常用类-day02

学习内容和目标

内容	知识点	掌握程度
StringBuffer和StringBuilder	可变字符串本质和常用方法	掌握
	可变字符串拓容原理	了解
	StringBuffer和StringBuilder实战场景和区别	掌握
Math类	Math类工具方法	了解
	[m,n]区间随机整数公式	掌握
Random类	Random常用方法	了解
Date类	日期时间概念	了解
	日期时间格式化和解析、SimpleDateFormat	掌握
	DateFormat	了解
Calendar类	日历概述和本质、日历常用字段	掌握
	日历常用方法	掌握
正则表达式	正则规则和语法	掌握
	正则练习	掌握

1.1 StringBuffer和StringBuilder类(掌握)

1.1.1 可变字符串概述(了解)

String类型提供了对字符串的只读操作，如果需要对包装的字符数组进行增、删、改、查时，就需要可变字符串。

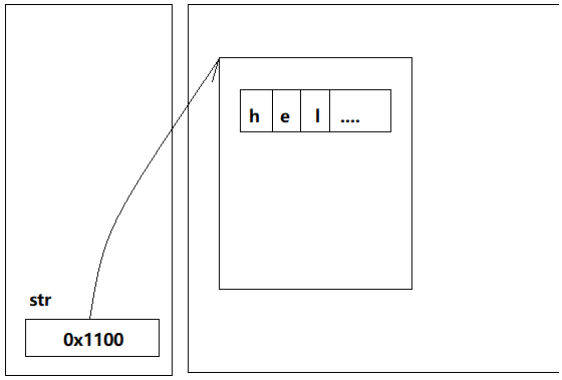
java中提供了两类可变字符串的类型StringBuffer、StringBuilder。

1.1.2 StringBuffer(掌握)

StringBuffer 也可以看成一个包装类，包装了一个字符数组，并提供了对该字符数组进行增、删、改、查的方法。所以，我们完全可以把StringBuffer看成一个"容器"!

StringBuffer封装的数组默认空间是16个字符，当超过默认空间后，可以自动拓容。

可变字符串内存图



StringBuffer常用方法

```java
public static void main(String[] args) {
    // 创建一个可变字符串容器，默认容量是16个字符
    StringBuffer sb = new StringBuffer();

    // 创建一个可变字符串容器，容量是100个字符
    // StringBuffer sb2 = new StringBuffer(100);

    // 【1】追加
    sb.append('A');
    sb.append('B');
    // 链式操作
    sb.append("C").append("D");
    System.out.println(sb.toString());

    // 【2】添加 insert
    sb.insert(0, 'E');
    sb.insert(0, "hello").insert(0, "world");
    System.out.println(sb);

    // 返回容器的容量
    System.out.println(sb.capacity());
    // 返回容器中字符的个数
    System.out.println(sb.length());

    // 当空间不足时，能否继续添加？=> 自动拓容
    sb.append(123);
    System.out.println(sb.length());
    System.out.println(sb.capacity());

    // 【2】删除
    // 删除指定位置index的字符
    //sb.deleteCharAt(10);

    // delete(start,end) 含头不含尾
    sb.delete(0, 10);
    System.out.println(sb);

    // 【3】修改
    sb.replace(0, 5, "eabcd");
    System.out.println(sb);

    sb.setCharAt(0, 'f');
```

```
    System.out.println(sb);
}
```

总结：

【1】 通过对可变字符串的API，你发现了什么?

【2】什么是链式操作?

> 授课技巧提示:
>
> 使用PPT演示增删改过程

1.1.3 拓容原理(了解+)

StringBuffer 内部维护了一个字符数组char[] value，默认空间16个长度，用户也可手动指定。
StringBuffer 在初始化时内部初始化了一个16个长度的字符数组。

查看StringBuffer源代码，总结拓容原理

拓容原理:
当添加元素时就是向字符数组添加元素，当容量不够时，利用value.length << 1 + 2后得到新容量newCapacity，然后利用数组复制的方式复制得到一个newCapacity的数组，然后把复制的新数组地址赋值给包装类StringBuffer 的value属性。

```
public static void main(String[] args) {
    // 拓容原理
    StringBuffer sb = new StringBuffer();
    sb.append("helloworld123456");
    System.out.println(sb.capacity());
    System.out.println(sb.length());

    // 自动拓容
    sb.append("7");
    System.out.println(sb.length());
    System.out.println(sb.capacity());

    // 继续添加17个
    sb.append("1234567890abcdefgh");
    System.out.println(sb.length());
    System.out.println(sb.capacity());
}
```

