[1 开发工具使用 3](#_Toc40686782)

[1.1 Eclipse 3](#_Toc40686783)

[1.2 配置服务Tomcat 3](#_Toc40686784)

[1.2.1 选择Apache 3](#_Toc40686785)

[1.2.2 选择tomcat路径 3](#_Toc40686786)

[1.2.3 选择默认JDK(注意不是JRE) 4](#_Toc40686787)

[1.2.1 点击add按钮添加新的JDK 5](#_Toc40686788)

[1.2.2 Server选择m默认的JDK 8](#_Toc40686789)

[1.3 Eclipse启动tomcat 8](#_Toc40686790)

[1.3.1 添加项目 9](#_Toc40686791)

[1.3.2 配置启动服务 9](#_Toc40686792)

[1.3.3 Eclipse清理项目缓存 10](#_Toc40686793)

[1.3.4 Eclipse清理缓存 10](#_Toc40686794)

[1.4 Eclipse相关设置 11](#_Toc40686795)

[1.4.1 设置字体大小 11](#_Toc40686796)

[1.4.2 配置Maven 14](#_Toc40686797)

[1.4.3 配置SVN 15](#_Toc40686798)

[2 基础知识应用 15](#_Toc40686799)

[2.1 驼峰命名法 15](#_Toc40686800)

[2.1.1 大驼峰 15](#_Toc40686801)

[2.1.2 小驼峰 15](#_Toc40686802)

[2.2 文档注释 15](#_Toc40686803)

[2.2.1 单行注释 16](#_Toc40686804)

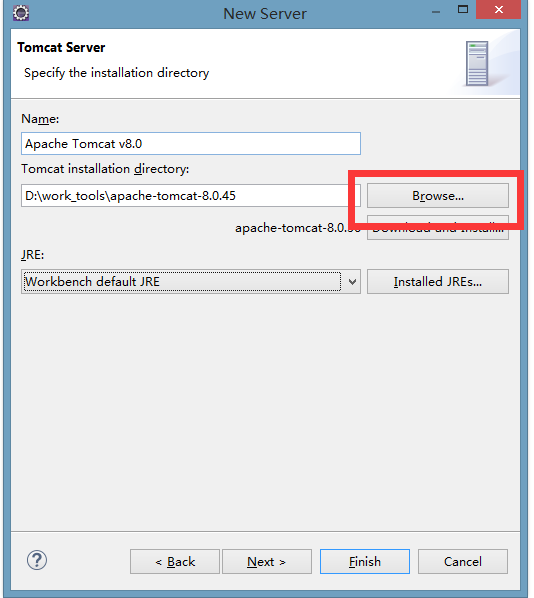
[2.2.2 多行注释 16](#_Toc40686805)

[2.2.3 文档结构注释 16](#_Toc40686806)

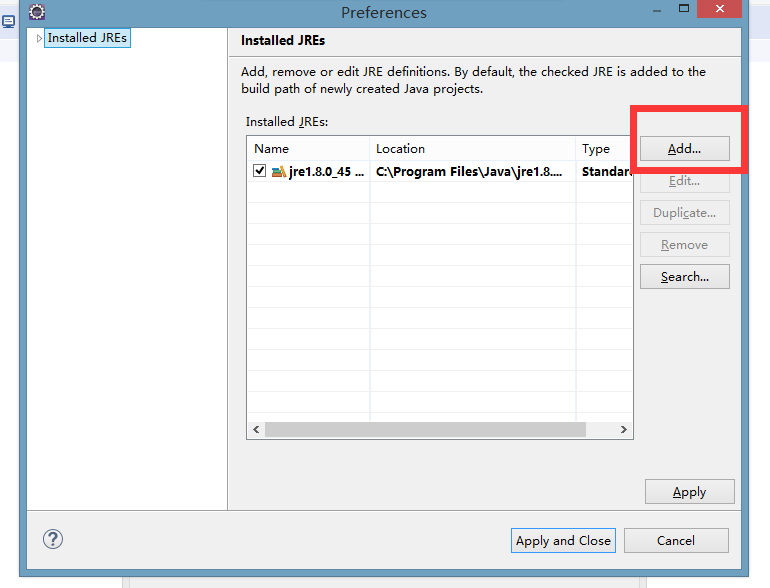
[2.3 Debug模式调试 16](#_Toc40686807)

[2.3.1 进入debug模式，操作 17](#_Toc40686808)

