

## 1 目录

1	相关说明.....	1
2	安装与优化.....	2
2.1	安装 JDK.....	2
2.2	安装 Eclipse.....	4
2.3	Eclipse 优化.....	6
2.3.1	优化开发布局结构.....	6
2.3.2	修改字体样式.....	8
2.3.3	添加代码自动补全功能.....	9
2.3.4	统一编码.....	10
2.3.5	运行自动保存.....	11
2.4	导入工程.....	11
2.5	导入工程配置错误修正.....	13
2.6	删除工程.....	15
2.7	Eclipse 常用快捷键.....	16
3	第一个 Java 程序.....	17
4	封装设计实例.....	21
5	流程控制循环与数组实例.....	28

# 1 相关说明

软件	版本文件	系统类型	下载地址
JDK	jdk-8u161-windows-x64.exe	64bit	<a href="#">下载页面</a>
Eclipse	eclipse-jee-oxygen-R-win32-x86_64.zip	64bit	<a href="#">下载页面</a>

说明: **2.3Eclipse** 为学生课后调试程序提供帮助, 实验室开发环境已经包含优化的全部内容。  
实验室桌面文件夹: 系统程序设计, 包含开发所需全部内容。

百度网盘共享地址, 包含 **JDK8.161** 及 **Eclipse**, 文档, ppt 等内容  
链接:<https://pan.baidu.com/s/1skAM0FV> 密码:w4se

课程基于以下软件及版本, 可从官方网站下载, 或从以上网盘下:

JDK 8.161

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Eclipse for Java EE, Oxygen 4.7