1.1.4 StringBuilder(掌握)

StringBuffer在源代码级别上已经做到了线程安全，所以StringBuffer非常适合多线程环境。
如果在单线程条件下使用可变字符串序列时，一定优先考虑StringBuilder。

1.1.5 面试题: StringBuffer和StringBuilder的区别(了解+)

相同点：都是字符串可变缓冲区，api提供了相同的增删改查操作。
不同点:
StringBuffer 线程安全，效率低；StringBuilder线程不安全，效率高。
StringBuffer jdk1.0;StringBuilder jdk1.5

1.2. Math（了解）

Math 类包含用于执行数学运算的方法，如初等指数、对数、平方根和三角函数等，该类的方法都是static修饰的，在开发中其实运用并不是很多，里面有一个求随机数的方法，偶尔会用到。

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

        // [1]求绝对值
        System.out.println(Math.abs(-10));
        System.out.println(Math.abs(10));

        // [2]求平方根和立方根
        System.out.println(Math.cbrt(27));
        System.out.println(Math.sqrt(9));

        // [3]ceil/floor
        float a = 9.2f;
        // 比9.2大的最小整数：向上取整
        System.out.println(Math.ceil(a));
        // 比9.2小的最大整数：向下取整
        System.out.println(Math.floor(a));

        // [4]求最值
        System.out.println(Math.max(10, 9));
        System.out.println(Math.min(10, 9));

        // [5] 指数
        System.out.println(Math.pow(3, 2));

        // [6]随机数 取值返回[0,1)
        System.out.println(Math.random());
        // 返回[m,n]区间的随机整数
        // (int)(Math.random()*(n-m+1)) + m;
}
```

Math也存在三角函数相关的工具方法，在实际开发了解即可。

```
// 角度和弧度准换(了解)
// 360 = 2PI
// 把弧度转化成角度
System.out.println(Math.toDegrees(Math.PI/4));
// 把角度转化成弧度
System.out.println(Math.toRadians(180));

// 三角函数（了解）
System.out.println(Math.sin(Math.PI/6));
System.out.println(Math.cos(Math.PI/6));
System.out.println(Math.tan(Math.PI / 4));
```

小练习：

```
// 求坐标两点（x1,y1）-(x2,y2)之间的距离
// Math.sqrt(Math.pow((x1-x2),2)+Math.pow((y1-y2),2))
```

1.3. Random（了解）

Random类用于生产一个伪随机数（通过相同的种子，产生的随机数是相同的），Math类的random方法底层使用的就是Random类的方式。

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

    // 1> Random
    Random r = new Random();
    // val的范围[0,100)之间
    int val = r.nextInt(100);
    System.out.println(val);

    // 2> 需求:随机产生4位的小写英文字母的验证码
    /**
      *   分析:
      *   [1]. 产生一个a-z的随机字符c => 'a' + [0,25]
      *   [2]. 把产生的随机字符存入数组？ 还是可变字符串?
      */

    StringBuilder code = new StringBuilder();
    char c;
    Random r2 = new Random();
    for(int i=0;i<4;i++){
        // [0,25]
        c = (char)('a' + r2.nextInt(26));
        code.append(c);
    }

    System.out.println("code = " + code);
}
```

1.4. UUID （了解）

UUID表示通用**唯一**标识符 (Universally Unique Identifier)，其算法通过电脑的网卡、当地时间、随机数等组合而成，优点是真实的唯一性，缺点是字符串太长了。

jdk1.5 才出现的类

```
public static void main(String[] args) {
    UUID uid = UUID.randomUUID();
    // uid = 678f9568-8967-4637-a48e-f0eae30faf43(36位)
    System.out.println("uid = " + uid);

    // 需求：生产一个商品的唯一序列码
    UUID uuid = UUID.randomUUID();
    String uuidStr = uuid.toString();

```

```
    // 注意java空字符的写法
    // uuidStr.replace('-','');
    // uuidStr = uuidStr.replace('-','\u0000');

    uuidStr = uuidStr.replace("-","");
    System.out.println("uuidStr = " + uuidStr);
}
```

1.5. 日期时间

1.5.1 日期时间概述

问题1：计算机如何表示时间?

