JAVASE-DAY01/DAY02 晚课

1. DAY01

1.1 String 常用的 API

- length（） 方法用于返回字符串的长度，也就是字符串中字符的个数
- indexOf（） 方法用于查找指定字符或者子字符串在字符串中第一次出现的位置，如果找到，则返回该字符串或者子字符串的索引，如果未找到，则返回 -1
- chatAt（）方法用于获取字符串中指定位置的字符，它接受一个整数参数，表示要获取字符的索引，返回该位置的字符
- subString（）方法用于提取字符串中的子串，它接受一个或者两个整数参数，表示子串的起始位置和结束位置，包头不包尾
- toUpperCase（）方法用于将字符串中的所有字符转换为大写形式
- trim（）方法用于去除字符串两端的空白字符（空格，制表符，换行符等）返回一个新的字符串参数，该字符串是原字符串去除空白字符后的结果
- startsWith（）方法用于检查字符串是否以指定的前缀开始，它接受一个字符串参数，如果原字符串以该前缀开始，则返回 true，否则，返回 false
- valueOf（）方法用于将其他类型的值转换为字符串表示，它接受一个值作为参数，并返回表示该值的字符串。

1.2 案例：URL 格式验证器

- **案例描述：**
  - 编写一个程序，用于验证用户输入的字符串是否符合 URL 的格式要求。程序将检查字符串的长度、是否以特定前缀开头、是否包含特定字符，并输出验证结果给用户。
- **URL 格式要求：**

  http://baidu.COM
  - 要求 URL 的长度必须在 10 到 100 个字符之间 //true
    - length()方法检查字符串的长度
  - 两端不允许有空格 //去除空格
    - trim() 方法去除字符串两端的空格

- 并且是以 http:// 或者 https:// 开头，以.com 结束 //true
  - startsWith()方法检查是否是以 http://或者 https://开头
  - endsWith()方法检查字符串是否以 .com 结尾
- 不允许包含特殊的字符（&，#，？）// true
  - 使用 indexOf()方法检查是否包含特殊字符
- 要求全部都由小写字母组成 //xiaoxie

- **代码实现：**

```
Java
package cn.tedu.demo01;

import java.util.Scanner;

public class URLValidator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        //用户输入字符串
        System.out.println("请输入字符串");
        String next = scanner.next();

        //验证字符串 URL
        boolean b = validateURL(next);

        //输出验证结果
        if(b){
            System.out.println("URL 验证通过");
        }else{
            System.out.println("URL 验证失败");
        }
    }

    public static boolean validateURL(String input){
        // 1. 要求 URL 的长度必须在 10 到 100 个字符之间 || 或
        // 检查字符串长度
        if(input.length()<10 || input.length()>100){
            return false;
        }
```

```
        //2. 两端不允许有空格
        input = input.trim();

        //3. 并且是以 http:// 或者 https:// 开头，以.com 结束 && 与
        if(!input.startsWith("http://") ||
!input.startsWith("https://") && !input.endsWith(".com")){
            return false;
        }

        //3.不允许包含特殊的字符（&，#，？）
        if(input.indexOf("&")!=-1 || input.indexOf("#")!=-1 ||
input.indexOf("?")!=-1){
            return false;
        }
        //4. 全部都是以小写字母组成
        input = input.toLowerCase();

        return true;
    }
}
```

2. DAY02

2.1 StringBuffer 和 StringBuilder 的区别（需要背）

`StringBuffer` 和 `StringBuilder` 是 Java 中用于处理可变字符串的类，它们之间的主要区别在于它们的线程安全性和性能特性。

1. 线程安全性
   - `StringBuffer` 是线程安全的，它的方法是同步的，可以在多线程环境下安全地使用。这意味着多个线程可以同时访问和修改同一个 `StringBuffer` 对象，而不会导致数据不一致的问题。
   - `StringBuilder` 不是线程安全的，它的方法是非同步的。在多线程环境下使用 `StringBuilder` 需要额外的同步措施，否则可能导致数据不一致的问题。
2. 性能特性
   - `StringBuffer` 的方法是同步的，这使得它在执行线程安全的操作时会产生一定的性能开销。因为它的方法会进行同步处理，确保在多线程环境下的数据一致性，但在单线程环境下可能会比较耗费性能。

- `StringBuilder` 的方法是非同步的，它没有线程安全的保证，因此在单线程环境下执行操作时比 `StringBuffer` 更快速和高效。

3. 注意：

a. 除了线程安全性和性能特性之外，`StringBuffer` 和 `StringBuilder` 的其他方法和用法是相同的，它们都提供了用于修改和操作字符串的一系列方法。

b. 如果代码在多线程环境下被访问或者修改，或者需要线程安全的操作，那么应该使用 StringBuffer

c. 如果代码在单线程环境下执行，或者不需要线程安全的操作，并且对性能有较高的要求，那么应该使用 StringBuilder

2.2 包装类

- 每一个八大基本数据类型都有对应的包装类
- 在 JDK1.5 之后提出了自动拆装箱的概念，用于基本数据类型和包装类之间的相互转换的过程。
- 拆箱：
  - 拆箱是指将包装类对象转换为对应的基本数据类型，同样，Java 也提供了自动拆箱的功能，也就在需要使用基本数据类型的地方，可以直接使用包装类对象，而 Java 会自动将其拆箱为对应的基本数据类型
  - 例如： 将 Integer 对象拆箱为基本数据类型 int

```
Java
Integer integerObj = 20;
int num = integerObj; // 自动拆箱
```

