API基础第1天：

回顾：

1. 内存管理：
   - 堆：new出来的对象(包括实例变量、数组元素)
   - 栈：正在调用的方法的局部变量(包括方法的参数)
   - 方法区：.class字节码文件(包括静态变量、所有方法)
2. 面向对象三大特征：
   - 封装：类、方法、访问控制修饰符
   - 继承：代码复用　超类：所有共有　接口：部分共有　派生类：特有

     单一继承、多接口实现，具有传递性
   - 多态：所有对象都是多态的(向上造型) 所有抽象方法都是多态的(重写)

     向上造型，强转，instanceof
3. String：java.lang包中，final的，不能被继承，底层封装了一个不可变char数组，Unicode，一个字符占2个字节

   字符串对象一旦创建好，对象内容永远无法改变，但字符串引用变量可以重新赋值(创建一个新的对象)

   ------不变对象
4. 字符串常量池：
   - 当使用字面量方式来创建对象时，会将这个字面量对象的引用地址缓存到字符串常量池中，当使用相同字符串再创建字符串对象时，将会复用常量池中的引用，以减少内存开销

精华笔记：

1. String：
2. String的常用方法：
   - length()：获取字符串的长度(字符个数)
   - trim()：去除当前字符串两边的空白字符
   - toUpperCase()/toLowerCase()：将当前字符串中的英文部分给转换为全大写/全小写
   - startsWith()/endsWith()：判断当前字符串是否是以给定的字符串开始的/结束的
   - charAt()：返回当前字符串指定位置上的字符----根据位置找字符
   - indexOf()/lastIndexOf()：检索给定字符串在当前字符串中第一次/最后一次出现的位置，根据字符串找位置
   - substring()：截取当前字符串中指定范围内的字符串(含头不含尾--包含start，但不包含end)
   - 静态方法valueOf()：将其它数据类型转换为String
3. StringBuilder：
   - 由于String是不变对象，每次修改内容都会创建新的对象，因此String不适合频繁修改操作，为了解决这个问题，java提供了StringBuilder类。
   - StringBuilder是专门用于修改字符串的一个类，内部维护一个可变的char数组，所做操作都是在这个数组之上进行的，修改速度、性能非常优秀，并且提供了修改字符串的常见方式：增、删、改、插

4. StringBuilder的常用方法：
    - append()：增加内容----------增
    - delete()：删除部分内容----------删
    - replace()：替换部分内容----------改
    - insert()：插入内容----------插

笔记：

1. String：

```
/*
          常见面试题：
            String s = new String("hello");
            问:如上语句创建了几个对象?
            答:2个
              第一个:字面量"hello"
              ----java会创建一个String对象表示字面量"hello"，并将其存入常量池
              第二个:new String()
              ----new String()时会再创建一个字符串对象，并引用hello字符串的内容
*/
String s = new String("hello"); //创建2个对象
String s1 = "hello"; //复用常量池中的字面量对象
System.out.println("s:"+s);   //s:hello
System.out.println("s1:"+s1); //s1:hello
System.out.println(s==s1); //false，==比较的是地址是否相同

//在实际应用中，String比较相等一般都是比较字符串内容是否相等
//因此我们需要使用equals()方法来比较两个字符串的内容是否相同
System.out.println(s.equals(s1)); //true，equals()比较的是内容是否相同

/*
        String s1 = "123abc"; //堆中有一个123abc字面量对象，同时缓存到常量池中
        //编译器在编译时，若发现是两个字面量相连，则会直接连接好并将结果保存起来
        //如下语句相当于: String s2 = "123abc";
        String s2 = "123"+"abc"; //直接复用常量池中的对象
        System.out.println(s1==s2); //true

        String s3 = "123";
        //因为s3是一个变量，所以在编译期并不会直接连接好
        String s4 = s3+"abc"; //创建一个新的对象存储123abc
        System.out.println(s4==s1); //false
*/
```

