

# java\_1

## java

内容：  
初识JAVA  
程序了解  
软件了解  
开发规范

## 互联网分布图



## java方向分布图

	Nov 2017	Nov 2016	Change	Programming Language	Ratings	Change
1	1			Java	13.231%	-5.52%
2	2			C	9.293%	+0.09%
3	3			C++	5.343%	-0.07%
4	5		▲	Python	4.482%	+0.91%
5	4		▼	C#	3.012%	-0.65%
6	8		▲	JavaScript	2.972%	+0.27%
7	6		▼	Visual Basic .NET	2.909%	-0.26%
8	7		▼	PHP	1.897%	-1.23%
9	16		▲	Delphi/Object Pascal	1.744%	-0.21%
10	9		▼	Assembly language	1.722%	-0.72%
11	19		▲	R	1.605%	-0.11%
12	15		▲	MATLAB	1.604%	-0.36%
13	14		▲	Ruby	1.593%	-0.39%
14	13		▼	Go	1.570%	-0.43%
15	10		▼	Perl	1.562%	-0.80%
16	26		▲	Scratch	1.550%	+0.47%
17	17			Visual Basic	1.489%	-0.43%
18	20		▲	PL/SQL	1.453%	-0.06%
19	11		▼	Objective-C	1.412%	-0.83%
20	12		▼	Swift	1.389%	-0.65%

## java介绍

java概述(计算机编程语言) Java是一门面向对象编程语言，不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

Java具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。Java可以编写桌面应用程序、Web应用程序、分布式系统和嵌入式系统应用程序等。

java最强大的优势跨平台。

## 编程环境

**JDK** ( Java Development Kit ) 称为Java开发包或Java开发工具，是一个编写Java的Applet小程序和应用程序的程序开发环境。JDK是整个Java的**核心**，包括了Java运行环境 ( Java Runtime Envirnment ) ，一些Java工具和Java的**核心类库** ( Java API ) 。不论什么Java应用服务器实质都是内置了某个版本的JDK ( Java Development Kit ) 称为**Java开发包**或Java开发工具，是一个编写Java的Applet小程序和应用程序的程序开发环境。JDK是整个Java的核心，包括了Java运行环境 ( Java Runtime Envirnment ) ，一些Java工具和Java的核心类库 ( Java API ) 。不论什么Java应用服务器实质都是**内置了某个版本**的JDK。主流的JDK是Sun公司发布的JDK，除了Sun之外，还有很多公司和组织都开发了自己的JDK，如，IBM公司开发的JDK，BEA公司的Jrocket，还有GNU组织开发的JDK。

JRE是个**运行环境**，JDK是个**开发环境**。因此**写Java程序**的时候需要JDK，而运行Java程序的时候就需要JRE。而JDK里面已经**包含**JRE，因此只要安装了JDK，就可以编辑Java程序，也可以正常运行Java程序。

\*JDK,JRE,JVM[Java Virtual Machine ( Java虚拟机 ) ]三者之间关系：JDK: java development kit, java开发工具包，针对开发者，里面主要**包含**jre, jvm, jdk源码包，以及bin文件夹下用于开发，编译运行的一些指令器。[jdk包含jre,jre里包含jvm]