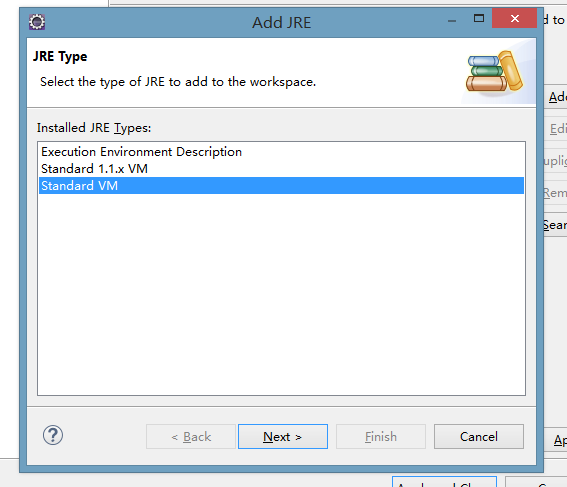
1. 开发工具使用
   1. Eclipse
   2. 配置服务Tomcat
      1. 选择Apache
      2. 选择tomcat路径



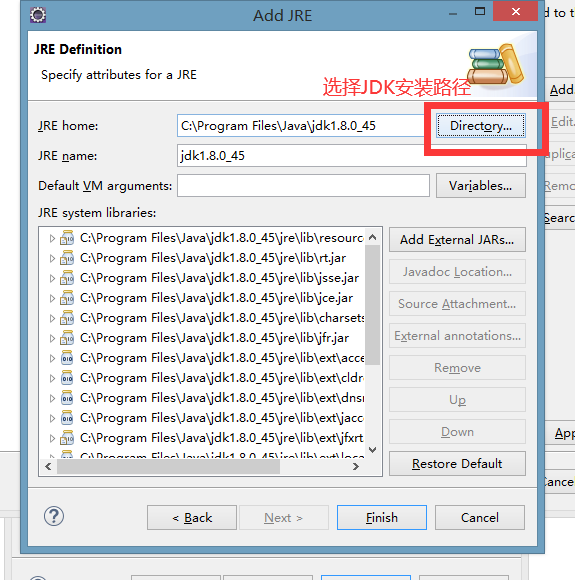
* + 1. 选择默认JDK(注意不是JRE)