2. String的常用方法：
    - length()：获取字符串的长度(字符个数)

```
public class LengthDemo {
    public static void main(String[] args) {
        String str = "我爱Java!";
        int len = str.length(); //获取str的长度
        System.out.println(len); //7
    }
}
```

- trim()：去除当前字符串两边的空白字符

```
public class TrimDemo {
    public static void main(String[] args) {
        String str = "     hello world     ";
        System.out.println(str); //     hello world
        str = str.trim(); //去除str两边的空白字符，并将去除之后的新的对象存储到
str中
        System.out.println(str); //hello world
    }
}
```

- toUpperCase()/toLowerCase()：将当前字符串中的英文部分给转换为全大写/全小写

```
public class ToUpperCaseDemo {
    public static void main(String[] args) {
        String str = "我爱Java!";
        String upper = str.toUpperCase(); //将str中英文部分转换为全大写，存到
upper中
        System.out.println(upper); //我爱JAVA!

        String lower = str.toLowerCase(); //将str中英文部分转换为全小写，存到
lower中
        System.out.println(lower); //我爱java!
    }
}
```

- startsWith()/endsWith()：判断当前字符串是否是以给定的字符串开始的/结束的

```
public class StartsWithDemo {
    public static void main(String[] args) {
        String str = "thinking in java"; //java编程思想(经典书)
        boolean starts = str.startsWith("think"); //判断str是否是以think开
头的
        System.out.println(starts); //true

        boolean ends = str.endsWith(".png"); //判断str是否是以.png结尾的
        System.out.println(ends); //false
    }
}
```

- charAt()：返回当前字符串指定位置上的字符----根据位置找字符

```
public class CharAtDemo {
    public static void main(String[] args) {
        //                      111111---和下面的连起来10/11/12/13/14/15
        //            0123456789012345
        String str = "thinking in java";
        char c = str.charAt(9); //获取str中下标9所对应的字符
        System.out.println(c); //i
    }
}
```

- indexOf()/lastIndexOf()：检索给定字符串在当前字符串中第一次/最后一次出现的位置，根据字符串找位置

```
public class IndexOfDemo {
    public static void main(String[] args) {
        //                      111111
        //            0123456789012345
        String str = "thinking in java";
        int index = str.indexOf("in"); //检索in在str中第1次出现的位置
        System.out.println(index); //2
        //从下标为3的位置开始找in第1次出现的位置
        index = str.indexOf("in",3);
        System.out.println(index); //5
        index = str.indexOf("abc"); //若字符串在str中不存在，则返回-1
        System.out.println(index); //-1

        index = str.lastIndexOf("in"); //找in最后一次出现的位置
        System.out.println(index); //9
    }
}
```

- substring()：截取当前字符串中指定范围内的字符串(含头不含尾--包含start，但不包含end)

```
public class SubstringDemo {
    public static void main(String[] args) {
        //                      1
        //            01234567890
        String str = "www.tedu.cn";
        String name = str.substring(4,8);
        System.out.println(name); //tedu

        name = str.substring(4); //从下标4开始一直到末尾
        System.out.println(name); //tedu.cn
    }
}
```

- 静态方法valueOf()：将其它数据类型转换为String

```
public class ValueOfDemo {
    public static void main(String[] args) {
        int a = 123;
        String s1 = String.valueOf(a); //将int型变量a转换为String类型并赋值
给s1
```

```
        System.out.println(s1); //123---字符串类型

        double b = 123.456;
        String s2 = String.valueOf(b); //将double型变量b转换为String类型并赋
值给s2
        System.out.println(s2); //123.456---字符串类型

        String s3 = b+""; //任何类型与字符串相连，结果都变为字符串类型，效率低(一
会讲)
        System.out.println(s3); //123.456---字符串类型
    }
}
```