因为日期时间是变化的，计算机不直接存储具体时间。

时间戳（timestamp）：具体时间（特定的瞬间）距离历元(1970年01月01日 00:00:00:000) 经过的毫秒数，用long类型存储。

计算机很容易存储long类型数据，所以计算机通过时间戳存储并表示时间。

问题2：为什么时间戳一样，时间却不同?

中国时间和国外时间的时间戳都一样，但时区不同。时区导致时间不同。

计算机以格林尼治所在地的标准时间作为时间统一协调时，这个时间在民间称为格林尼治时间（GMT），为了统一国际用法，也称世界协调时(UTC)

问题3：如何计算当地时间?

- 当地时间 = UTC + 时区偏移
- 中国位于东八区，时区偏移为(+8:00)
- 中国时间 = UTC + 8:00
- 日本时间 = UTC + 9:00

1.5.2 Date（掌握）

Date 位于java.util包中，表示特定的瞬间，内部包装了一个long类型的fastTime，通过运算可以把fastTime转换成年月日时分秒。

常用构造方法

```
public class Test01 {
    public static void main(String[] args) {

        Date date =  new Date();
        // Mon Nov 18 09:47:25 CST(china standard time) 2019
        System.out.println(date.toString());

        // 获取date对象的时间戳
        long ts = date.getTime();
```

```
        System.out.println(ts);

        // 构造一个对象表示未来时间或者过去时间?
        // 需求：构造日期时间对象表示明天的此时
        long ts2 = ts + 24 * 60 * 60 * 1000;
        Date date2 = new Date(ts2);
        System.out.println(date2.toString());
    }
}
```

日期时间比较大小

```
public class Test02 {
    public static void main(String[] args) {

        Date date = new Date();

        long ts = date.getTime() + 24 * 60 * 60 * 1000;
        Date date2 = new Date(ts);

        // [1]时间比较:比较内部封装的时间戳fastTime
        System.out.println(date.before(date2));
        System.out.println(date.after(date2));

        System.out.println(date.compareTo(date2));
        System.out.println(date.equals(date2));

    }
}
```

setTime可以修改Date中封装的时间戳

```
public class Test02 {
    public static void main(String[] args) {

        Date date = new Date();

        long ts = date.getTime() + 24 * 60 * 60 * 1000;
        Date date2 = new Date(ts);

        // getTime/setTime
        // 修改date2封装的时间戳
        date2.setTime(0);
        System.out.println(date2);
    }
}
```

15.2.1 SimpleDateFormat（掌握）

打印Date对象时，默认打印的是欧美人的日期时间风格，如果需要输出自定义的时间格式，比如2020年12月12日 12:12:12格式或者2020-12-12 12:12:12，此时可以使用SimpleDateFormat类。

SimpleDateFormat类，顾名思义是日期的格式化类，主要包括两个功能的方法：

- 格式化（format）：Date类型转换为String类型：String format(Date date)
- 解析（parse）：String类型转换为Date类型：Date parse(String source)

无论是格式化还是解析都需要设置日期时间的模式，所谓模式就是一种格式。

字母	日期或时间元素	表示	示例
G	Era 标志符	Text	AD
y	年	Year	1996; 96
M	年中的月份	Month	July; Jul; 07
w	年中的周数	Number	27
W	月份中的周数	Number	2
D	年中的天数	Number	189
d	月份中的天数	Number	10
F	月份中的星期	Number	2
E	星期中的天数	Text	Tuesday; Tue
a	Am/pm 标记	Text	PM
H	一天中的小时数（0-23）	Number	0
k	一天中的小时数（1-24）	Number	24
K	am/pm 中的小时数（0-11）	Number	0
h	am/pm 中的小时数（1-12）	Number	12
m	小时中的分钟数	Number	30
s	分钟中的秒数	Number	55
S	毫秒数	Number	978
z	时区	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	时区	RFC 822 time zone	-0800

日期模式举例：

```
yyyy-MM-dd                  如2020-12-12
HH:mm:ss                    如20：12：12
yyyy-MM-dd HH:mm:ss         如2020-12-12 20：12：12
yyyy/MM/dd HH:mm:ss         如2020/12/12 20：12：12
yyyy年MM月dd日 HH时mm分ss秒   如2020年12月12日 20时12分12秒
```