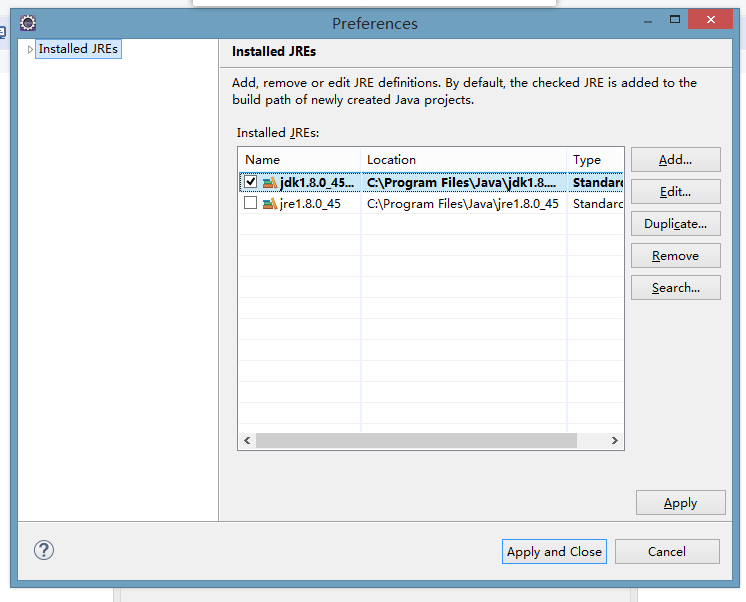
* + 1. 点击add按钮添加新的JDK



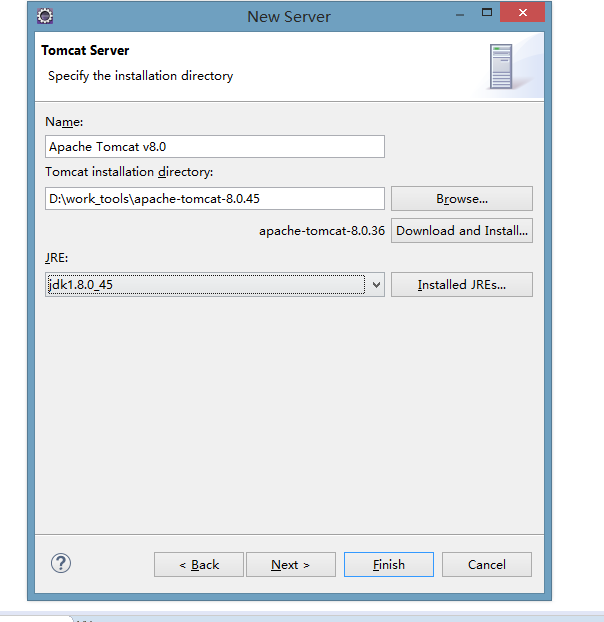
* + - 1. 添加JDK路径



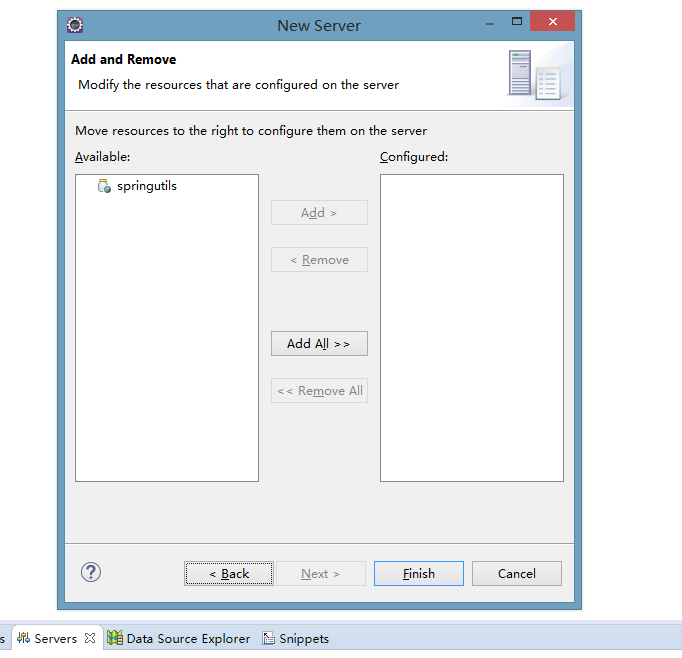
* + - 1. 选择新添加的JDK



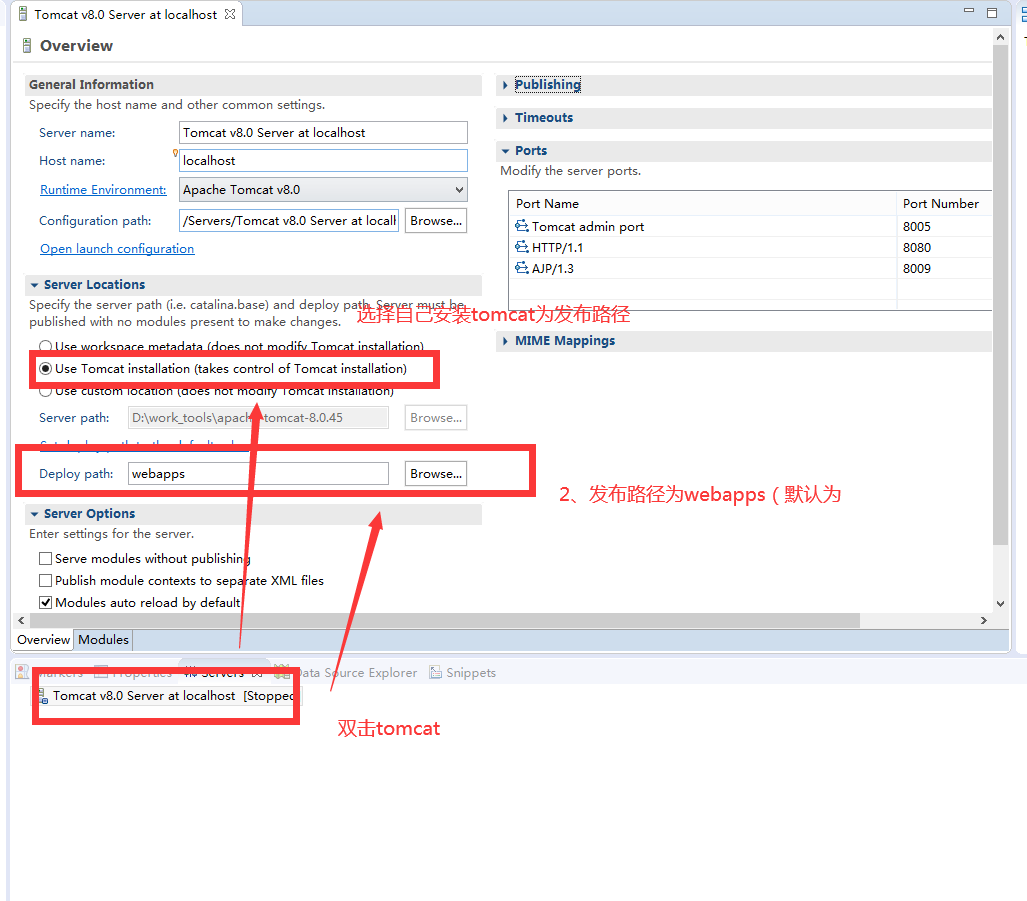
* + 1. Server选择m默认的JDK