<https://eclipse.org/downloads/eclipse-packages/>

Online tutorial

<https://docs.oracle.com/javase/tutorial/index.html>

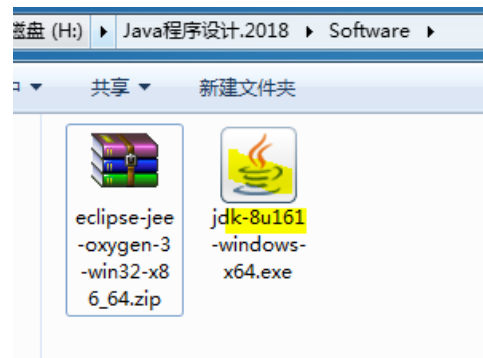
## 2 安装与优化

### 2.1 安装 JDK

安装 JDK 至默认路径，如系统包含 Java 8 版本 JDK 则无需安装。

**安装后，无需在系统环境中配置 Path 等变量**

运行 JDK 安装程序，启动反应较慢



建议使用默认安装路径，也可修改

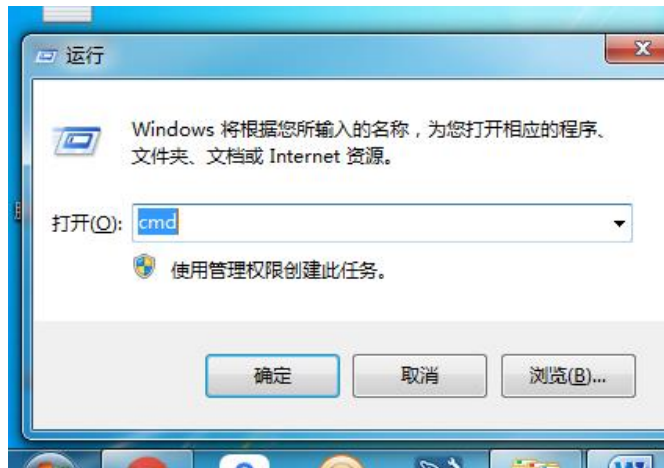


继续安装 JRE，建议使用默认安装路径

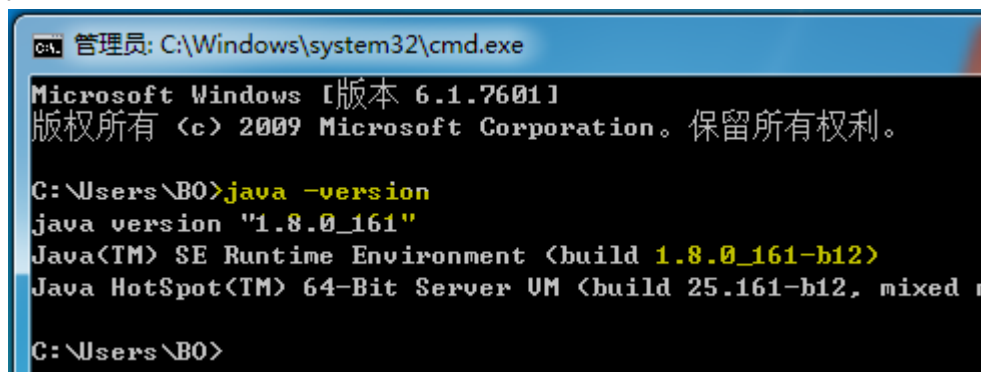


安装后，无需在系统环境中配置 Path 等变量

在 windows 桌面，按快捷键 win+r，进入运行，输入 cmd 命令运行命令行控制台



在命令行输入查看 java 版本命令，确定 JDK 已正确安装  
`java -version`

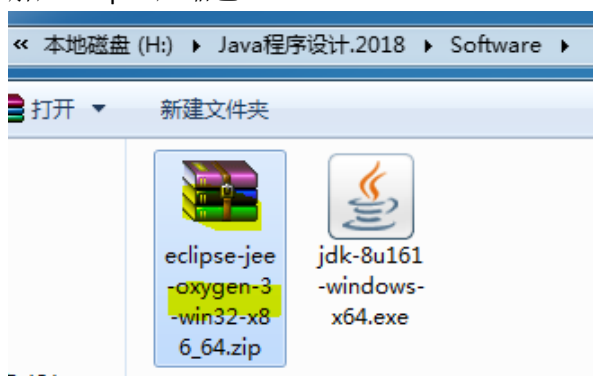


说明系统已正确安装 JDK，可以开始编写 Java 应用程序！

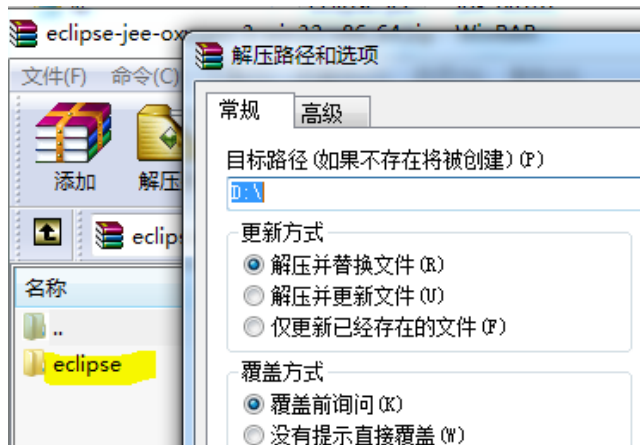
## 2.2 安装 Eclipse

建议使用压缩而非安装版 Eclipse

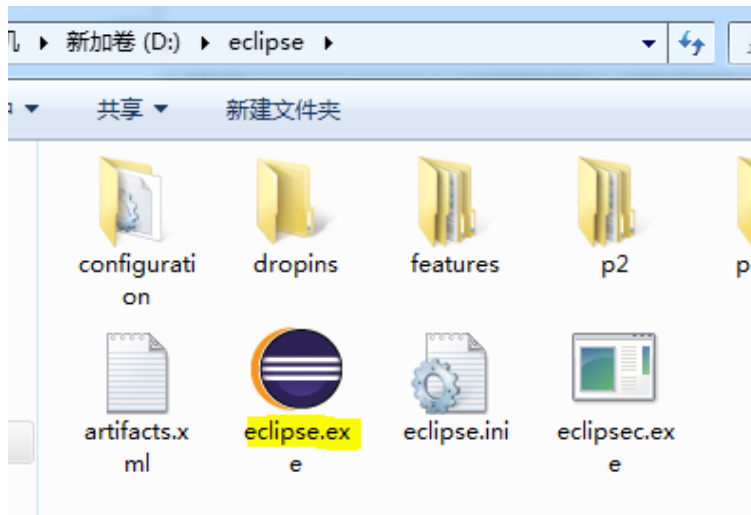
解压 Eclipse 压缩包



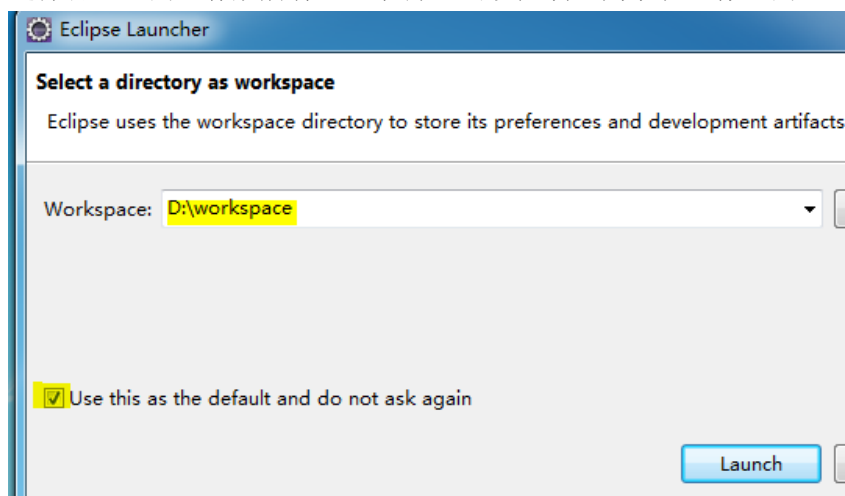
解压至合适的路径



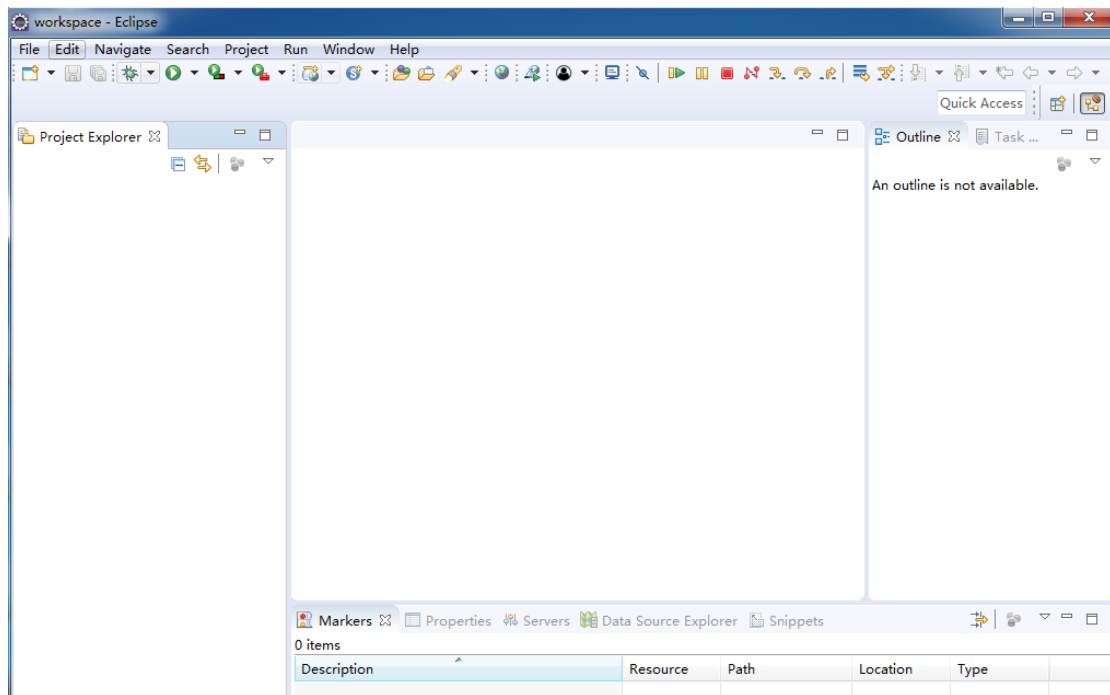
运行 eclipse



选择合适空间，存放所有 Java 程序，可以声明此为默认工作空间



Eclipse 基于 JavaEE 的默认布局结构

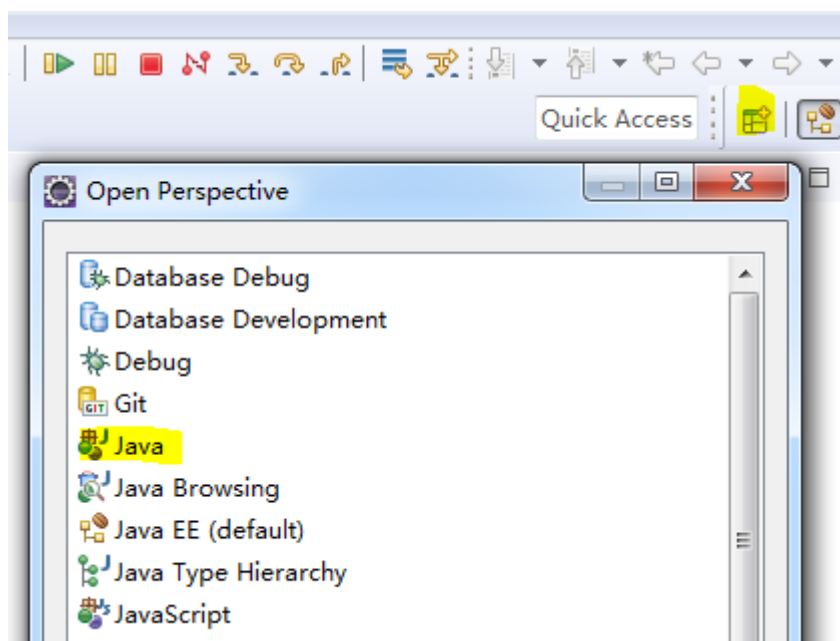


## 2.3 Eclipse 优化

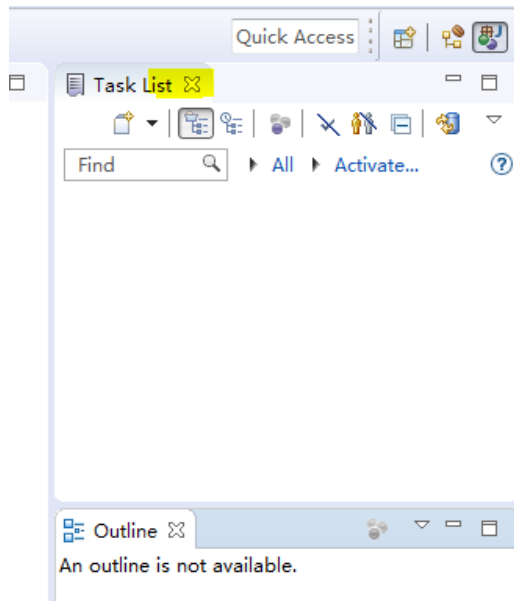
### 2.3.1 优化开发布局结构

以下为建议的基于 JavaSE 的开发布局结构

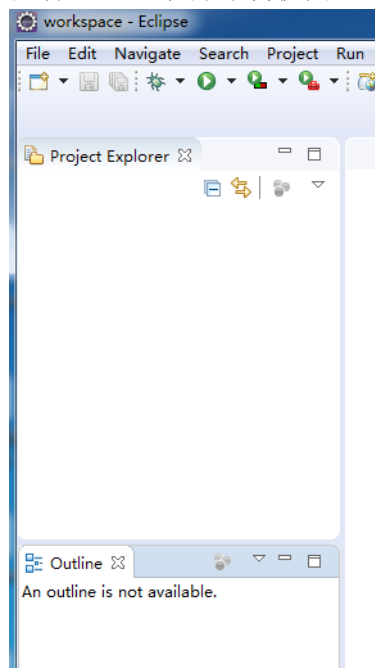
切换默认 JavaEE 开发视图模式，至 Java 开发视图模式。JavaEE 视图模式适合 Java Web 开发，Java 视图模式适合基本 JavaSE 的调试与开发。