格式化和解析代码如下：

```java
public class SimpleDateFormatDemo {
    public static void main(String[] args) throws Exception {
        java.util.Date d = new java.util.Date();
        // 创建SimpleDateFormat对象，设置日期时间转换模式
        SimpleDateFormat sdf = new SimpleDateFormat();
        String pattern = "yyyy-MM-dd HH:mm:ss";
        sdf.applyPattern(pattern);
        // 格式化（format）：Date类型转换为String类型：String format(Date date)
        String str = sdf.format(d);
        System.out.println(str);//2018-05-17 14:48:38


        // 解析（parse）：String类型转换为Date类型：Date parse(String source)
        java.util.Date dd = sdf.parse(str);
        System.out.println(dd);//Thu May 17 14:48:38 CST 2018
    }
}
```

1.5.2.2 DateFormat(了解+)

DateFormat 是日期/时间格式化子类的抽象类，它以与语言无关（无需指定语言环境）的方式格式化并解析日期或时间。

DateFormat 提供了很多类方法，以获得基于默认或给定语言环境和多种格式化风格的默认日期/时间 Formatter。格式化风格包括 FULL、LONG、MEDIUM 和 SHORT。

获取日期格式化器

获取方式	描述	
DateFormat.getDateInstance();	获取默认风格、默认语言环境的格式化器	
DateFormat.getDateInstance(int style);	获取指定风格、默认语言环境的格式化器	
DateFormat.getDateInstance(int style, Locale aLocale);	获取指定风格,指定语言环境的格式化器	

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

        // 获取默认语言环境(与语言环境无关)，默认格式化风格的格式器
        // DateFormat df = DateFormat.getDateInstance();
        // 获取默认语言环境，指定格式化风格的格式器
        // DateFormat df = DateFormat.getDateInstance(DateFormat.FULL);
        // 获取指定语言环境，指定格式化风格的格式器
        DateFormat df = DateFormat.getDateInstance(DateFormat.FULL,
Locale.CHINA);

        /*
         * DateFormat.SHORT 19-11-18
         * DateFormat.MEDIUM 2019-11-18
         * DateFormat.LONG 2019年11月18日
         * DateFormat.FULL 2019年11月18日 星期一
         */

        Date date = new Date();
        String dateStr = df.format(date);
        System.out.println("dateStr = " + dateStr);
    }
```

获取时间格式化器

获取方式	描述
DateFormat.getDateInstance();	获取默认风格、默认语言环境的格式化器
DateFormat.getDateInstance(int style);	获取指定风格、默认语言环境的格式化器
DateFormat.getDateInstance(int style, Locale aLocale);	获取指定风格,指定语言环境的格式化器

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

        // 获取默认语言环境(与语言环境无关)，默认格式化风格的时间格式器
        // DateFormat df = DateFormat.getTimeInstance();

        // 获取默认语言环境，指定格式化风格的时间格式器
        // DateFormat df = DateFormat.getTimeInstance(DateFormat.FULL);
        // 获取指定语言环境，指定格式化风格的时间格式器
        DateFormat df = DateFormat.getTimeInstance(DateFormat.FULL,
Locale.CHINA);

        /*
         * DateFormat.SHORT 上午10:14
         * DateFormat.MEDIUM 10:14:36
         * DateFormat.LONG 上午10时14分52秒
         * DateFormat.FULL 上午10时15分07秒 CST
         */

        Date date = new Date();
        String dateStr = df.format(date);
        System.out.println("dateStr = " + dateStr);
    }
```

获取日期时间格式化器

获取方式	描述
DateFormat.getDateTimeInstance()	获取默认风格、默认语言环境的日期时间格式化器
DateFormat.getDateTimeInstance(int dateStyle, int timeStyle);	获取指定风格、默认语言环境的日期时间格式化器
DateFormat.getDateTimeInstance(int dateStyle, int timeStyle, Locale aLocale);	获取指定风格,指定语言环境的日期时间格式化器

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

        // 获取默认语言环境(与语言环境无关)，默认格式化风格的日期时间格式器
        // DateFormat df = DateFormat.getDateTimeInstance();

        // 获取默认语言环境，指定格式化风格的日期时间格式器
        // DateFormat df =
DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL);
```

```
        // 获取指定语言环境，指定格式化风格的日期时间格式器
        DateFormat df =
DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL,
Locale.CHINA);

        /*
         * DateFormat.SHORT ?
         * DateFormat.MEDIUM ?
         * DateFormat.LONG ?
         * DateFormat.FULL ?
         */

