API第一天：

回顾：

1. 访问控制修饰符：----------封装，保护数据的安全
    - public：公开的，任何类
    - private：私有的，本类
    - protected：受保护的、本类、派生类、同包类
    - 默认的：什么也不写，本类、同包类
2. final：最终的、不能改变的
    - 变量不能被改变、方法不能被重写、类不能被继承
3. static：静态的
    - 静态变量：static，类，方法区，一份，类名点来访问。所有对象所共享的数据
    - 静态块：static，类，类被加载时自动执行，一次。初始化/加载静态资源/变量
    - 静态方法：static，类，方法区，一份，类名点来访问，没有this，不能直接访问实例成员。方法的操作与对象无关
4. static final常量：
    - 必须声明同时初始化，类名点来访问，不能改变，大写
    - 编译器在编译时会将常量直接替换为具体的数，效率高
    - 在程序运行过程中数据永远不变，并且经常使用
5. 枚举：
    - 引用数据类型，对象数目是固定的，用于装一组常量

精华笔记：

1. String字符串类型：
    - java.lang.String类使用final修饰，不能被继承
    - String的底层封装的是一个字符数组
    - String在内存中采用Unicode编码格式，每个字符占用2个字节的内存空间
    - 字符串对象一旦创建，对象内容永远无法改变，但字符串引用可以重新赋值(指向新的对象)
        - String称为不变对象
2. 字符串常量池：是java对String的一个优化措施
    - java推荐我们使用字面量/直接量(直接"")的方式来创建字符串对象，并且会将所有以字面量方式创建的对象缓存到常量池中，当使用相同字面量再创建字符串对象时将会复用常量池中的对象，以减少内存开销。
3. String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定符串开始/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的字符串(含头不含尾)
- 静态方法valueOf()：将其它数据类型转换为String

4. StringBuilder类：
   - 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类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、性能优秀，并且提供了修改字符串的常见方式：增、删、改、插。
5. StringBuilder的常用方法：
   - append()：追加内容----------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字符串类型：
   - java.lang.String类使用final修饰，不能被继承
   - String的底层封装的是一个字符数组
   - String在内存中采用Unicode编码格式，每个字符占用2个字节的内存空间
   - 字符串对象一旦创建，对象内容永远无法改变，但字符串引用可以重新赋值(指向新的对象)
     - String称为不变对象
2. 字符串常量池：是java对String的一个优化措施
   - java推荐我们使用字面量/直接量(直接"")的方式来创建字符串对象，并且会将所有以字面量方式创建的对象缓存到常量池中，当使用相同字面量再创建字符串对象时将会复用常量池中的对象，以减少内存开销。

```
public class StringDemo {
    public static void main(String[] args) {
        /*
          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"，并将其存入常量池中
              第二个:new String()
              ----new String()时会再创建一个字符串对象，并引用hello字符串内容
         */
        String s = new String("hello"); //创建了2个对象
        String s1 = "hello"; //复用常量池中的字面量对象地址
        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的内容是否相同
        //因此我们需要使用equals()方法来比较两个字符串内容是否相同
```

```
        System.out.println(s.equals(s1)); //true，equals()比较的是内容是否相同---必须掌握

        /*
          使用字面量("")来创建字符串对象时，JVM会检查常量池中是否有该对象
          1)若没有，则会创建字符串对象，并将其引用存入到常量池中
          2)若有，则直接将常量池中的对象(引用)返回，并不会创建新的字符串对象
         */
        /*
        String s1 = "123abc"; //常量池还没有，因此创建该字符串对象，并存入常量池中
        String s2 = "123abc"; //常量池中已经有了，直接复用了
        String s3 = "123abc"; //常量池中已经有了，直接复用了
        //引用类型==，比较的是地址是否相同-----这是规定
        System.out.println(s1==s2); //true
        System.out.println(s1==s3); //true
        System.out.println(s2==s3); //true

        s1 = s1+"!"; //创建新的字符串对象("123abc!")，并将地址赋值给s1
        System.out.println(s1==s2); //false
        */

        /*
        String s1 = "123abc"; //堆中创建123abc字符量对象，并缓存到常量池中
        //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
        //如下语句会被编译为: String s2 = "123abc";
        String s2 = "123"+"abc";
        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编译好
        String s4 = s3+"abc"; //创建一个新的对象存储123abc
        System.out.println(s1==s4); //false
        */
    }
}
```

3. String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        ";
         str =str.trim(); //去除str两边的空白字符(一个新的字符串对象)，并存入
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中的英文部分转换为全大写，并
赋值给upper
        System.out.println(upper); //我爱JAVA!
        String lower = str.toLowerCase(); //将str中的英文部分转换为全小写，并
赋值为lower
        System.out.println(lower); //我爱java!
    }
}
```

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

```
public class StartsWithDemo {
    public static void main(String[] args){
        String str = "thinking in java"; //java编程思想(java经典书---工具
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----和下面的连成10/11/12/13/14/15
        //            0123456789012345
        String str = "thinking in java";
        char c = str.charAt(8); //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----和下面的连成10/11/12/13/14/15
        //            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
        index = str.indexOf("in",3); //从下标为3的位置开始找in第1次出现的位置
        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"); //检索in在str中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的字符串(含头不含尾)

```
public class SubstringDemo {
    public static void main(String[] args) {
        //                      1----与下面的数字连成10
        //            01234567890
        String str = "www.tedu.cn";
        int start = str.indexOf(".")+1; //4
        int end = str.indexOf(".",start); //8
        String name = str.substring(start,end); //截取下标4到7的字符串
        System.out.println(name); //tedu

    }
}
```

- 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，结果都会变为字符串类型，效率低
        System.out.println(s3); //123.456---字符串类型
    }
}
```

4. StringBuilder类：

- 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类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、性能优秀，并且提供了修改字符串的常见方式：增、删、改、插。

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

```
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)
        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");
        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和StringBuilder互转:
        String str = "abc";
        StringBuilder builder3 = new StringBuilder(str); //abc串
        String str2 = builder3.toString();
         */
    }
}
```

补充：

1. API：应用程序接口，讲java中给大家提供好的非常常用的类、接口、方法.....

2. java.lang包：语言包，java将特别特别常用的类封装到lang包中了，它认为你写程序过程中一定会用到这些类，所以java.lang包中的类是不需要import的。
3. 字符串内容若需要查看，则建议用String----------实际应用中一般都是查看

   字符串内容若需要频繁修改，则建议StringBuilder
4. 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
```