* 1. Eclipse启动tomcat
     1. 添加项目



* + 1. 配置启动服务

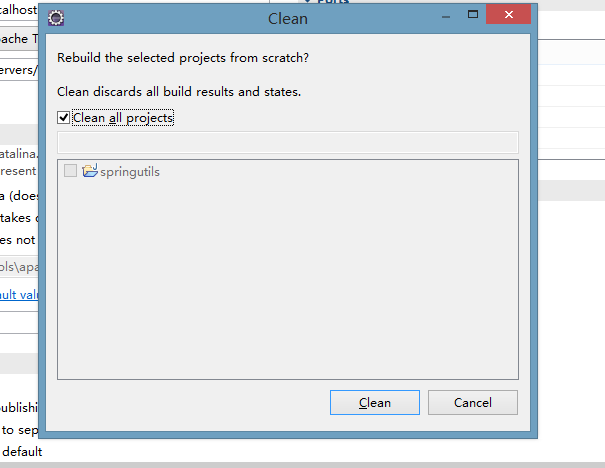


* + 1. Eclipse清理项目缓存

在tomcat 8.v severt at localhost上邮件，选择clean来清理该项目产生的缓存

* + 1. Eclipse清理缓存

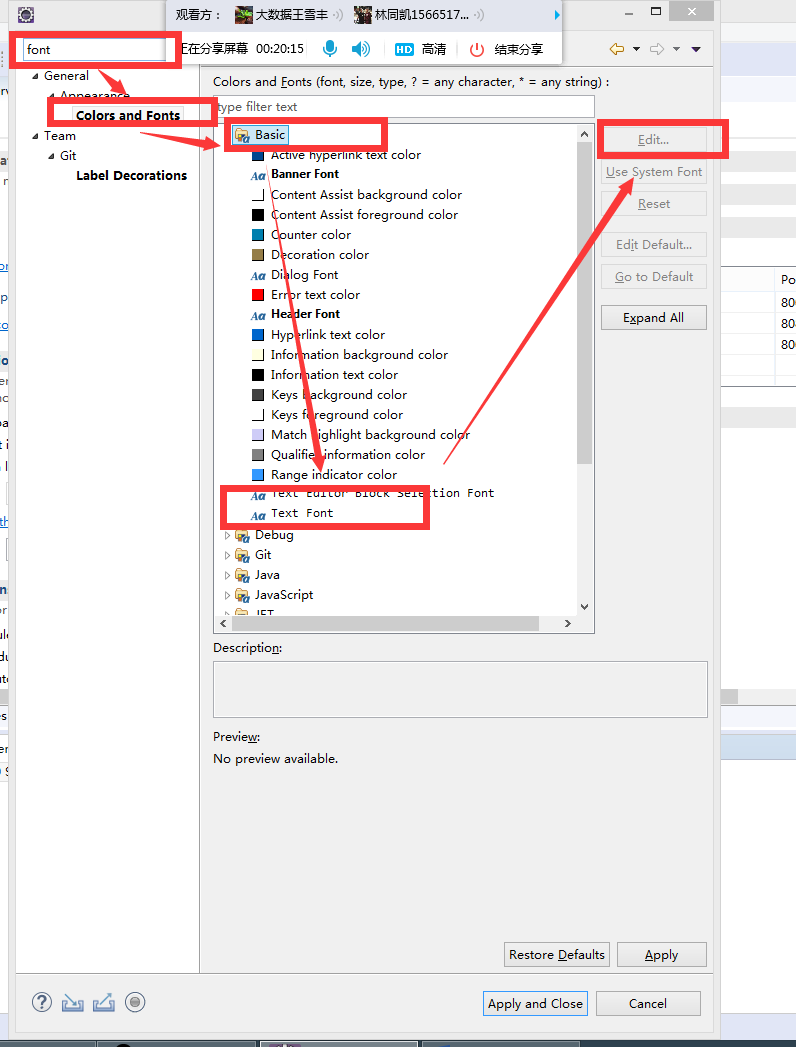
选择菜单栏中project选项，选择clean,选择要清理的项目缓存文件进行清理。