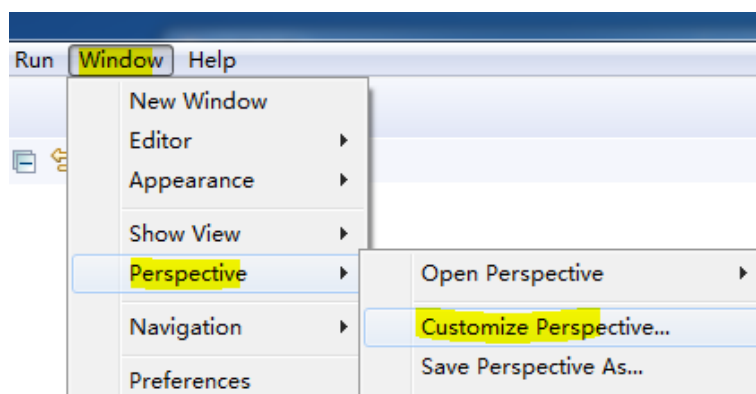
删除 Task list 视图



拖动 Outline 程序结构视图，至 Project Explorer 工程视图下方

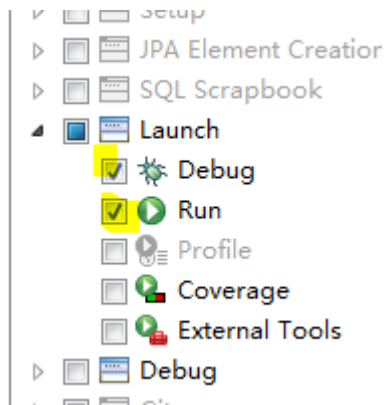


删除 tool bar 多余功能按钮

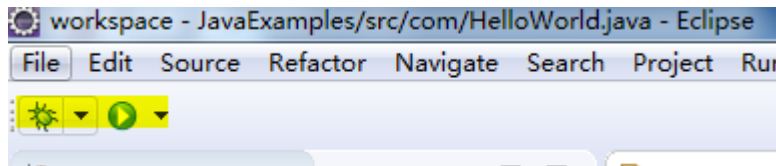




仅保留 debug run 即可

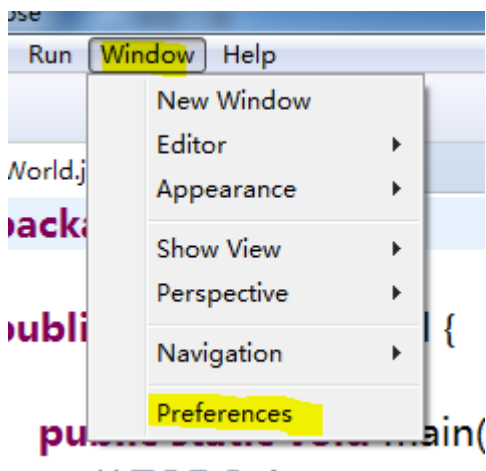


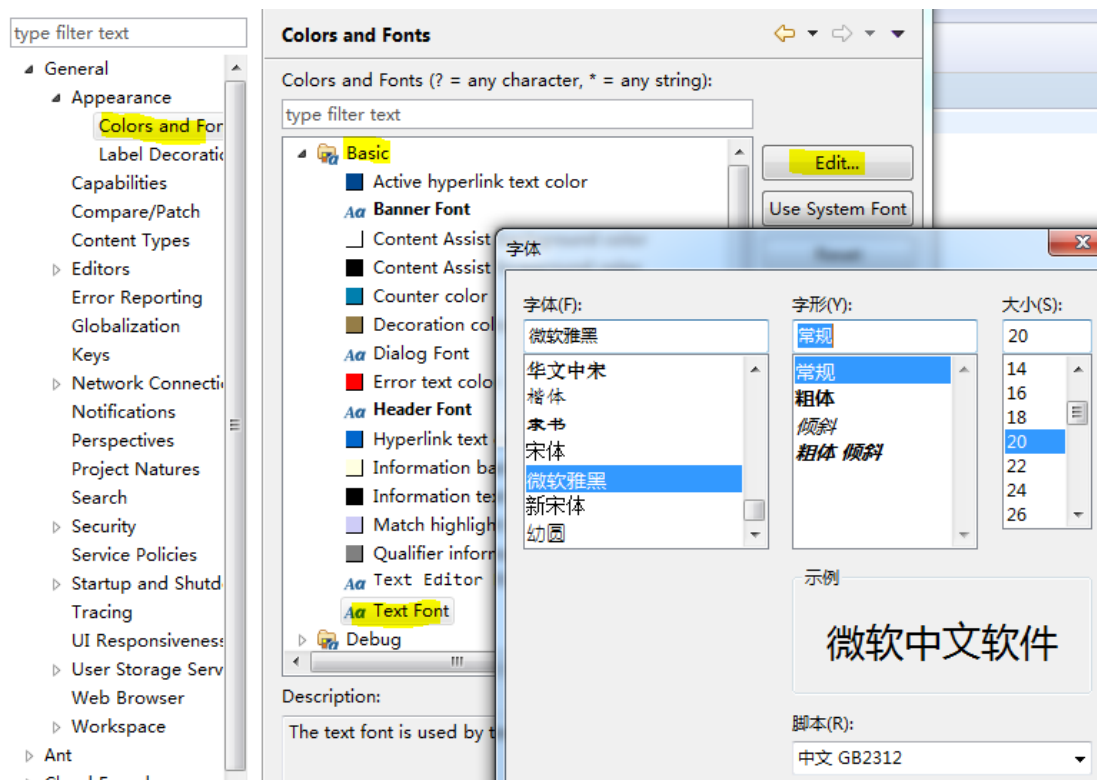
效果



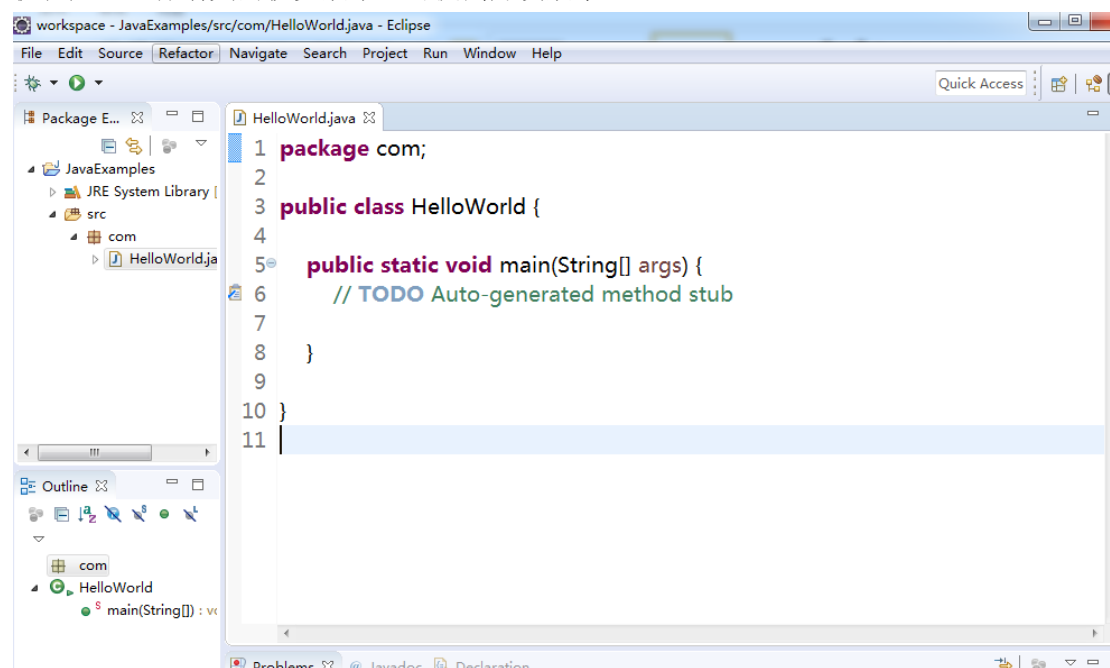
### 2.3.2 修改字体样式

Eclipse 默认字体样式视觉效果较差，可自定义





优化后，整体清爽的视觉效果，可提高开发效率



### 2.3.3 添加代码自动补全功能

输入任何字符都可触发 Java、XML、JSP、HTML 文件的自动补全功能。

XML JSP HTML 等的自动补全，可在 Web 开发技术课程中添加

Eclipse-window-Preferences-java-Editor-Content Assist

Auto activation triggers for Java:

```
@._-+*$%*abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYZ()[]><
```

Eclipse-window-Preferences-XML-XML Files-Editor-Content Assist

Prompt when these characters are inserted:

```
@._-+*$%*abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYZ()[]><
```

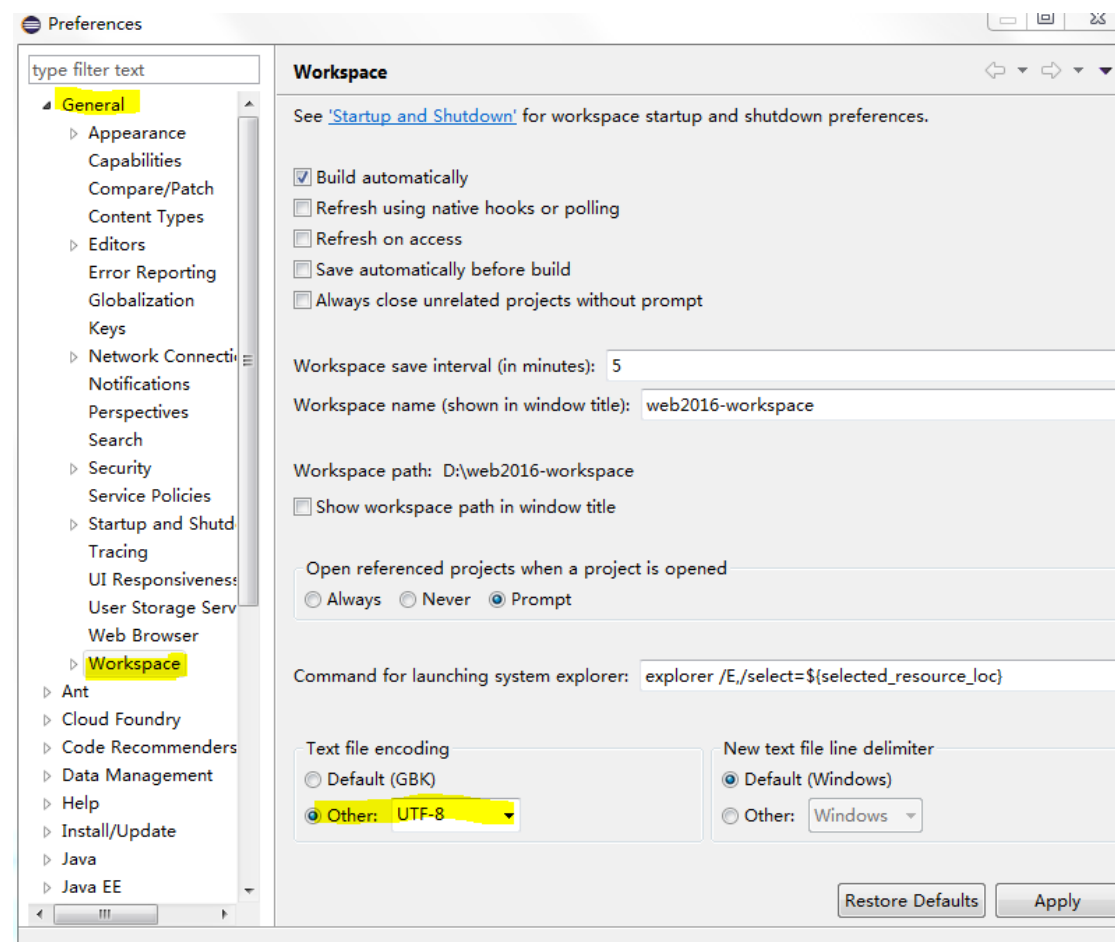
Eclipse-window-Preferences-Web-HTML Files-Editor-Content Assist