---

## 编程工具

Eclipse：一个开放源代码的、基于Java的可扩展开发平台。

NetBeans：开放源码的Java集成开发环境，适用于各种客户机和Web应用。

IntelliJ IDEA：在代码自动提示、代码分析等方面的具有很好的功能。

MyEclipse：由Genuitec公司开发的一款商业化软件，是应用比较广泛的Java应用程序集成开发环境。

EditPlus：如果正确配置Java的编译器“Javac”以及解释器“Java”后，可直接使用EditPlus编译执行Java程序

---

## 工作原理

四方面组成：

- ( 1 ) Java编程语言
- ( 2 ) Java类文件格式
- ( 3 ) Java虚拟机
- ( 4 ) Java应用程序接口

编辑并运行一个Java程序时，需要同时涉及四方面。使用文字编辑软件 ( 如记事本、写字板、UltraEdit等 ) 或**集成开发环境** ( Eclipse、MyEclipse,idea等 ) 在**java源文件**中定义不同的类，通过**调用类** ( 这些类实现了Java API ) 中的方法来访问资源系统，把源文件编译生成一种二进制**中间码**，存储在**class文件**中，然后再通过运行与操作系统平台环境相对应的Java虚拟机来运行class文件，执行编译产生的字节码，调用class文件中实现的方法来满足程序的Java API调用。

---

## 安装工具

1. jdk[直接安装]

2. eclipse(英文版,不要汉化,工作全是使用英文开发工具)

