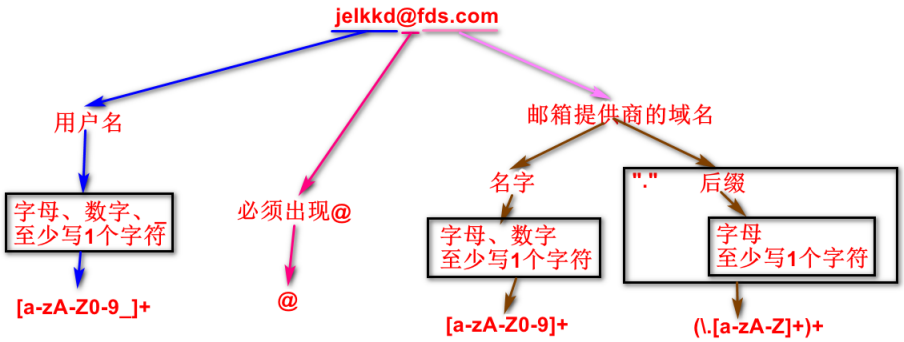
API基础第2天：

笔记：

1. 正则表达式：
    - 用于匹配一个字符串的内容是否符合格式要求
    - 正则表达式的语法：（了解即可）

```
1.[]:表示一个字符,该字符可以是[]中指定的内容
  例如:
  [abc]:该字符可以是a或b或c
  [^abc]:该字符只要不是a或b或c即可
  [a-z]:表示任意一个小写字母
  [A-Z]:表示任意一个大写字母
  [a-zA-Z]:表示任意一个字母
  [a-zA-Z0-9]:表示任意一个字母或数字
  [a-zA-Z0-9_]:表示任意一个字母或数字或下划线
2.预定义字符:
  .:表示任意一个字符,没有范围限制
  \d:表示任意一个数字,等同于[0-9]
  \D:表示不是数字
  \w:表示任意一个单词字符,等同于[a-zA-Z0-9_]（单词字符指字母/数字/_）
  \W:表示不是单词字符
  \s:表示任意一个空白字符
  \S:表示不是空白字符
3.量词:
  ?:表示前面的内容出现0-1次
    例如: [abc]? 可以匹配:a 或 b 或 c 或什么也不写
  +:表示前面的内容最少出现1次
    例如: [abc]+ 可以匹配:b或aaaaaaaaaa...或abcabcbabcbabcba....
    但是不能匹配:什么都不写 或 abcfdfsbbaqbb34bbwer...
  *:表示前面的内容出现任意次(0-多次)
    例如: [abc]* 可以匹配:b或aaaaaaaaaa...或abcabcba....或什么都不写
    但是不能匹配:abcfdfsbbaqbb34bbwer...
  {n}:表示前面的内容出现n次
    例如: [abc]{3} 可以匹配:aaa 或 bbb 或 aab 或abc 或bbc
    但是不能匹配: aaaa 或 aad
  {n,m}:表示前面的内容出现最少n次最多m次
    例如: [abc]{3,5} 可以匹配:aaa 或  abcab 或者 abcc
    但是不能匹配:aaaaaa 或 aabbd
  {n,}:表示前面的内容出现n次以上(含n次)
    例如: [abc]{3,} 可以匹配:aaa 或 aaaaa.... 或 abcbabbcbabcba....
    但是不能匹配:aa 或 abbdaw...
4.()用于分组,将括号内的内容看成一个整体
    例如: (abc){3} 表示abc整体出现3次. 可以匹配abcabcabc
    但是不能匹配aaa 或abcabc
    (abc|def){3}表示abc或def整体出现3次.
    可以匹配: abcabcabc 或 defdefdef 或 abcdefabc
    但是不能匹配abcdef 或abcdfbdef
```



邮箱正则表达式：**[a-zA-Z0-9_]+@[a-zA-Z0-9]+(\.[a-zA-Z]+)+**

2. String支持与正则表达式相关的方法：

- matches()：使用给定的正则表达式(regex)验证当前字符串的格式是否符合要求，符合则返回true，否则返回false

```
/**
 * boolean matches(String regex):匹配
 * 使用给定的正则表达式(regex)验证当前字符串的格式是否符合要求，符合则返回true，否则返回false
 */
public class MatchesDemo {
    public static void main(String[] args) {
        String email = "wangkj@tedu.cn";
        String regex = "\\w+@[a-zA-Z\\d]+(\\.[a-zA-Z]+)+";
        /*
        \\.：在Java中，\是转义的意思，若想让\表示反斜杠，则要转义\，即第一个\转义第二个\，
        让第二个\表示反斜杠。现在变成了\.，在正则表达式中，.表示任意一个字符，
        若想让.表示点，则要转义.，让.表示点
        \\w：匹配大小写字母或数字或下划线
        \\d：匹配数字
        +：一次或多次
         */

        boolean match = email.matches(regex);
        System.out.println(match);
    }
}
```