Prompt when these characters are inserted:

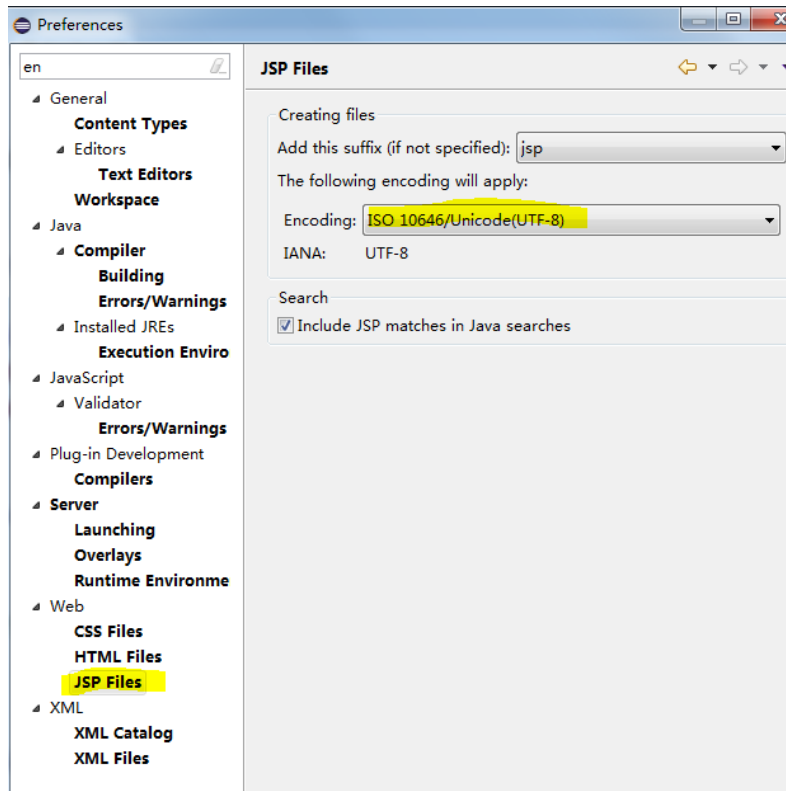
```
@._-+*$%*abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYZ()[]><
```