* 1. Eclipse相关设置
     1. 设置字体大小

选择windows选择perfance,在弹出框中输入font

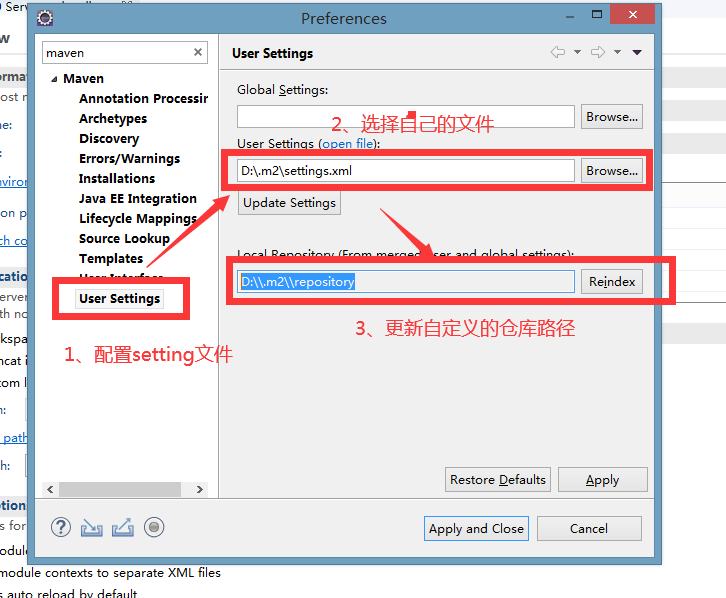
* + - 1. 设置选项卡：



* + - 1. 字体设置

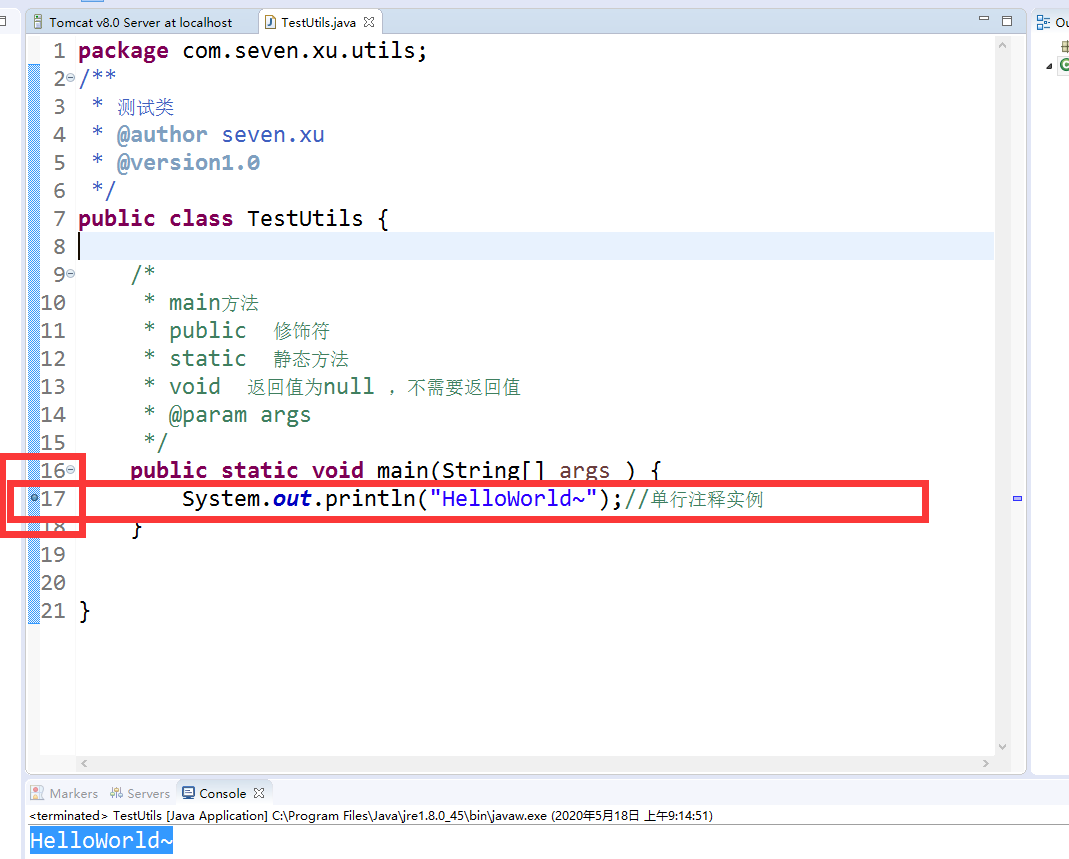


* + 1. 配置Maven