        Date date = new Date();
        String dateStr = df.format(date);
        System.out.println("dateStr = " + dateStr);
    }
```

1.6.Calendar（掌握+）

Calendar内部封装了一个long time 表示时间戳，其内部提供了方法通过对time计算出年月日时分秒...等日历字段，这些字段都被存储到一个数组中，通过get(字段)可以去数组中提取对于字段的值。

Calendar还提供了**用来对日期做相加减，重新设置日期时间功能。**

Calendar本身是一个抽象类，通过getInstance方法获取对象，其底层创建的是Calendar的子类对象。

```
public class CalendarDemo1 {
    public static void main(String[] args) throws Exception {
        // 根据当前地区，当前语言环境，当前时间构造一个通用的日历对象
        Calendar cal = Calendar.getInstance();
        /*
         *
java.util.GregorianCalendar[time=1583911881152,areFieldsSet=true,areAllFiel
dsSet=true,lenient=true,zone=sun.util.calendar.ZoneInfo[id="Asia/Shanghai",
offset=28800000,dstSavings=0,useDaylight=false,transitions=19,lastRule=null
],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1,YEAR=2020,MONTH=2,WEEK_OF
_YEAR=11,WEEK_OF_MONTH=2,DAY_OF_MONTH=11,DAY_OF_YEAR=71,DAY_OF_WEEK=4,DAY_O
F_WEEK_IN_MONTH=2,AM_PM=1,HOUR=3,HOUR_OF_DAY=15,MINUTE=31,SECOND=21,MILLISE
COND=152,ZONE_OFFSET=28800000,DST_OFFSET=0]
         */

        // System.out.println(cal);

        // 获取日历字段信息
        // 获取日历中的年月日
        System.out.println(cal.get(Calendar.YEAR));
        // 月从0开始，0表示1月，1表示2月...
        System.out.println(cal.get(Calendar.MONTH));
        System.out.println(cal.get(Calendar.DATE));
        // System.out.println(cal.get(Calendar.DAY_OF_MONTH));

        System.out.println(cal.get(Calendar.HOUR));
        System.out.println(cal.get(Calendar.MINUTE));
        System.out.println(cal.get(Calendar.SECOND));
```

```
        System.out.println(cal.get(Calendar.MILLISECOND));

        // 获取星期（周几）
        // 一周的第一天是周日
        System.out.println(cal.get(Calendar.DAY_OF_WEEK));

        // 获取一月中天数?
        System.out.println(cal.getActualMaximum(Calendar.DAY_OF_MONTH));
        System.out.println(cal.getActualMinimum(Calendar.DAY_OF_MONTH));

        // 获取一年的天数
        System.out.println(cal.getMaximum(Calendar.DAY_OF_YEAR));

        // 增加日历字段对于的值
        cal.add(Calendar.YEAR, 100);//在当前年份上增加100
        System.out.println(c.get(Calendar.YEAR));//2118
    }
}
```

需求1：查询某个时间最近一周的信息，如何表示最近这一周的开始时间和结束时间

假如给出时间为：2018-05-18 15:05:30，那么最近一周的开始和结束时间分别为：

开始时间：2018-05-12 00:00:00

结束时间：2018-05-18 23:59:59

```
public class CalendarDemo2 {
    public static void main(String[] args) throws Exception {
        String input = "2018-05-18 15:05:30";// 输入时间
        String pattern = "yyyy-MM-dd HH:mm:ss";
        SimpleDateFormat sdf = new SimpleDateFormat();
        sdf.applyPattern(pattern);
        Date d = sdf.parse(input);

        // -----------------------------------------
        Calendar c = Calendar.getInstance();
        c.setTime(d);// 把当前输入时间转换为Calendar对象

        c.set(Calendar.HOUR_OF_DAY, 23);
        c.set(Calendar.MINUTE, 59);
        c.set(Calendar.SECOND, 59);
        Date endDate = c.getTime();
        System.out.println(endDate.toLocaleString());

        c.add(Calendar.SECOND, 1);// 秒数增加1
        c.add(Calendar.DAY_OF_MONTH, -7);// 天数减去7
        Date beginDate = c.getTime();
        System.out.println(beginDate.toLocaleString());
    }
}
```

需求2：手写一个日历

```
日   一   二   三   四   五   六
                1   2   3
4   5   6   7   8   9   10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27* 28  29  30  31
```

```java
public static void main(String[] args) {
        /**
         * 日    一   二   三    四    五    六
         *                     1     2     3
         * 4     5    6    7    8     9     10
         * 11    12   13   14   15    16    17
         * 18    19   20   21   22    23    24
         * 25    26   27*  28   29    30    31
         */