### 2.3.4 统一编码

为防止出现中文乱码并便于开发，将工程、文件等编码统一为 UTF-8 格式

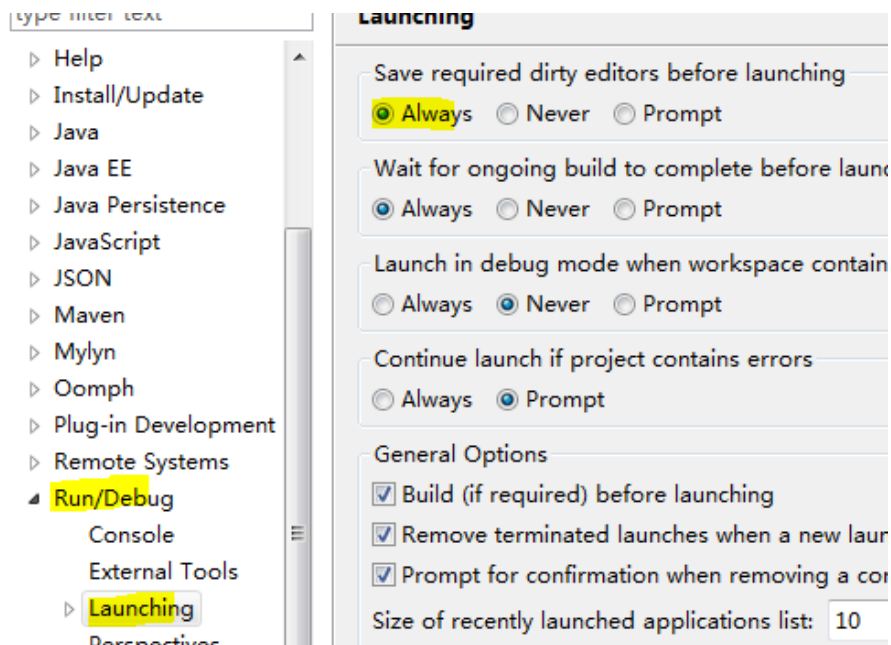


设置 JSP 文件格式



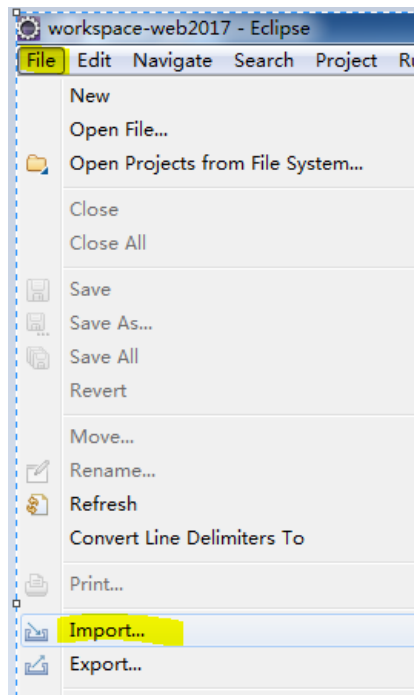
### 2.3.5 运行自动保存

运行时，自动保存未保存代码，避免相关代码忘记保存

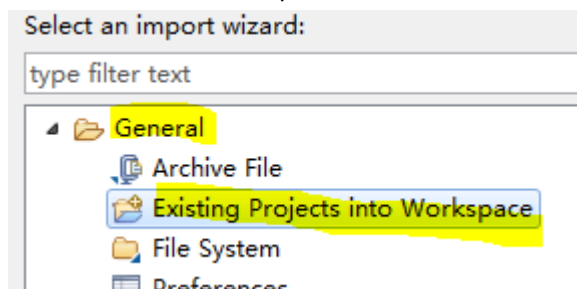


## 2.4 导入工程

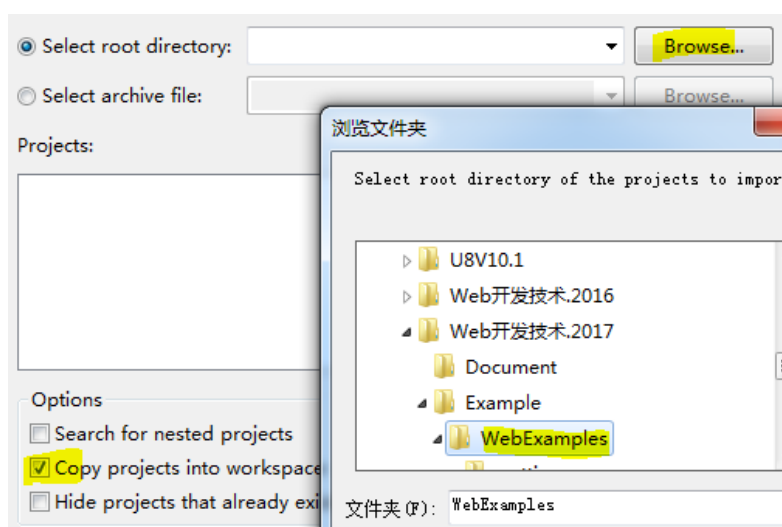
导入已有工程，至本地工作空间



选择导入工程到 workspace

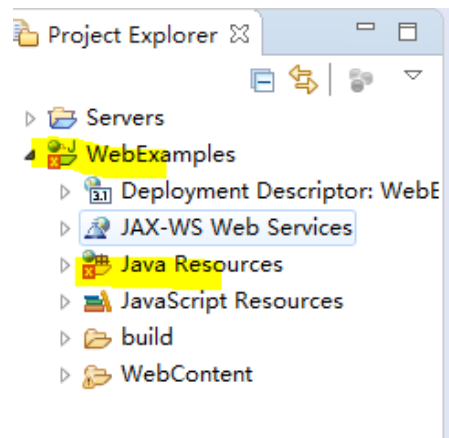


选择工程路径，选择复制到 workspace 工作空间，不会对原工程项目产生影响

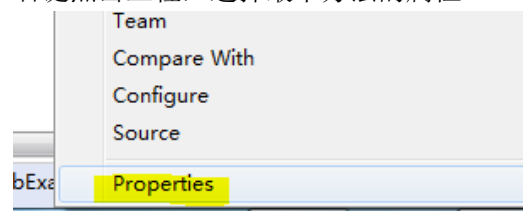


## 2.5 导入工程配置错误修正

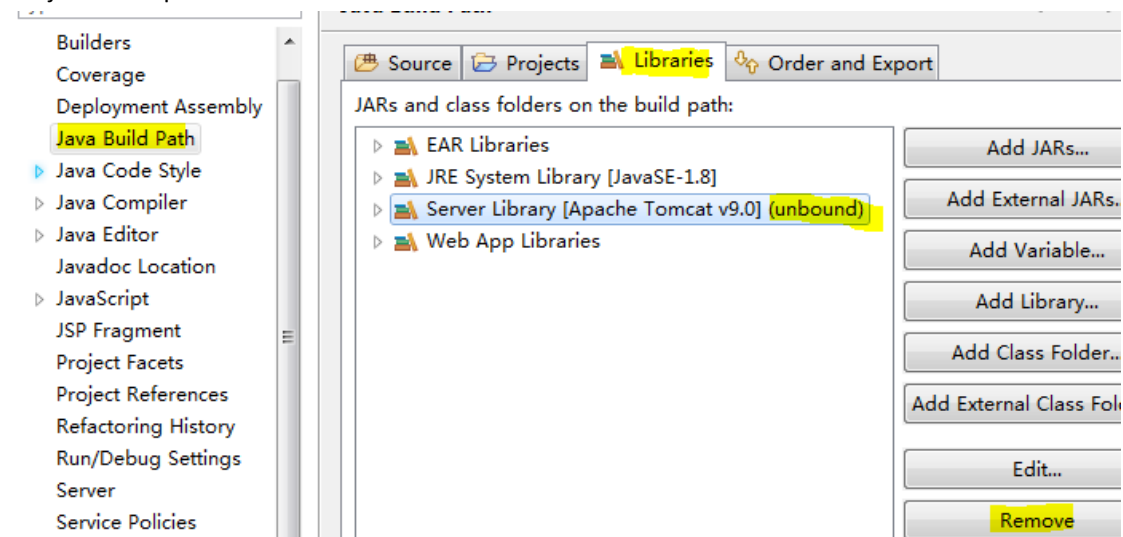
导入工程后，由于工程 eclipse 配置与本地 eclipse 配置不同，例如 JRE 配置，会显示错误信息



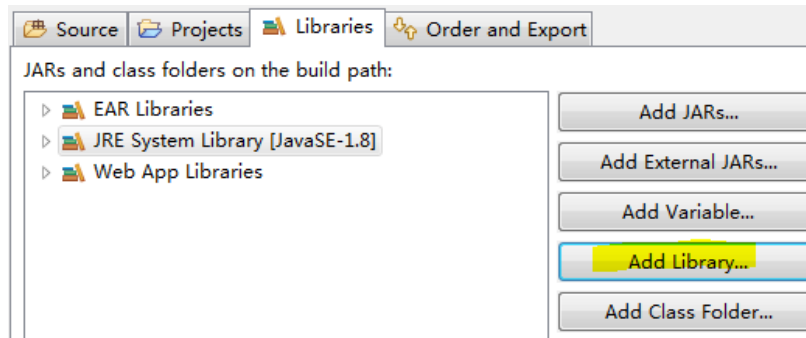
右键点击工程，选择最下方法的属性



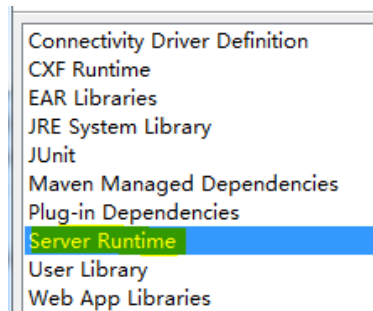
在 java build path 下，移除工程配置中 unbound 的库，例如 JRE，tomcat 服务器等



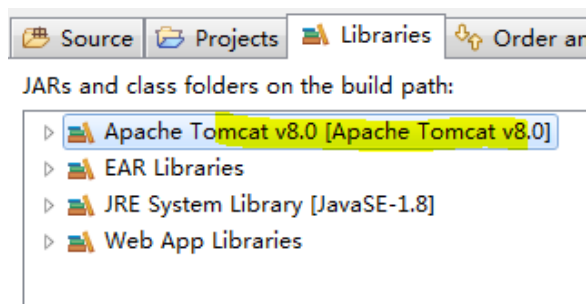
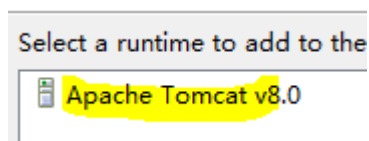
移除工程中原服务器配置，添加本地 eclipse 服务器的支持