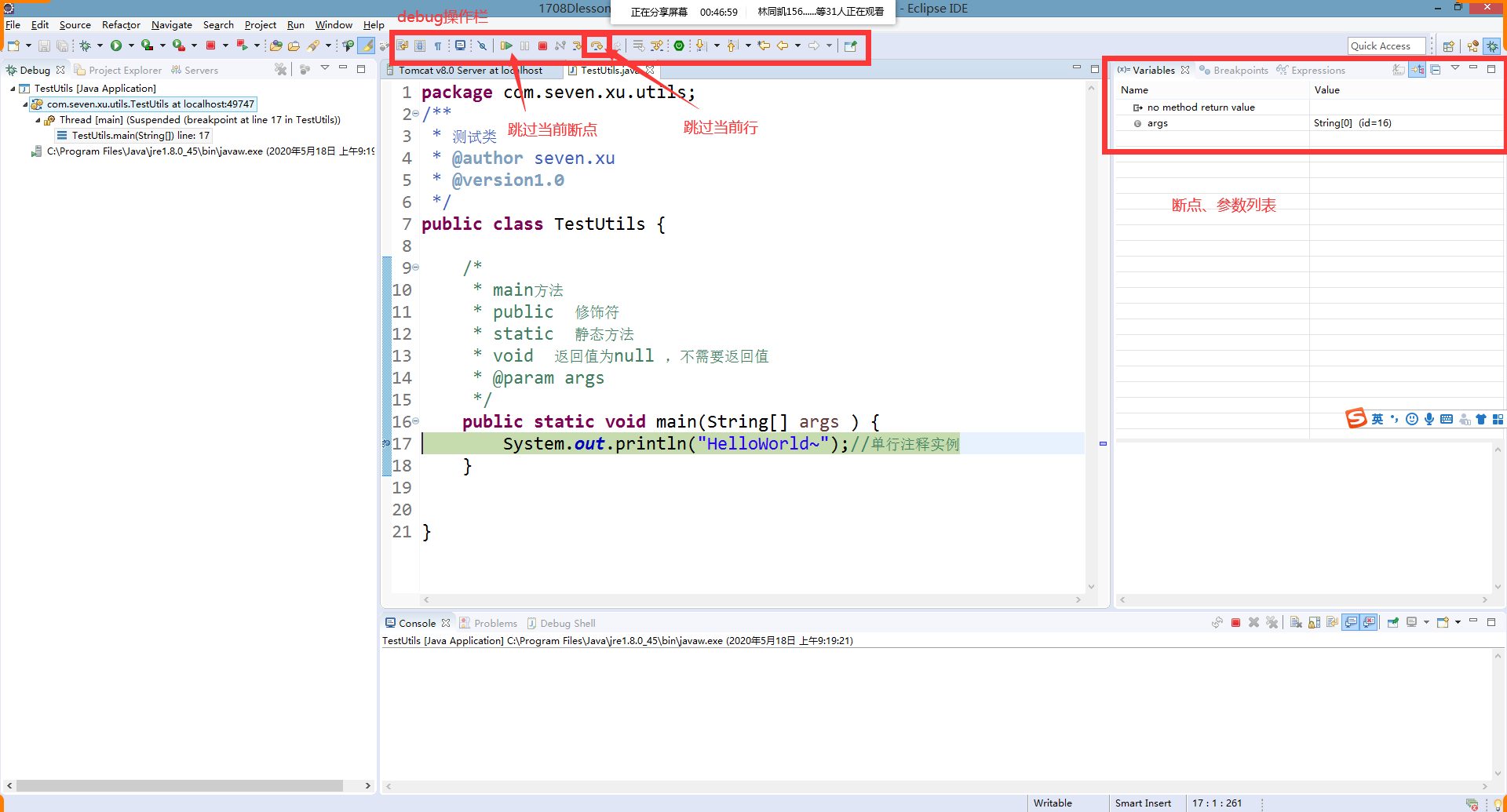


* + 1. 配置SVN
       1. Eclipse集成SVN插件
       2. Eclipse配置仓库位置
       3. Eclipse操作SVN
          1. 拉去代码
          2. 更新代码
          3. 提交代码
          4. 解决冲突
  1. Debug模式调试