- replaceAll()：将当前字符串中满足正则表达式(regex)的部分替换为给定的字符串(s)

```
/**
 * String replaceAll(String regex,String s):替换
 * 将当前字符串中满足正则表达式(regex)的部分替换为给定的字符串(s)
 */
public class ReplaceAllDemo {
    public static void main(String[] args) {
        String line = "abc123def456ghi789";
        line = line.replaceAll("\\d+", "#NUMBER#");
        System.out.println(line);
    }
}
```

- split()：将当前字符串按照满足正则表达式的部分进行拆分，并将拆分出的以String[]形式来返回

```
import java.util.Arrays;

/**
 * String[] split(String regex):拆分
 * 将当前字符串中满足正则表达式的部分移除，剩下的部分自动分隔成各个元素，依次存进String[]数组中
 */
public class SplitDemo {
    public static void main(String[] args) {
        String line = "abc123def456ghi789";
        String[] data = line.split("\\d+"); //移除数字部分
        System.out.println(Arrays.toString(data)); //将data数组按String格式输出

        line = "123.456.78";
        data = line.split("\\.");
        System.out.println(Arrays.toString(data));

        //如果最开始就是可拆分项(.)，那么数组第1个元素为空字符串""
        //如果连续两个(两个以上)可拆分项，那么各个.中间也会拆出一个空字符串""
        //如果末尾连续多个可拆分项，那么拆出的空字符串会被忽略
        line = ".123.456...78......"; //头有一个，末尾没有，中间两点拆一个
        data = line.split("\\.");
        System.out.println(Arrays.toString(data)); //6个数组元素
    }
}
```

3. Object：对象
   - 是所有类的超类，所有类都直接或间接地继承了Object，万物皆对象，为了多态
   - Object中有几个经常被派生类重写的方法：toString()和equals()
     - 调用Object类的toString()默认返回：类的全称@地址，地址没有参考意义，所以通常重写toString()返回具体属性的值

       > 注意：String、StringBuilder都重写了toString()返回字符串内容

       ```
       import java.util.Objects;

       public class Point {
           private int x;
           private int y;

           // Alt + Insert 快捷生成方法

           public Point(int x, int y) {
               this.x = x;
               this.y = y;
           }

           public int getX() {
               return x;
           }
       ```

```
    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    @Override
    public String toString() {
        return "Point{" +
                "x=" + x +
                ", y=" + y +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Point point = (Point) o;
        return x == point.x && y == point.y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
}
```

```
public class ObjectDemo {
    public static void main(String[] args) {
        /*
        输出引用变量时默认调用Object类的toString()方法
        toString()返回：类的全称@地址
        但地址通常对开发没有帮助
        所以要重写toString()
        java中的String、StringBuilder就已经重写了
         */
        Point p = new Point(100, 200);
        System.out.println(p); //输出引用变量时默认调用Object类的
toString()方法
        System.out.println(p.toString()); //与上一句等价
    }
}
```

- 调用Object类的equals()默认用==进行比较（即比较的还是地址），没有参考意义，所以通常重写equals()来比较具体的属性值

> 注意：
>
> 1. String类重写equals()来比较字符串内容了，但StringBuilder并没重写equals()，所以StringBuilder依然还是调用Object的equals()来比较地址

> 2. 重写equals()的基本原则：
>
> - 原则上要比较两个对象的各个属性是否相同
> - 两个对象必须是同一类型，若类型不同则返回false

```
public class Point { /*代码同上文*/ }
```

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

        Point p1 = new Point(100,200);
        Point p2 = new Point(100,200);
        System.out.println(p1==p2); //false，==比较的是地址
        //因为重写Point类的重写equals()中比较的是属性的值是否相同
        System.out.println(p1.equals(p2)); //true

        String s1 = new String("hello");
        String s2 = new String("hello");
        //String重写equals()了
        System.out.println(s1.equals(s2)); //true，比较内容