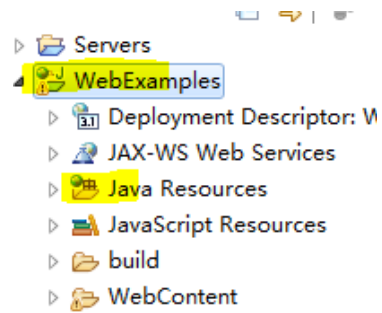
添加运行时服务器



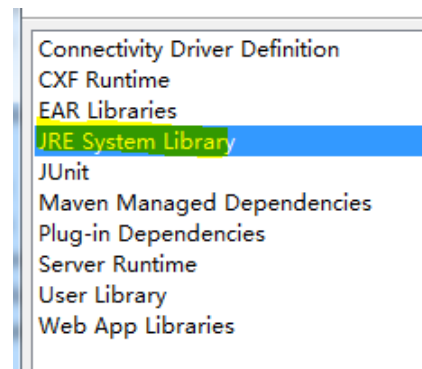
指定 eclipse 配置的服务器



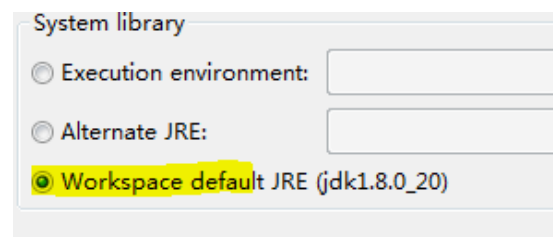
成功后，工程错误被修正



同理如果工程 JRE 配置错误 unbound，可添加本地 JRE 配置

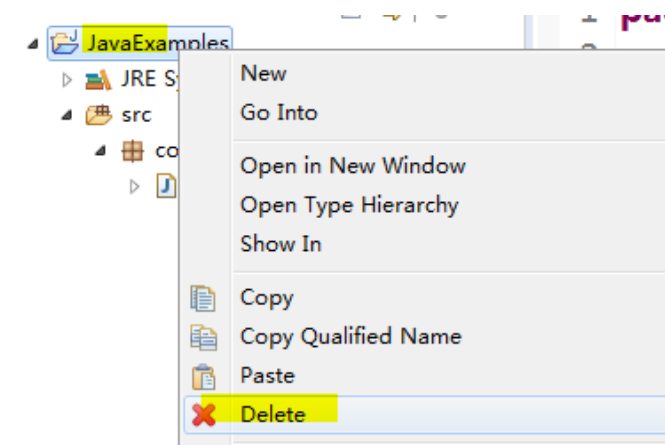


使用 eclipse workspace 默认 JRE 配置即可



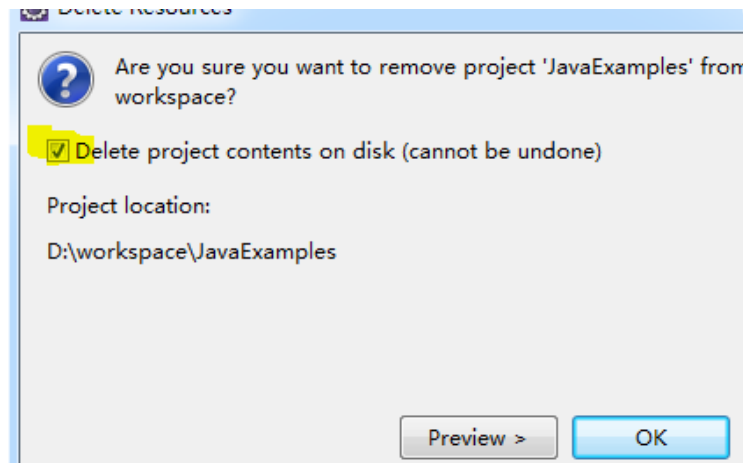
## 2.6 删除工程

右键选择欲删除的工程，选择删除





确定勾选 **delete project contents on disk**，在硬盘完全删除工作空间中的工程

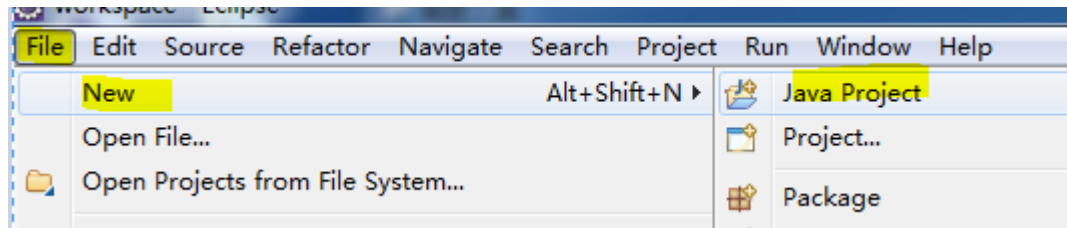


## 2.7 Eclipse 常用快捷键

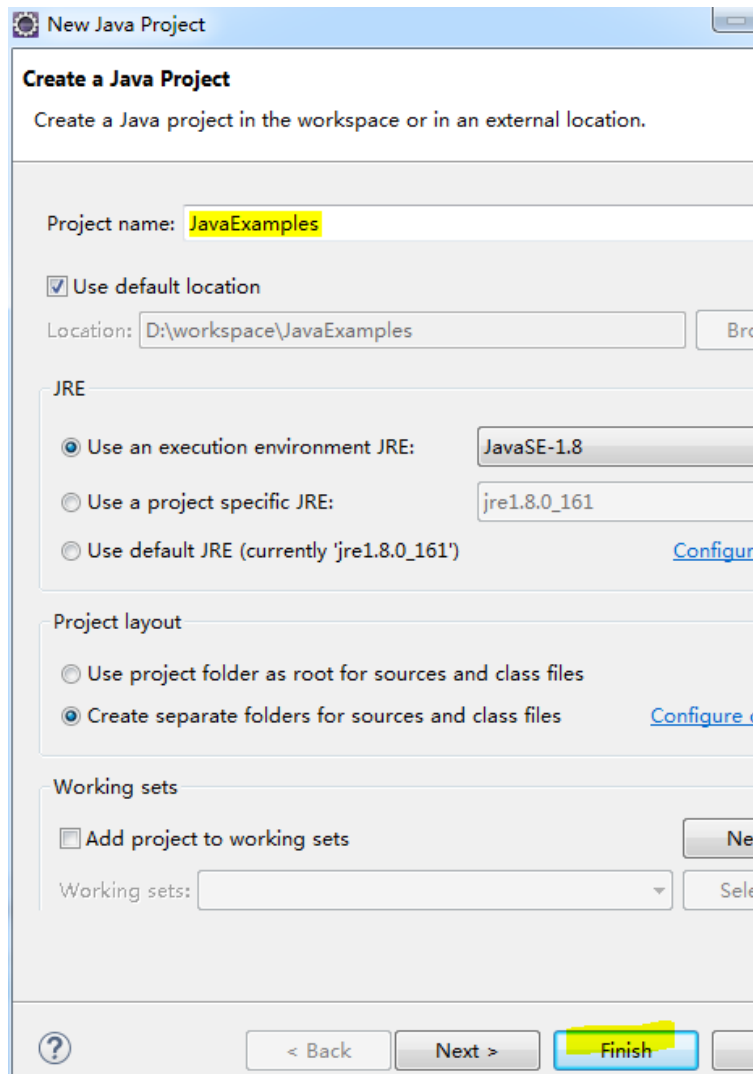
sys0	System.out.println()
Alt+shift+r	变量方法等重命名
Ctrl+shift+/	注释代码
Ctrl+shift+\	删除代码注释
Alt+shift+s	自动创建 Getter/setter，有参/无参构造函数，重写等
Ctrl+shift+o	删除 import 中未使用的成员
Ctrl+shift+f	选择代码后，重新排版
/*+回车确定	多行注释
/**+回车确定	文档注释