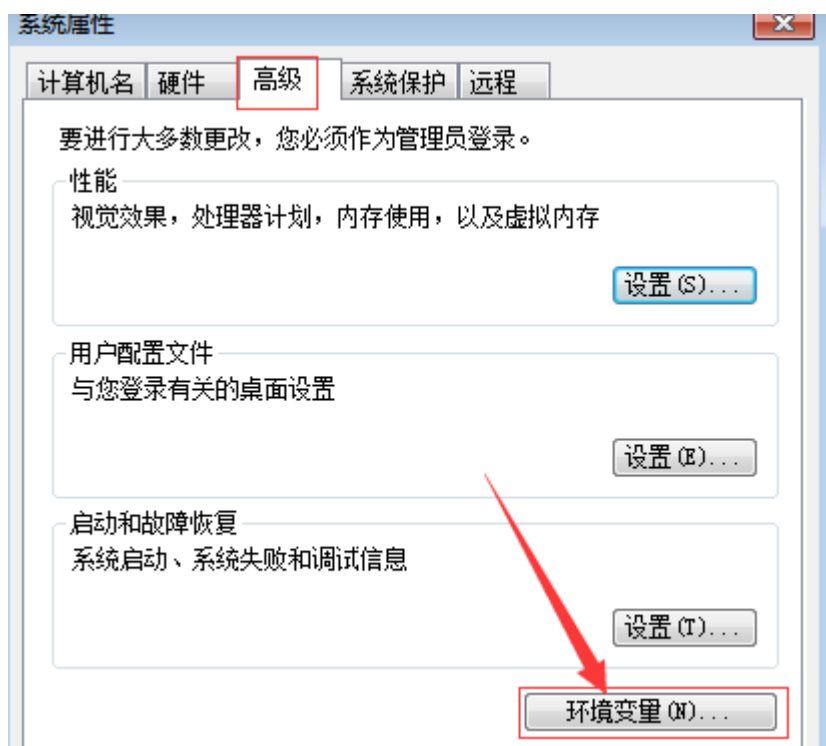
或myeclipse,idea等工具介绍

---

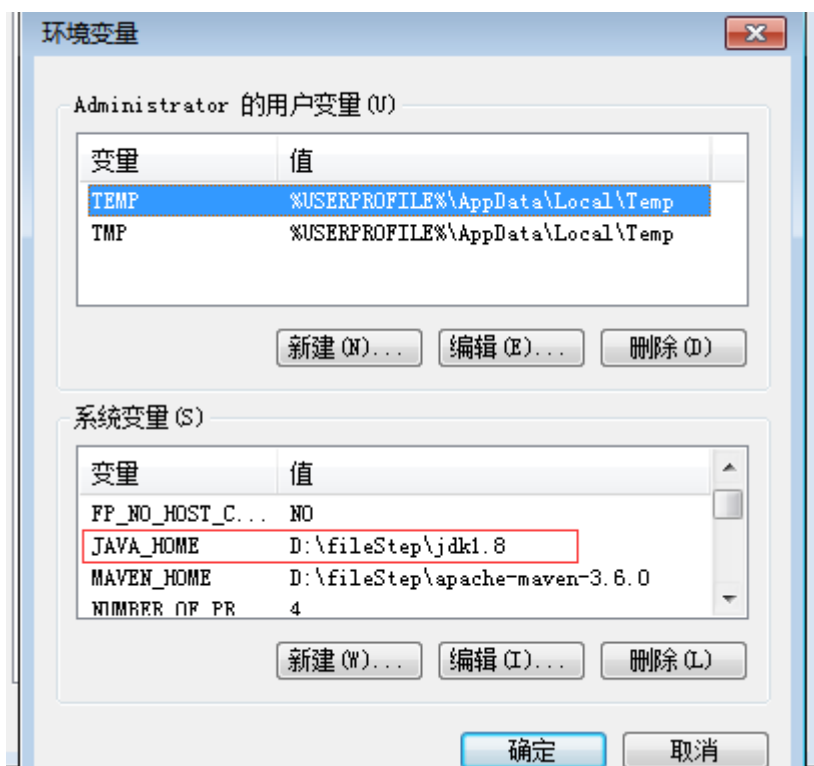
## 环境变量设置

1. 计算机 - 》属性 - 》高级系统设置 - 》高级 - 》环境变量

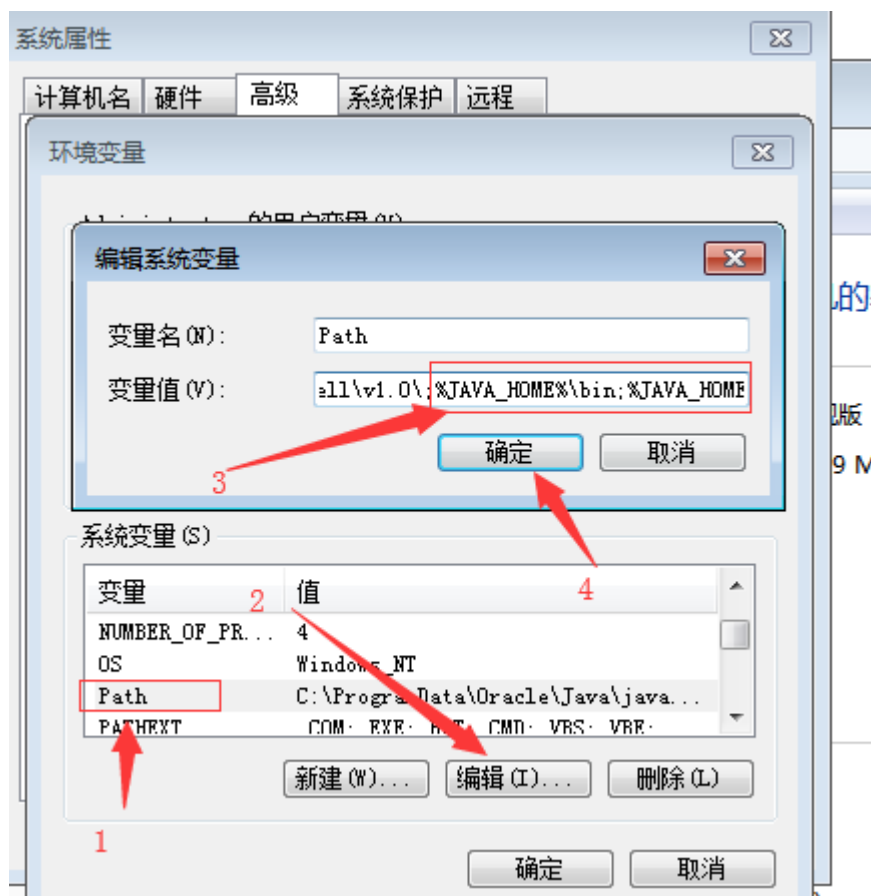
2. 寻找图示