- 装箱：
  - 装箱是指将基本数据类型转换为对应的包装类对象。Java 提供了自动装箱的功能，即在需要使用包装类对象的地方，可以直接使用基本数据类型，而 Java 会自动将其转换为对应的包装类对象。
  - 例如，将基本数据类型 `int` 装箱为 `Integer` 对象：

```
Java
int num = 10;
Integer integerObj = num; // 自动装箱
```

2.3 Object

- equals 方法
  - 在 Object 类中默认的 equlas 方法是通过 == 来进行比较的，默认比较的是对象的地址，如果想要比较对象的内容，则需要在自定义类中重写 equals 方法
- toString 方法
  - 在输出对象的时候，如果不重写 toString 方法，则默认输出的是当前对象的地址，如果需要输出当前对象中属性的值，则需要重写 toString 方法。

2.4 正则表达式

正则表达式能够描述字符串的格式，通常用于验证字符串内容

- **一个字符**

[ ] 用于描述单一字符，方括号内部可以定义这个字符的内容，也可以描述一个范围

正则表达式	说明
[abc]	a、b、c中任意一个字符
[^abc]	除了a、b、c的任意字符
[a-z]	a、b、c、……、z中的任意一个字符
[a-zA-Z0-9]	a~z、A~Z、0~9中任意一个字符
[a-z&&[^bc]]	a~z中除了b和c以外的任意一个字符，其中&&表示“与”的关系

- **预定义字符**

" . " 点 在正则表达式中表示任意一个字符除了换行符

" \ " 在正则表达式中是转义字符，当我们需要描述一个已经被正则表达式使用的特殊字符时，我们就可以使用 " \ " 将其转变为原本的意思

正则表达式	说明
.	任意一个字符
\d	任意一个数字字符，相当于[0-9]
\w	单词字符,相当于 [a-zA-Z0-9_]
\s	空白字符，相当于[\t\n\x0B\f\r]
\D	非数字字符
\W	非单词字符
\S	非空白字符

- **数量词**

通常我们需要描述的字符串会出现很多重复出现的元素，但是又不需要严格的限制出现的次数时，我们就可以使用 " * "，“ + ”这些量词。

  - “ + ”：表示内容可以连续出现至少 1 次以上
  - “ * ”：表示内容出现 0~若干次
  - “ ？ ”：表示内容出现 0~1 次
  - { n }：表示内容必须出现 n 次
  - { n, }：表示内容出现至少 n 次
  - { n,m }：表示内容出现 n-m 次

正则表达式	说明
X?	表示0个或1个X
X*	表示0个或任意多个X
X+	表示1个到任意多个X（大于等于1个X）
X{n}	表示n个X
X{ n , }	表示n个到任意多个X（大于等于n个X）
X{ n , m }	表示n个到 m 个 X

2.5 案例：文本处理器

- **案例描述：**
  - 编写一个程序，用于处理用户输入的文本。程序将验证文本是否符合特定的格式要求，并进行相应的处理和转换。
- **验证规则：**
  - 验证文本是否包含字母和数字
    - 如果包含字母和数字，则将字符串倒序输出
    - 如果不包含字母和数字，则将元音字母替换为星号，并输出替换后的字符串
    - 最终将文本拆分为单词，并输出每个单词
  - 步骤
    - 先接收用户所输入的数据 String input
    - 验证是否包含字母和数字 String 验证正则的方法 mathchs(" [a-zA-Z0-9]*")
      - true ---倒序输出 StringBuilder--reverse()
      - false---将数字（本身是元音字母，为了简单需求，先写为数字）替换为星号 replace() 不支持正则的使用，replaceAll()支持正则的使用
        - input.replaceAll("\\d","*");
        -
    - 将文本拆分为单独的字符
      - String[] words=input.split("\\s");
- **代码实现：**

```
Java
package cn.tedu.demo01;



import java.util.Scanner;

public class TextProcessor {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        //用户输入文本
        System.out.println("请输入文本");
```

```
        String input = scanner.next();

        //验证文本格式 如果包含字母和数字，则将字符串倒序输出
        if(input.matches("[0-9A-Za-z]*")){
            //使用 StringBuffer 构建反转字符串
            StringBuilder reversed = new StringBuilder(input);
            reversed.reverse();
            System.out.println("反转字符串："+reversed.toString());
        }else{
            //如果不包含字母和数字，则将数字替换为星号，并输出替换后的
字符串
           input=input.replaceAll("\\d","*");
            System.out.println("替换后的字符串：" + input);
        }
        // 使用 split() 方法将文本拆分为单词
        String[] words = input.split("\\s+");
        System.out.println("单词列表：");
        for (String word : words) {
            System.out.println(word);
        }
    }
}
```