### 3 第一个 Java 程序

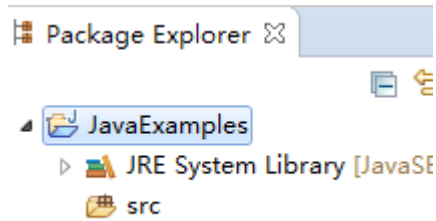
创建一个 Java project



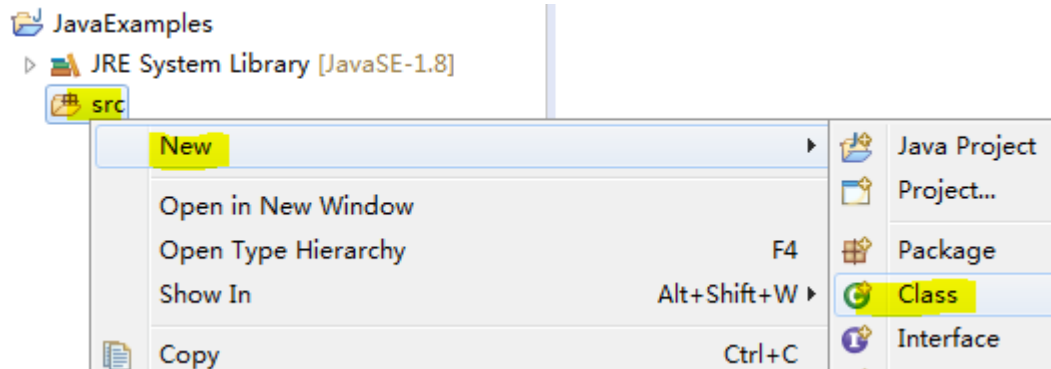
工程名称为 JavaExamples，其他设置全部默认



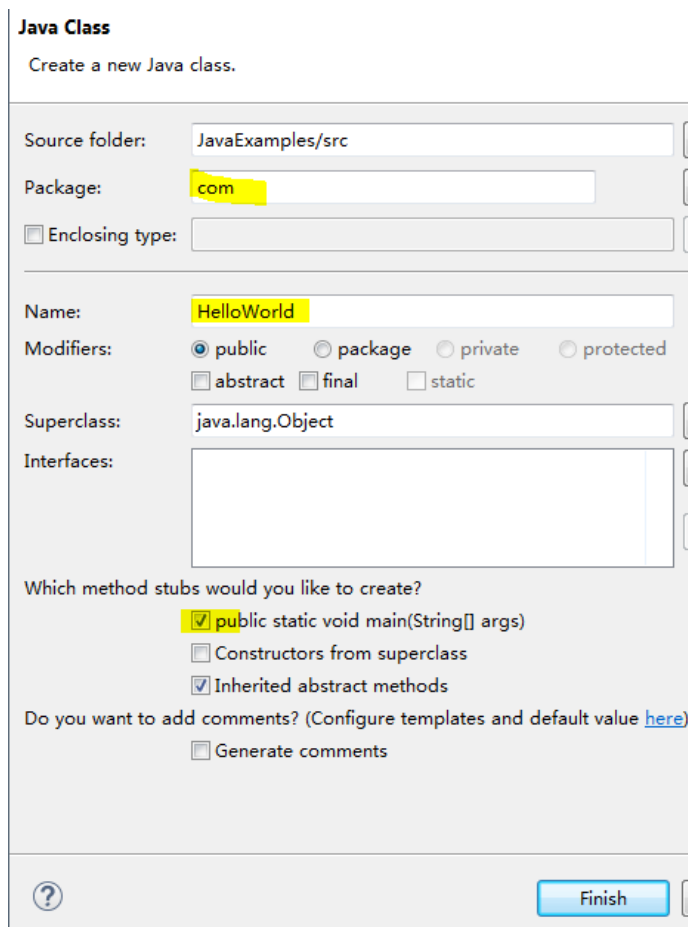
一个基本 JavaSE 工程，创建完成



右键单击 src，java 源码目录，创建第一个 Java class

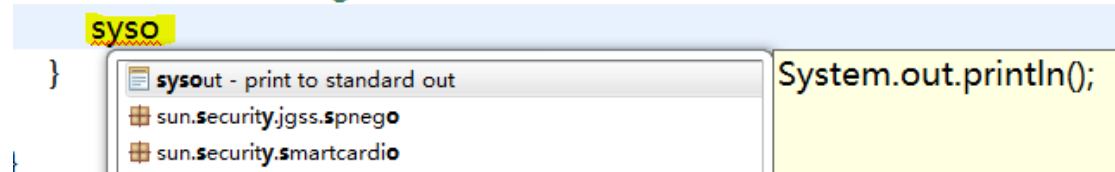


声明包、类的名称，由 eclipse 自动生成 main 主函数



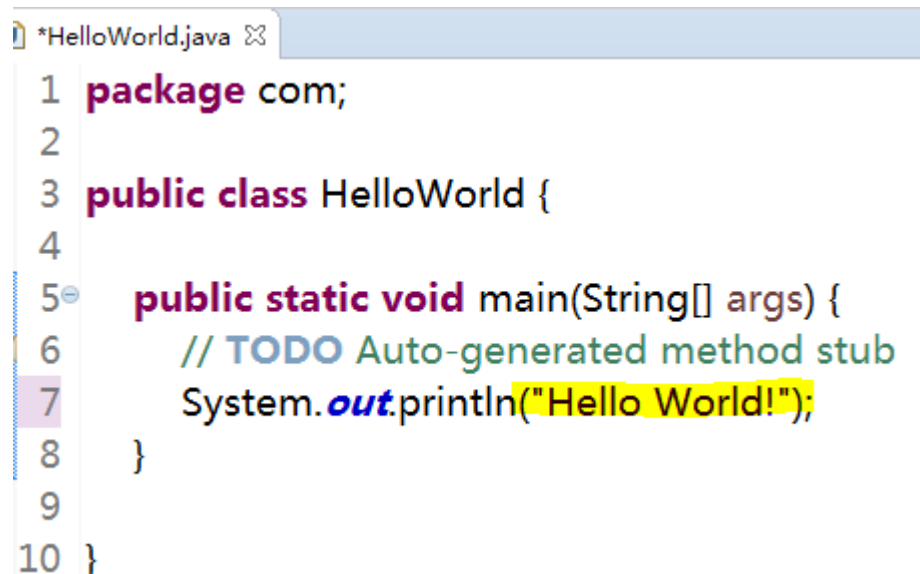
在主函数中输入 `sysout` 快捷命令，回车确定

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    sysout  
}
```



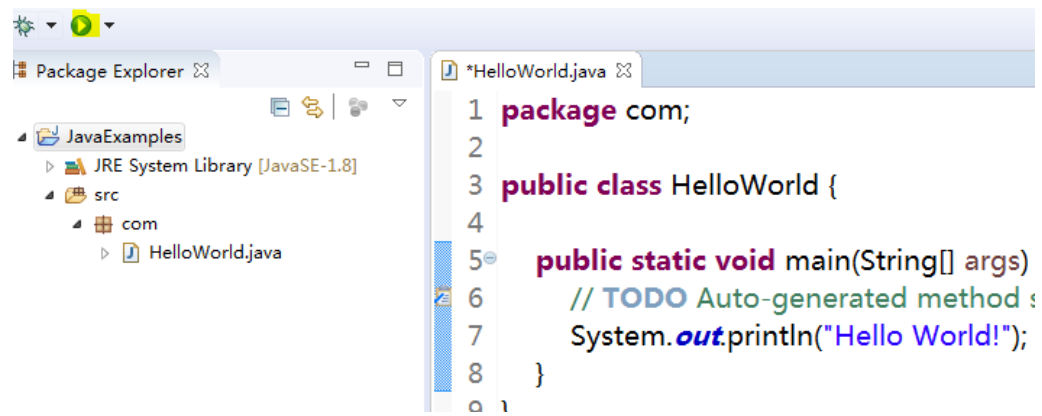
自动生成输出显示方法，在输出方法键入输出内容

```
package com;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("Hello World");  
    }  
}
```

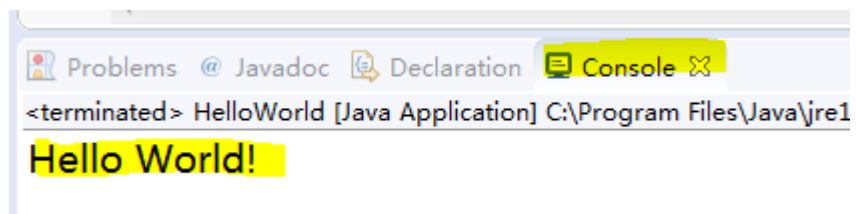


```
*HelloWorld.java  
1 package com;  
2  
3 public class HelloWorld {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub  
7         System.out.println("Hello World!");  
8     }  
9  
10 }
```

点击 `run`，运行按钮

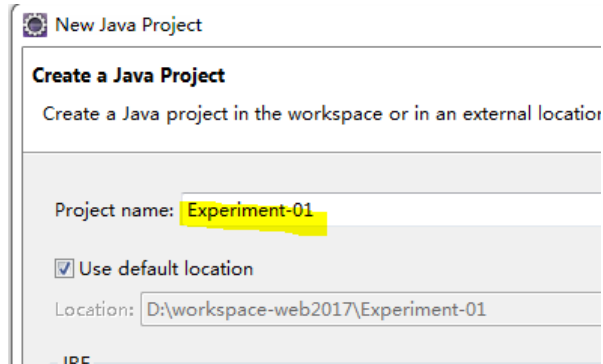


在控制台显示输出

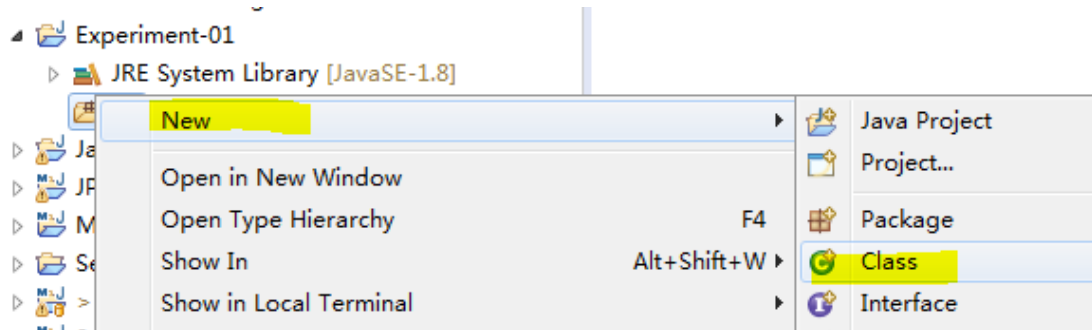


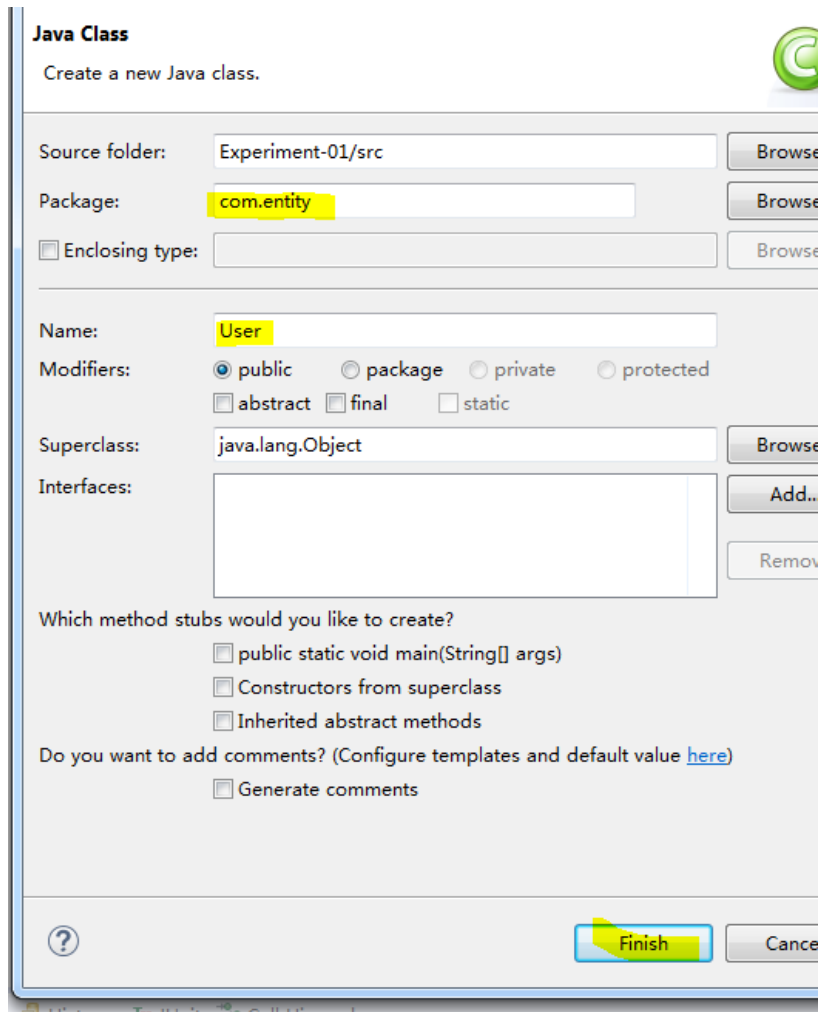
## 4 封装设计实例

基于 3 第一个 Java 程序方法，创建 Java project，Experiment-01



在 com.entity 下，创建 User 类





```
User.java
1 package com.entity;
2
3 public class User {
4
5 }
6
```

### 程序代码必须严格遵守 Java 代码样式规范

**需求 0:** 设计一个企业员工用户类型，包含

用户名，员工号：企业员工号为数字；

当前是否在职：只有在职与不在职 2 种可能；默认为在职

用户所属部门编号：企业的部门编号固定，市场部门为 1，售后；每一个用户都必须属于一个部门

当前员工总数：无需实现增删操作

为每个属性添加单行注释，为查询总数方法提供文档注释

尝试手动编写属性的 `getter/setter` 方法

应声明些什么属性？每个属性声明为什么类型合适？员工所属部门编号，为外部创建对象时使用；当前员工总量应如何声明？应允许如何操作？

```
package com.entity;

public class User {
    // 销售部门编号
    public static final int MARKET = 1;
    // 售后部门编号
    public static final int AFTER_MAKKET = 2;
    // 员工总数
    private static int amount;

    // 姓名
    private String name;
    // 员工号
    private int number;
    // 在职
    private boolean service = true;
    // 部门编号
    private int departNumber;

    /**
     * 获取当前员工总数
     * @return 员工数
     */
    public static int getAmount() {
        return amount;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getNumber() {
        return number;
    }
}
```



```
}

public void setNumber(int number) {
    this.number = number;
}

public boolean isService() {
    return service;
}

public void setService(boolean service) {
    this.service = service;
}

public int getDepartNumber() {
    return departNumber;
}

public void setDepartNumber(int departNumber) {
    this.departNumber = departNumber;
}
}
```

**需求+1:** 创建一个包含主函数的测试类, 创建一个用户对象, 通过 setter 方法添加属性数据, 添加部门编号时, 使用常量

在 com 下, 创建包含主函数的测试类 test

```
package com.entity;

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        User user = new User();
        user.setName("BO");
        user.setNumber(123456);
        user.setService(true);
        user.setDepartNumber(User.AFTER_MAKKET);
    }
}
```

**需求+1:** 为用户类添加性别属性，基于枚举类型实现  
如何创建枚举类型？性别枚举类型声明在哪里合适？

性别只有 2 个固定选项，因此使用枚举，性别仅用户类使用，因此定义在内部，将性别定义为用户的属性，并提供 getter/setter 方法

在 com.entity 下，创建 User02 类，主要代码