3. JAVA\_HOME设置



4. path设置



\* : %JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

\* : 记得结束的分号必须是英文分号，包括后面编写代码全是使用英文分号

\* : win10系统的环境变量的path设置名称，不需要有结束分号。

## 安装查询jdk

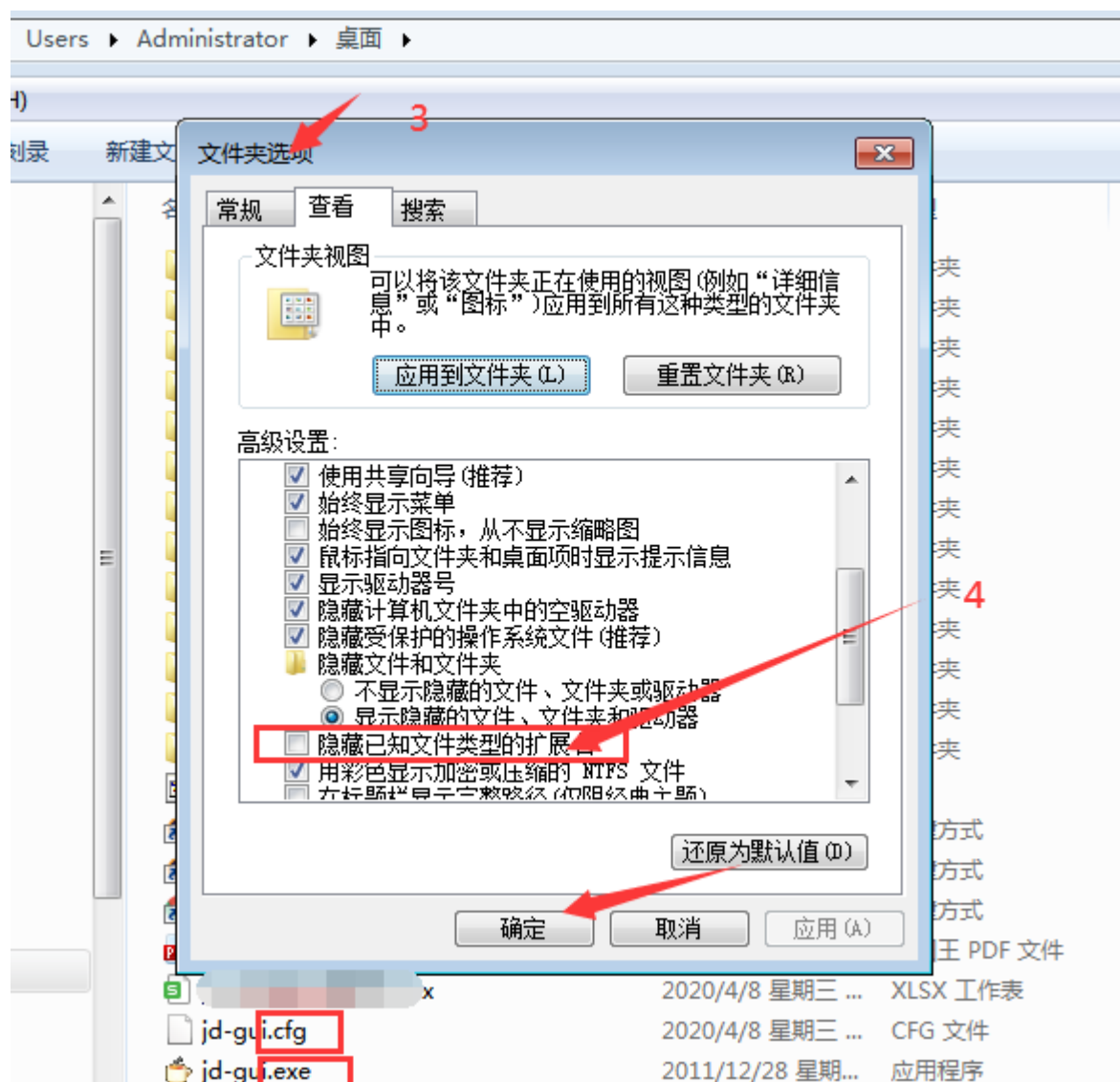