        /**
         * 分析
         * [1] 获取本月第一天从星期几开始
         * [2] 获取本月总有几天，并从1-x开始循环输出日期
         * [3] 在循环过程中，需要考虑周六换行输出,并把今天用*标记
         */

    // 根据当前日期时间、当前语言环境获取日历对象
    Calendar cal = Calendar.getInstance();
    int today = cal.get(Calendar.DATE);
    System.out.println("今天:" + today);

    // 获取本月第一天星期几
    cal.set(Calendar.DATE, 1);
    int weekOfFirstDay = cal.get(Calendar.DAY_OF_WEEK);
    System.out.println("本月第一天星期: " + weekOfFirstDay);

    // 获取本月最小日期和最大日期
    int minDay = cal.getActualMinimum(Calendar.DAY_OF_MONTH);
    int maxDay = cal.getActualMaximum(Calendar.DAY_OF_MONTH);

    System.out.println("日\t一\t二\t三\t四\t五\t六");
    // 循环输出 空格
    for (int i = 1; i < weekOfFirstDay; i++) {
        System.out.print("\t");
    }
    for (int day = minDay; day <= maxDay; day++) {
        if(day == today){
            System.out.print(day + "*\t");
        }else{
            System.out.print(day + "\t");
        }

        cal.set(Calendar.DATE,day);
        if(cal.get(Calendar.DAY_OF_WEEK) == Calendar.SATURDAY){
            System.out.println();
        }
    }
```

```
49 }
```

1.7. 正则表达式

正则表达式，简写为regex和RE。

正则表达式用来判断某一个字符串是不是符合某一种规则，在开发中通常用于判断操作、替换操作、分割操作等。

19.1. 正则表达式规则

正则表达式匹配规则一：

No.	规范	描述	No.	规范	描述
1	\\	表示反斜线（\）字符	2	\t	表示制表符
3	\n	表示换行	4	[abc]	字符a、b或c
5	[^abc]	除了a、b、c之外的任意字符	6	[a-zA-Z0-9]	表示由字母、数字组成
7	\d	表示数字	8	\D	表示非数字
9	\w	表示字母、数字、下划线	10	\W	表示非字母、数字、下划线
11	\s	表示所有空白字符（换行、空格等）	12	\S	表示所有非空白字符
13	^	行的开头	14	$	行的结尾
15	.	匹配除换行符之外的任意字符			

正则表达式匹配规则二：

数量表示（X表示一组规范）

No.	规范	描述	No.	规范	描述
1	X	必须出现一次	2	X?	可以出现0次或1次
3	X*	可以出现0次、1次或多次	4	X+	可以出现1次或多次
5	X{n}	必须出现n次	6	X{n,}	必须出现n次以上
7	X{n,m}	必须出现n~m次			

逻辑运算符（X、Y表示一组规范）

No.	规范	描述	No.	规范	描述
1	XY	X规范后跟着Y规范	2	X|Y	X规范或Y规范
3	(X)	做为一个捕获组规范			

19.2. 正则表达式练习

判断一个字符串是否全部有数字组成

判断一个字符串是否是手机号码

判断一个字符串是否是18位身份证号码

判断一个字符串是否6到16位，且第一个字必须为字母

```
public class REDemo {
    public static void main(String[] args) throws Exception {
        // 判断一个字符串是否全部有数字组成
        System.out.println("12345678S".matches("\\d"));// false
        System.out.println("12345678".matches("\\d"));// false
        System.out.println("12345678".matches("\\d*"));// true
        System.out.println("1234".matches("\\d{5,10}"));// false
        System.out.println("12345678".matches("\\d{5,10}"));// true

        // 判断一个字符串是否是手机号码
        String regex1 = "^1[3|4|5|7|8][0-9]{9}$";
        System.out.println("12712345678".matches(regex1));// false
        System.out.println("13712345678".matches(regex1));// true

        // 判断一个字符串是否是18位身份证号码
        String regex2 = "\\d{17}[[0-9]x]";
        System.out.println("511123200110101234".matches(regex2));// true
        System.out.println("51112320011010123x".matches(regex2));// true
        System.out.println("51112320011010123S".matches(regex2));// false

        // 判断一个字符串是否6到16位，且第一个字必须为字母
        String regex3 = "^[a-zA-Z]\\w{5,15}$";
        System.out.println("will".matches(regex3));// false
        System.out.println("17will".matches(regex3));// false
        System.out.println("will17willwillwill".matches(regex3));// false
        System.out.println("will17".matches(regex3));// true
    }
}
```