        StringBuilder builder1 = new StringBuilder("hello");
        StringBuilder builder2 = new StringBuilder("hello");
        //StringBuilder没有重写equals()
        System.out.println(builder1.equals(builder2)); //false，比较
地址

        //s1与builder1的类型不同，所以equals()一定是false
        System.out.println(s1.equals(builder1)); //false
    }
}
```

4. 包装类：

- java定义了8个包装类，目的是为了解决基本类型不能直接参与面向对象开发的问题，使得基本类型可以通过包装类的形式存在。
- 包含：Integer、Character、Byte、Short、Long、Float、Double、Boolean。其中Character和Boolean是直接继承Object类的，而其余6个包装类继承自java.lang.Number类。
- JDK1.5推出了一个新的特性：自动拆装箱。当编译器编译时若发现是基本类型与包装类型相互赋值，将自动补充代码来完成转换工作，这个过程称为自动拆装箱。

```
public class IntegerDemo {
    public static void main(String[] args) {
        //演示包装类的常用操作：
        //1)可以通过包装类来得到基本类型的取值范围：
        int max = Integer.MAX_VALUE; //获取int的最大值
        int min = Integer.MIN_VALUE; //获取int的最小值
        System.out.println("int的最大值为:"+max+"，最小值为:"+min);
        long max1 = Long.MAX_VALUE; //获取long的最大值
        long min1 = Long.MIN_VALUE; //获取long的最小值
        System.out.println("long的最大值为:"+max1+"，最小值为:"+min1);

        //包装类型可以将字符串转换为对应的基本类型------必须熟练掌握
        String s1 = "38";
        int age = Integer.parseInt(s1); //将字符串s1转换为int类型
```

```
        System.out.println(age); //38------int

        String s2 = "123.456";
        double price = Double.parseDouble(s2); //将字符串s2转换为double类型
        System.out.println(price); //123.456------double

        /*
        //触发自动装箱特性，会被编译为: Integer i = Integer.valueOf(5);
        Integer i = 5; //基本类型到包装类型---装箱

        //触发自动拆箱特性，会被编译为: int j = i.intValue();
        int j = i; //包装类型到基本类型-------拆箱
        */

        /*
        Integer i1 = new Integer(5); //不常用
        Integer i2 = new Integer(5);
        System.out.println(i1==i2); //false,因为==是比较地址

        //Integer.valueOf()会复用-128到127范围内的数据---建议使用valueOf()方
式
        Integer i3 = Integer.valueOf(5);
        Integer i4 = Integer.valueOf(5);
        System.out.println(i3==i4); //true
         */
    }
}
```

补充:

1. 进制:

```
1)十进制:
  1.1)规则:逢10进1
  1.2)数字:0 1 2 3 4 5 6 7 8 9
  1.3)基数:10
  1.4)权:万 千 百 十 个
2)二进制:
  2.1)规则:逢2进1
  2.2)数字:0 1
  2.3)基数:2
  2.4)权:128 64 32 16 8 4 2 1
3)十六进制:
  3.1)规则:逢16进1
  3.2)数字:0 1 2 3 4 5 6 7 8 9 a b c d e f
  3.3)基数:16
  3.4)权:65536 4096 256 16 1
```

二进制转换为十进制的规则(正数)：所有为1的权相加

> 要求：今天必须熟练记住最后4个权(8 4 2 1)

```
权：      32  16  8  4  2  1
二进制：   1   1   0  1  0  1
十进制：   32+16+4+1----------------53

权：      32  16  8  4  2  1
二进制：   0   0   1  1  0  1
十进制：   8+4+1--------------------13

权：      32  16  8  4  2  1
二进制：   1   0   0  1  1  1
十进制：   32+4+2+1-----------------39
```

2. 十六进制的权：

```
16的0次幂---------------1
16的1次幂---------------16
16的2次幂---------------256
16的3次幂---------------4096
16的4次幂---------------65536
```

二进制的权：

```
2的0次幂----------------1
2的1次幂----------------2
2的2次幂----------------4
2的3次幂----------------8
2的4次幂----------------16
2的5次幂----------------32
```

十进制的权：

```
10的0次幂---------------1
10的1次幂---------------10
10的2次幂---------------100
10的3次幂---------------1000
10的4次幂---------------10000
```

3. 明日单词：

```
1)binary:二进制
```