操作指令：

cmd下输入命令：java -version

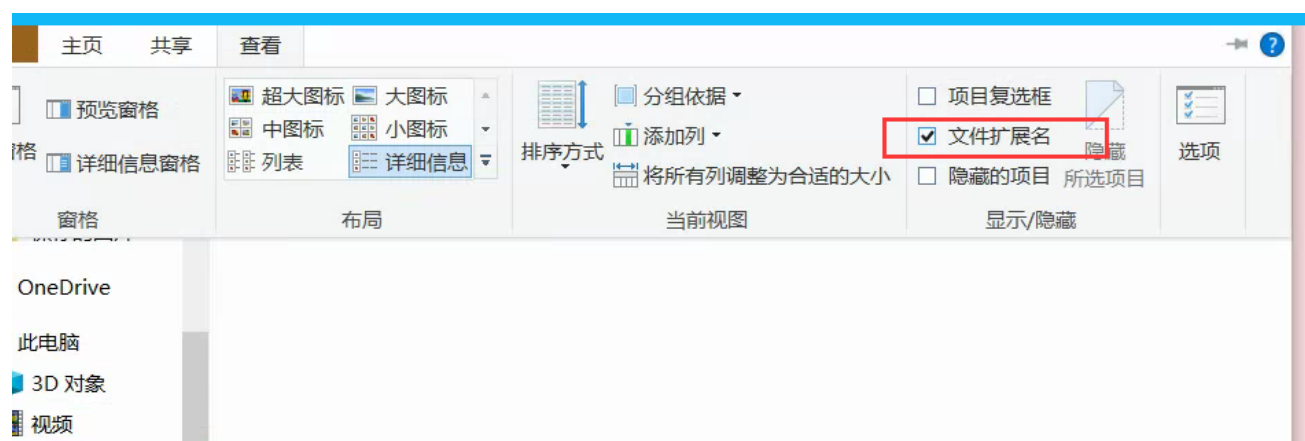
```
C:\Users\Administrator>java -version
java version "1.8.0_60"
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)
C:\Users\Administrator>
```