在要测试行的最前方双击，打断点，本main方法中右键，选择Debug AS采用debug模式启动。



* + 1. 进入debug模式，操作



1. 基础知识应用
   1. 驼峰命名法
      1. 大驼峰

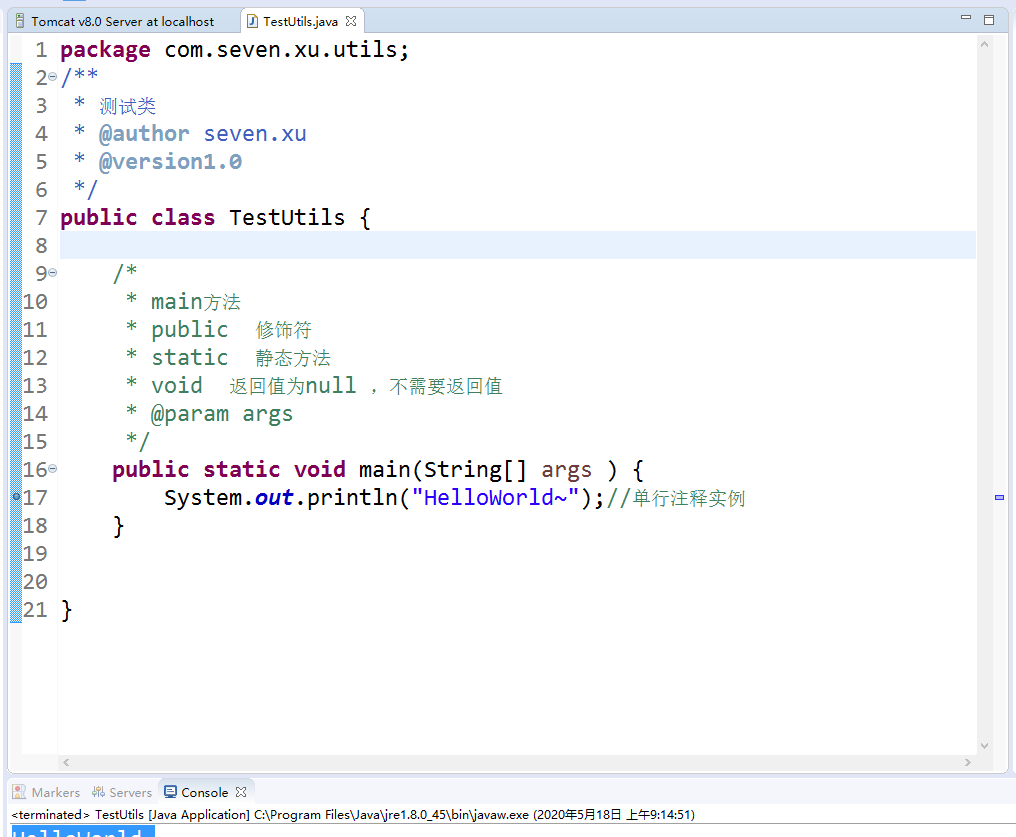
所有单词的首字母都大写一般用来标识类名称。

* + 1. 小驼峰

所有单词除了首个单词的首字母小写外，其余的首字母都大写一般用来标识方法名称、变量名称。

* 1. 文档注释

三种注释方式如下：



* + 1. 单行注释

一般用在方法中，单独注释一行

* + 1. 多行注释

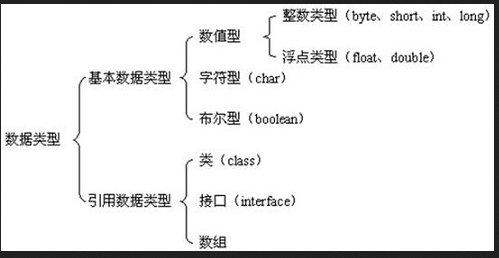
一般用在方法上，给方法添加注释信息用

* + 1. 文档结构注释

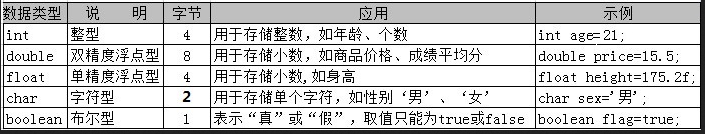
一般用在类上，用来给类添加相关注释信息

* 1. JAVA数据类型

Java 语言是一种强类型语言。通俗点说，在 Java 中存储的数据都是有类型的，而且必须在编译时就确定其类型。 Java 中有两类数据类型：在 Java 的领域里，基本数据类型变量存的是数据本身，而引用类型变量存的是保存数据的空间地址。说白了，基本数据类型变量里存储的是直接放在抽屉里的东西，而引用数据类型变量里存储的是这个抽屉的钥匙，钥匙和抽屉一一对应。



* + 1. 常用的基本数据类型以及字节长度：



* + 1. 数值大小限制(超过则报错-1)：

byte 127(2^7-1) -128(-2^7)

short 32767(2^15-1) -32768(-2^15)

int 2147483647(2^31-1) -2147483648(-2^31)

long 9223372036854775807(2^63-1) -9223372036854775808(2^63)

默认值0L long a=100000L;

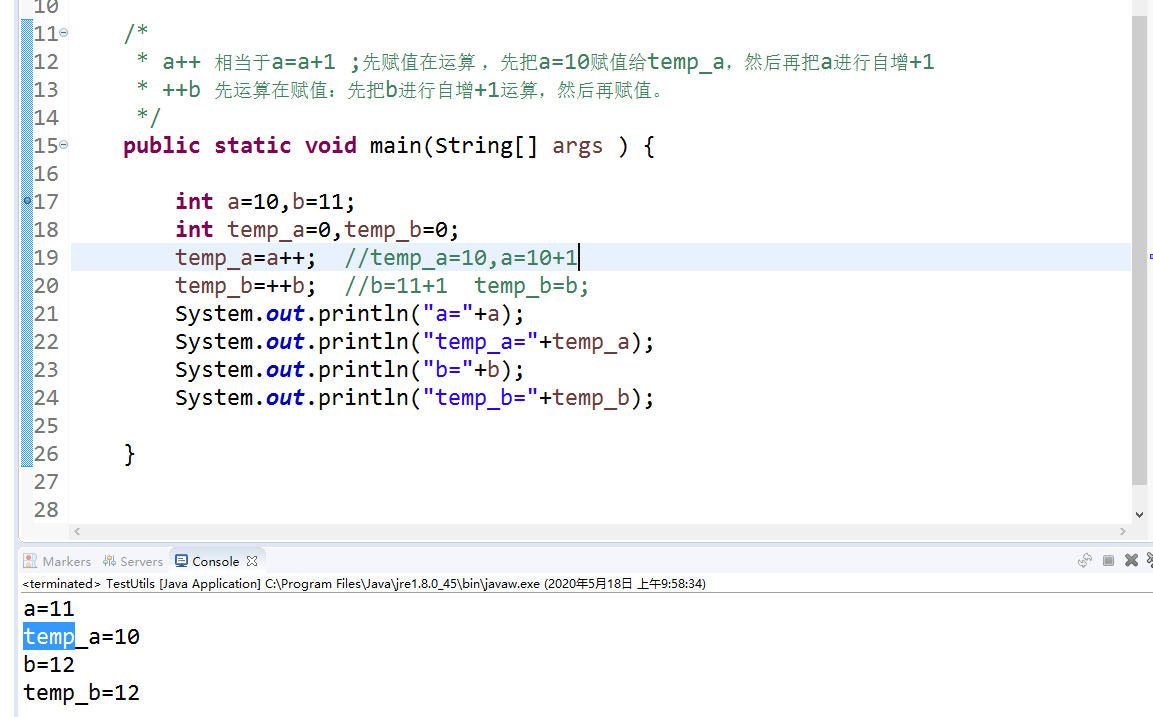
float 默认值是0.0f float f1=253.4f;

double 默认值是0.0d double d1=123.4;

boolean 默认值是false;