```
package com.entity;

public class User02 {
    // 仅包含需求实现，省略User中代码
    private Sex sex;
    public enum Sex {
        MALE, FEMALE
    }
    public Sex getSex() {
        return sex;
    }
    public void setSex(Sex sex) {
        this.sex = sex;
    }
}
```

测试封装属性方法代码

```
User02 user02 = new User02();
user02.setSex(Sex.MALE);
```

**需求+1:** 设计企业部门类型，包括部门名称，机构编号；  
由于是固定的，依然可以使用常量  
在 com.entity 下，创建 Group

```
package com.entity;

public class Group {
    // 销售部门编号
    public static final int MARKET = 1;
    // 售后部门编号
    public static final int AFTER_MAKKET = 2;
    private String name;
    private int number;
    // 省略getter/setter方法
}
```

将机构类型作为用户的一个属性，替换原实现

```
package com.entity;
public class User02 {
    // 仅包含需求实现，省略User中代码
    private Sex sex;
    private Group group;
    // 省略getter/setter方法

    public enum Sex {
        MALE, FEMALE
    }
}
```

需求+1: 创建 User02 对象，封装属性测试

由于用户对象需要部门属性，因此需先创建部门对象，在将部门对象封装到用户对象的属性中

理解任意类型都可作为任意类型的属性

封装测试代码

```
Group group = new Group();
group.setName("销售");
group.setNumber(Group.MARKET);
User02 user02 = new User02();
user02.setGroup(group);
```

包含所有测试的代码

```
package com;

import com.entity.Group;
import com.entity.User;
import com.entity.User02;
import com.entity.User02.Sex;

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        testUser();
        testUser02();
        testGroup();
    }

    private static void testUser() {
```

```
User user = new User();
user.setName("BO");
user.setNumber(123456);
user.setService(true);
user.setDepartNumber(User.AFTER_MARKET);
}

private static void testUser02() {
    User02 user02 = new User02();
    user02.setSex(Sex.MALE);
}

private static void testGroup() {
    Group group = new Group();
    group.setName("销售");
    group.setNumber(Group.MARKET);
    User02 user02 = new User02();
    user02.setGroup(group);
}
}
```

## 5 流程控制循环与数组实例

创建 Java project, Experiment-02

### 需求 0

创建 1 个包含 5 个元素的整型数组，添加考试成绩，将其个低于 60 分的成绩打印显示。

成绩(可自定义): 94,64,55,59,88

如何初始化一个整型数组？使用基本 for 循环，以及 foreach 循环分别实现

在 com 下创建包含 main 主函数的 Test 类，声明一个实现以上需求方方法

```
package com;

public class Test {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        testArray();
    }

    private static void testArray() {
        int[] results = { 94, 64, 55, 59, 88 };
        for (int i = 0; i < results.length; i++) {
            if (results[i] < 60) {
                System.out.println(results[i]);
            }
        }
        for (int i : results) {
            if (i < 60) {
                System.out.println(i);
            }
        }
    }
}
```

打印输出结果

```
55
59
55
59
```

创建学生，包含学号，姓名，成绩，班级，属性；创建 4 个参数的构造函数用于初始化；班级只可能是 1 班或 2 班  
用于封装测试

在 com.entity 下，创建 Student 类，允许使用快捷键 alt+shift+s，生成 getter/setter 方法

```
package com.entity;

public class Student {
    public static final String CLASS_1 = "1班";
    public static final String CLASS_2 = "2班";
    private int number;
    private String name;
    private int result;
    private String clazz;

    public Student(int id, String name, int result, String clazz) {
        super();
        this.number = id;
        this.name = name;
        this.result = result;
        this.clazz = clazz;
    }
    // 省略getter/setter方法
}
```

#### 需求+1

通过 1 个学生数组类型的静态变量，模拟学生数据表，创建静态变量初始化方法，提供外部使用的方法。

如何创建学生类型的数组？如何初始化并赋值？如何创建学生数组类型的静态变量，并提供对外使用的方法？

在数组中，模拟添加 2-5 个元素，即可

理解基于引用类型数组的引用传递

在 com 下，创建 Database 类

```
package com;

import com.entity.Student;

public class Database {
    private static Student[] students = createStudents();

    private static Student[] createStudents() {
        Student stu1 = new Student(1001, "BO", 94, Student.CLASS_1);
        Student stu2 = new Student(1021, "SUN", 64, Student.CLASS_1);
        Student stu3 = new Student(1024, "ZHANG", 44, Student.CLASS_1);
        Student stu4 = new Student(1031, "LIU", 59, Student.CLASS_2);
    }
}
```

```
Student stu5 = new Student(1035, "GUO", 88, Student.CLASS_2);
Student[] students = new Student[5];
students[0] = stu1;
students[1] = stu2;
students[2] = stu3;
students[3] = stu4;
students[4] = stu5;
return students;
}
public static Student[] getStudents() {
    return students;
}
}
```

#### 需求+1

创建学生相关服务，提供方法，实现显示给定学生数组的姓名与成绩

抽象设计该方法，方法本身仅应基于给定的学生数组，不应在方法内部直接获取数据库中的学生数据，此方法不应与任何数据源耦合

通过 foreach 实现遍历

理解 foreach item in items 中，变量 item 引用每一次遍历的元素对象

在 com.service 下，创建 StudentService 类，用于执行 student 相关操作

```
package com.service;

import com.entity.Student;

public class StudentService {

    /**
     * 基于foreach，打印数组中所有学生的姓名及成绩
     */
    public static void showStudent(Student[] students) {
        for (Student s : students) {
            System.out.println(s.getName() + " / " + s.getResult());
        }
    }
}
```

#### 需求+1

在 StudentService，提供一个方法，计算班级成绩平均分

抽象设计该方法，方法本身仅应基于给定的学生数组和班级，计算平均分调用者，传入的班级不存在怎么办？

```
//基于给定班级，计算平均分，班级不存在，为0分
public static float computAverage(Student[] stus, String clazz) {
    float totalScore = 0f;
    int num = 0;
    for (Student s : stus) {
        if (s.getClazz() == clazz) {
            num = num + 1;
            totalScore = totalScore + s.getResult();
        }
    }
    // 如果班级参数不存在，num = 0，0不能作为除数，则直接返回0
    return num == 0 ? 0 : totalScore / num;
}
```

#### 需求+1

系统中已经存在一组学生，现向系统添加一组学生，但是新添加的学生可能已经存在，则应剔除重复学生

在 StudentService 中声明方法，提供源数组与新数组，方法内部基于学号判断比较，将新学生数组返回

应使用嵌套循环比较新旧数组，哪个数组在外层循环较好？不存在如何处理？

数组的长度是不可变的，改变数组长度，例如添加元素，需创建新数组。但新数组中的元素对象可引用原数组中的元素对象

数组不能在循环中改变长度

课程学时有限，JDK，第三方库，各种类接口方法，是不可能全部讲到的。因此，学生应掌握阅读相关 API 文档技能，通过网络学习巩固知识。

java.util.Arrays 类，提供了一系列针对数组的操作。以下 copyOf()方法实现了，对源数组扩展的功能。基于给定数组和新的长度，创建返回一个新的数组，源数组中的元素保留，位置索引不变，可在新位置添加元素

```
public static T[] copyOf(T[] original, int newLength)
```

<https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#copyOf-T:A-int->

```
public static Student[] getNewStudents(Student[] oldList, Student[] newList) {
    // 创建1个空学生数组，用于存放新学生
    Student[] students = new Student[0];
    // 遍历新列表
    for (Student nStu : newList) {
        // 新列表中学生是否存在的标识
        boolean exist = false;
        // 遍历旧列表
```



```
for (Student oStu : oldList) {  
    if (oStu.getNumber() == nStu.getNumber()) {  
        exist = true;  
        break;  
    }  
}  
  
// 不存在则将此学生追加到数组  
if (!exist) {  
    // 将数组长度+1  
    students = Arrays.copyOf(students, students.length + 1);  
    // 在新数组的最后追加不存在的学生  
    students[students.length - 1] = nStu;  
}  
}  
return students;  
}
```

在 Test 类中，声明测试用数组。创建 1 个包含已存在学生及新学生的数组

```
// 模拟导入4个学生，其中2个不存在  
private static Student[] getNewStudentList() {  
    Student stu1 = new Student(1001, "BO", 94, Student.CLASS_1);  
    // 不重复的学生  
    Student stu2 = new Student(1088, "SHI", 68, Student.CLASS_2);  
    Student stu3 = new Student(1035, "GUO", 88, Student.CLASS_2);  
    // 不重复的学生  
    Student stu4 = new Student(1084, "HE", 70, Student.CLASS_2);  
    Student[] newList = new Student[4];  
    newList[0] = stu1;  
    newList[1] = stu2;  
    newList[2] = stu3;  
    newList[3] = stu4;  
    return newList;  
}
```

在 main 主函数调用相关方法完成，比对测试

```
// 方法仅负责将比对后的新学生返回  
Student[] newStudents = StudentService.getNewStudents(Database.getStudents(),  
getNewStudentList());  
for (Student s : newStudents) {  
    System.out.println("新学生：" + s.getName());  
}
```

-----获取  
新学生：SHI  
新学生：HE