## 文件后台缀放开

win7系统如果文件后缀名不出来设置如下图



win10系统则设置如下图：

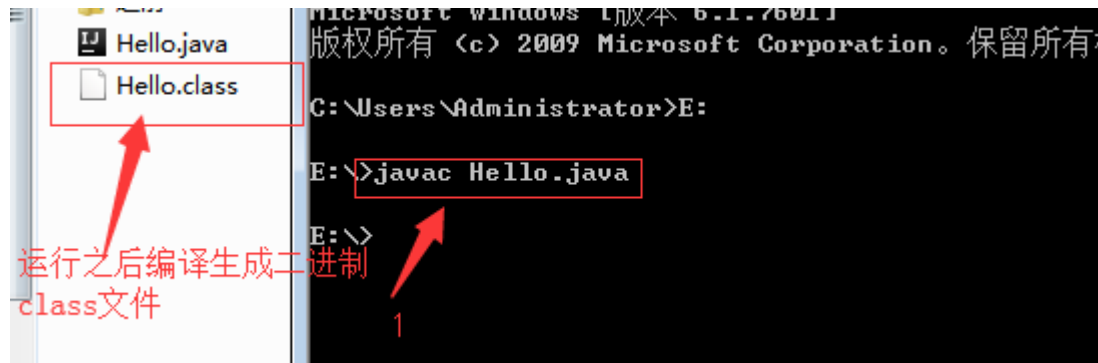


## 记事本编写程序

编写程序代码：

```
o. java x
public class Hello{
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```

