Test  
 public void testInteger05(){  
*// 获取36的2进制的表示形式* System.out.println( Integer.toBinaryString(36));  
*// 获取36的8进制的表示形式* System.out.println( Integer.toOctalString(36));  
*// 获取36的16进制的表示形式* System.out.println( Integer.toHexString(36));  
 }  
}

1. 将2个BigInteger对象进行加减乘除运算。

public class Demo05BigInteger {  
  
 @Test  
 public void testBigInteger01(){  
 BigInteger bi = new BigInteger("9999999999999999999999999999999999999");  
 System.out.println(bi);  
 }  
  
 @Test  
 public void testBigInteger02(){  
 BigInteger b1 = new BigInteger("99999999999999999999999999999999999999999");  
 BigInteger b2 = new BigInteger("99999999999999999999999999999999999999999");  
 *// 将2个BigInteger进行相加运算* System.out.println(b1.add(b2));  
 }  
  
 @Test  
 public void testBigInteger03(){  
 BigInteger b1 = new BigInteger("999999999999999999999999999999999999999999999");  
 BigInteger b2 = new BigInteger("111111111111111111111111111111111111111111111");  
 *// 将2个BigInteger进行相减运算* System.out.println(b1.subtract(b2));  
 }  
  
 @Test  
 public void testBigInteger04(){  
 BigInteger b1 = new BigInteger("999999999999999999999999999999999999999999999");  
 BigInteger b2 = new BigInteger("111111111111111111111111111111111111111111111");  
 *// 将2个BigInteger进行相乘运算* System.out.println(b1.multiply(b2));  
 }  
  
 @Test  
 public void testBigInteger05(){  
 BigInteger b1 = new BigInteger("999999999999999999999999999999999999999999999");  
 BigInteger b2 = new BigInteger("111111111111111111111111111111111111111111111");  
 *// 将2个BigInteger进行相乘运算* System.out.println(b1.divide(b2)); *// 9* }  
}

1. 创建2个BigDecimal对象，进行加减乘除运算。

public class Demo06BigDecimal {  
  
 @Test  
 public void testBigDecimal01(){  
 BigDecimal bd1 = new BigDecimal("0.123456789");  
 }  
  
 @Test  
 public void testBigDecimal02(){  
 BigDecimal bd1 = new BigDecimal("0.1");  
 BigDecimal bd2 = new BigDecimal("0.2");  
 *// 对2个小数进行相加运算* System.out.println(bd1.add(bd2));  
 }  
  
 @Test  
 public void testBigDecimal03(){  
 BigDecimal bd1 = new BigDecimal("0.1");  
 BigDecimal bd2 = new BigDecimal("0.2");  
 *// 对2个小数进行相减运算* System.out.println(bd2.subtract(bd1));  
 }  
  
 @Test  
 public void testBigDecimal04(){  
 BigDecimal bd1 = new BigDecimal("0.1");  
 BigDecimal bd2 = new BigDecimal("0.2");  
 *// 对2个小数进行相乘运算* System.out.println(bd2.multiply(bd1));  
 }  
  
 @Test  
 public void testBigDecimal05(){  
 BigDecimal bd1 = new BigDecimal("0.33");  
 BigDecimal bd2 = new BigDecimal("0.11");  
 *// 对2个小数进行相除运算，如果除不尽则报错* System.out.println(bd1.divide(bd2));  
 *// 对2个小数进行相除运算，如果除不尽则保留3位小数位,末尾采取四舍五入的方式* System.out.println(bd2.divide(bd1,3,BigDecimal.ROUND\_HALF\_UP));  
 }  
}

1. 测试DecimalFormat类的相关用法

public class Demo07DecimalFormat {  
  
 @Test  
 public void testDecimalFormat01(){  
 double pi = Math.PI;*//圆周率  
 //取一位整数* System.out.println(new DecimalFormat("0").format(pi));*//3  
 //取一位整数和两位小数* System.out.println(new DecimalFormat("0.00").format(pi));*//3.14  
 //取两位整数和三位小数，整数不足部分以0填补。* System.out.println(new DecimalFormat("00.000").format(pi));*// 03.142  
 //取所有整数部分* System.out.println(new DecimalFormat("#").format(pi));*//3  
 //以百分比方式计数，并取两位小数* System.out.println(new DecimalFormat("#.##%").format(pi));*//314.16%* long c =299792458;*//光速  
 //显示为科学计数法，并取五位小数* System.out.println(new DecimalFormat("#.#####E0").format(c));*//2.99792E8  
 //显示为两位整数的科学计数法，并取四位小数* System.out.println(new DecimalFormat("00.####E0").format(c));*//29.9792E7  
 //每三位以逗号进行分隔。* System.out.println(new DecimalFormat(",###").format(c));*//299,792,458  
 //将格式嵌入文本* System.out.println(new DecimalFormat("光速大小为每秒,###米。").format(c));  
 }  
}

1. 定义星期枚举，从周1到周日。通过switch遍历枚举，输出每一天的菜谱。

enum Weeks {  
 *//七个星期常量* MON,TUE,WES,THU,FRI,SAT,SUN  
}  
  
public class Demo09Enum02 {  
  
 @Test  
 public void testWeek(){  
 *//定义枚举变量并赋值* Weeks week = Weeks.SAT;  
 *//使用switch判断枚举变量* switch(week){  
 case MON:  
 System.out.println("星期一吃鱼香茄子");  
 break;  
 case TUE:  
 System.out.println("星期二吃蛋炒饭");  
 break;  
 case WES:  
 System.out.println("星期三吃小牛肉");  
 break;  
 case THU:  
 System.out.println("星期四吃热干面");  
 break;  
 case FRI:  
 System.out.println("星期五吃黄焖鸡");  
 break;  
 case SAT:  
 System.out.println("星期六吃泡面");  
 break;  
 case SUN:  
 System.out.println("星期天吃青椒肉丝");  
 break;  
 }  
 }  
  
}