3. StringBuilder：

- 由于String是不变对象，每次修改内容都会创建新的对象，因此String不适合频繁修改操作，为了解决这个问题，java提供了StringBuilder类。
- StringBuilder是专门用于修改字符串的一个类，内部维护一个可变的char数组，所做操作都是在这个数组之上进行的，修改速度、性能非常优秀，并且提供了修改字符串的常见方式：增、删、改、插

```
public class StringStringBuilderDemo {
    public static void main(String[] args) {
        StringBuilder s = new StringBuilder("a");
        for(int i=0;i<10000000;i++) { //1000万次
            s.append(i);
        }
        System.out.println("执行完毕");

        /*
        //String不适合频繁修改内容(效率低)
        String s = "a";
        for(int i=0;i<10000000;i++){ //1000万次
            s = s+i; //每次修改都会在内存中分配新的对象
        }
        System.out.println("执行完毕");
         */
    }
}
```

4. StringBuilder的常用方法：

- append()：增加内容----------增
- delete()：删除部分内容----------删
- replace()：替换部分内容----------改
- insert()：插入内容----------插

```
//StringBuilder的演示
public class StringBuilderDemo {
    public static void main(String[] args) {
        String str = "好好学习Java";
        //复制str的内容到builder中--------好好学习Java
        StringBuilder builder = new StringBuilder(str);
```

```
        //append():追加内容-----在末尾追加
        builder.append(", 为了找个好工作");
        System.out.println(builder); //好好学习Java，为了找个好工作

        //replace():替换部分内容(含头不含尾)
        //将下标为9到15的内容替换为---就是为了改变世界
        builder.replace(9,16,"就是为了改变世界");
        System.out.println(builder); //好好学习Java，就是为了改变世界

        //delete():删除部分内容(含头不含尾)
        builder.delete(0,8); //删除下标为0到7的
        System.out.println(builder); //，就是为了改变世界

        //insert():插入内容
        builder.insert(0,"活着"); //在下标为0的位置插入活着
        System.out.println(builder); //活着，就是为了改变世界



        /*
        //StringBuilder的创建方式:
        StringBuilder builder1 = new StringBuilder(); //空字符串
        StringBuilder builder2 = new StringBuilder("abc"); //abc串
        String str = "abc";
        StringBuilder builder3 = new StringBuilder(str); //abc串

        String str2 = builder3.toString(); //将builder3转换为String类型
        */
    }
}
```

补充:

1. 字符串内容若做查看操作，那建议用String----------实际应用中一般都是查看

   字符串内容若需要做频繁修改操作，那建议用StringBuilder

2. StringBuilder和StringBuffer的区别：-----现在先不要纠结，我先写着，学完线程之后才能理解
   - StringBuilder：非线程安全的，并发处理的，性能稍快
   - StringBuffer：线程安全的，同步处理的，性能稍慢
3. getter/setter

```
public class Point { //很多框架都是基于getter/setter来取值、赋值的----一种习惯
    private int x;
    private int y;

    public int getX(){ //getter获取值
        return this.x;
    }
    public void setX(int x){ //setter设置值
        this.x = x;
```

```
    }

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

//getter/setter的演示
public class GetterSetterDemo {
    public static void main(String[] args) {
        Point p = new Point();
        p.setX(100);
        p.setY(200);
        System.out.println(p.getX()+","+p.getY());

    }
}
```

4. 文档注释：
    - 是功能性的注释，描述类的功能是什么、方法是干什么的、常量的功能是什么
    - 一般写类、方法、常量的上面

```
/**
 * 用于演示文档注释
 * @author WKJ
 */
public class DocDemo {
    /**
     * 表示窗口的宽
     */
    public static final int WIDTH = 641;

    /**
     * 计算两个整数的和
     * @param num1 第1个加数
     * @param num2 第2个加数
     * @return 返回num1与num2的和
     */
    public static int plus(int num1,int num2){
        return num1+num2;
    }
}
```

5. 明日单词：

```
1)regex:正则
2)match:匹配
3)mail:邮件
4)split:分隔
5)all:所有
6)object:对象
7)point:点
8)line:行
9)integer:整型
10)parse:分析、解析
```

验证码一般都是不区分大小写的

验证码是AxYt----------统一转为小写----------axyt

输入的是aXYt----------统一转为小写----------axyt

注册----用户名： doudou ----------接收字符串的时候，常规都是trim()

登录----用户名：doudou