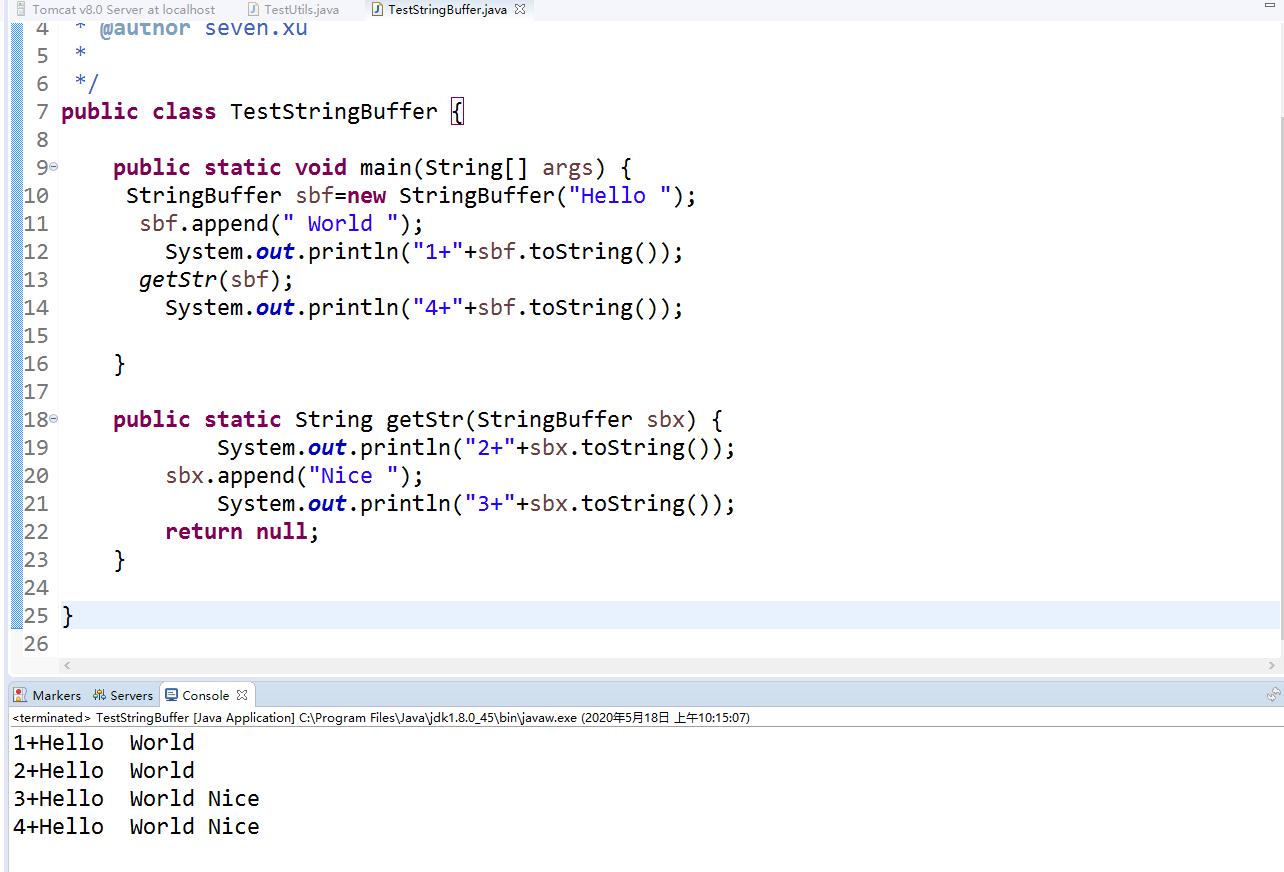
* + 1. 常用的运算符：重点

常用的运算符a++和++b的测试实例



* + 1. 引用传递、值传递：重点

引用类型方法调用传递时，传递的对象在内存中的地址



* + - 1. 引用传递在内存模型中的传递方式解析图。
         1. 栈:先进后出

###### 压栈操作

方法调用时，在内存栈中的操作

###### 出栈操作

方法调用结束时，在内存栈中的操作

* + - * 1. 堆：所有通过New关键字创建的对象都在堆中