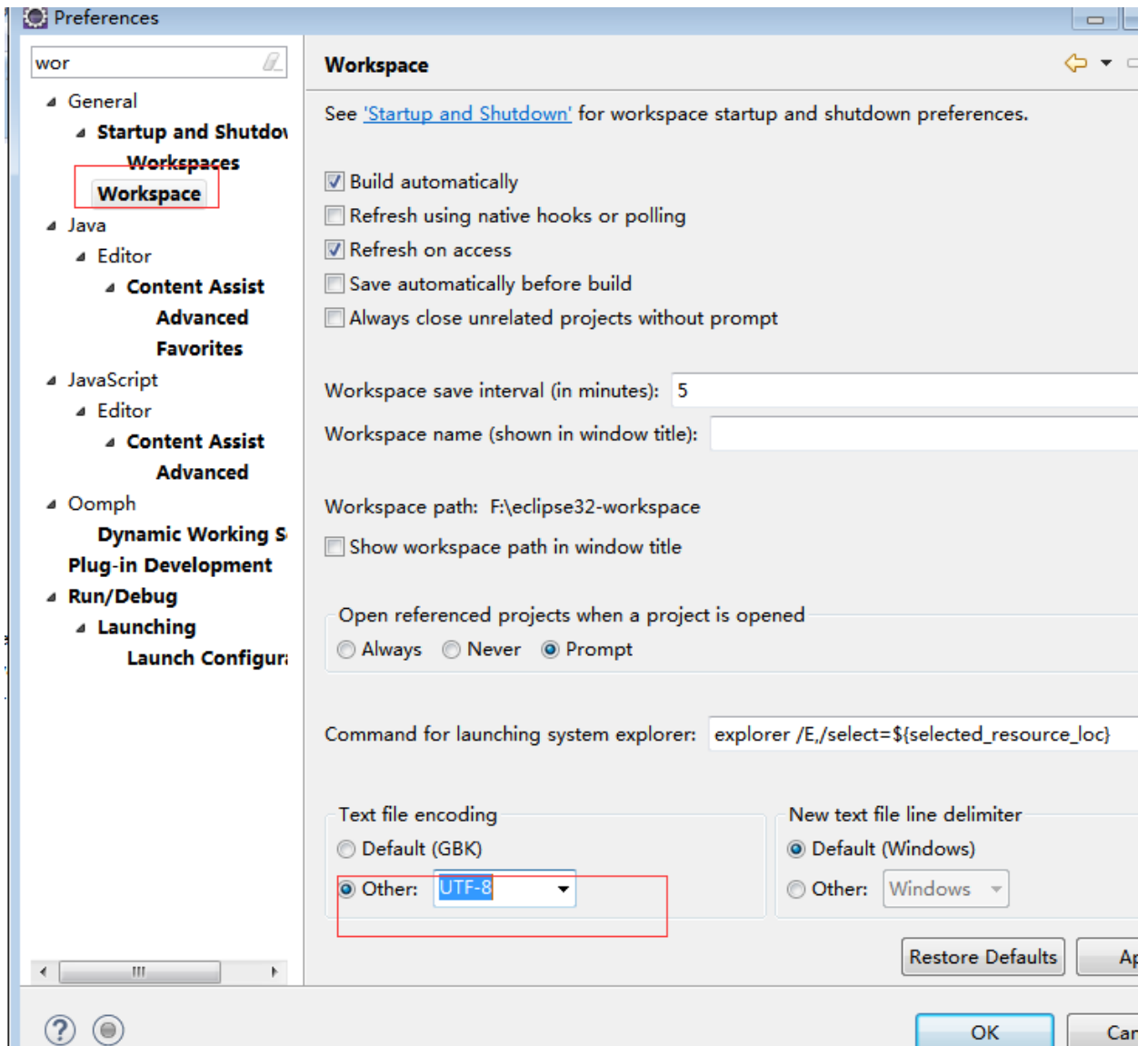
使用cmd输入命令javac 类名称，让其编译生成一个对应的class类[字节码]文件



直接在cmd里继续输入：**java** Hello回车，则会看到输出打印的信息，如hello world...

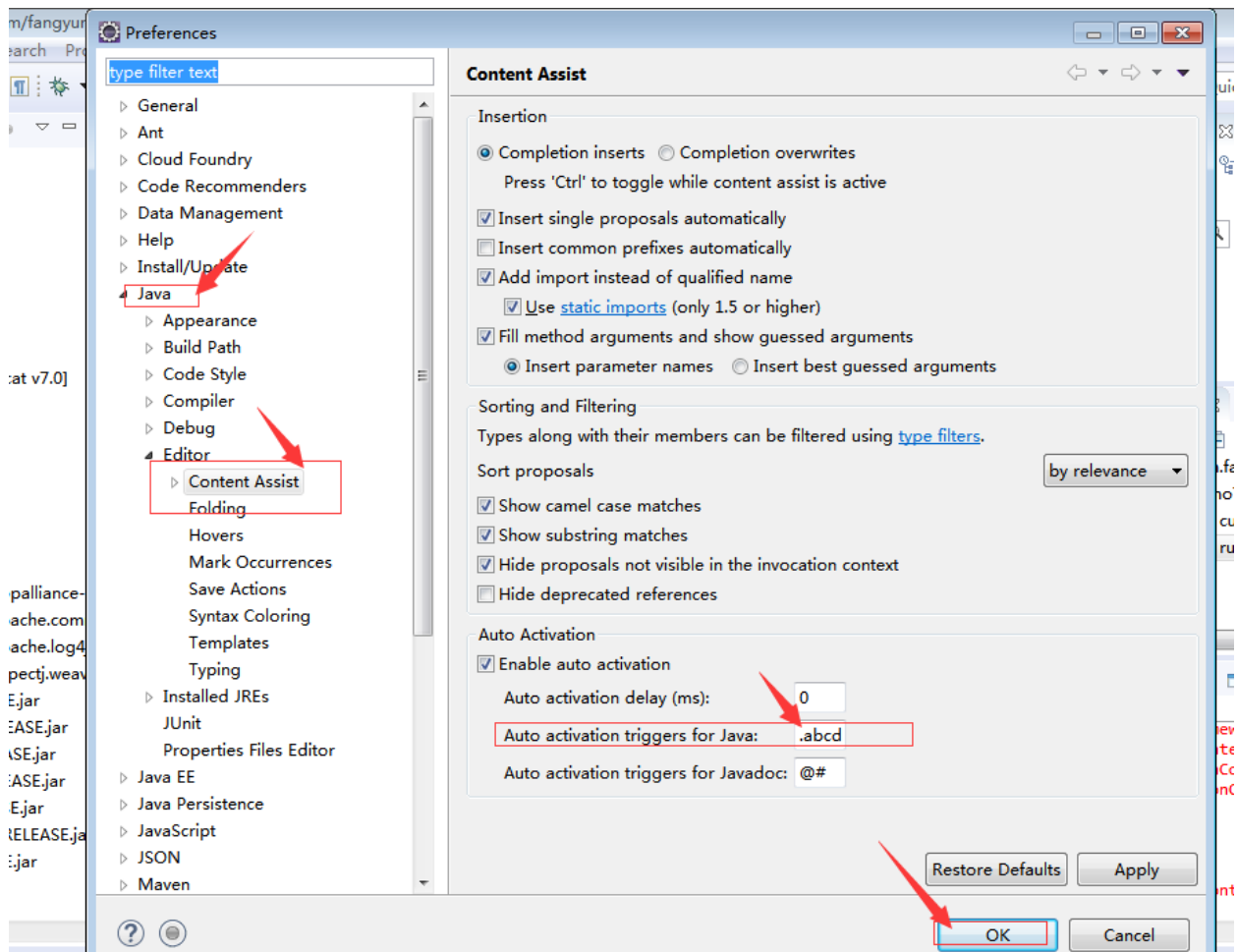
## eclipse开发工具环境设置

### 1. 全局工作空间设置



## 2. 编码快速设置

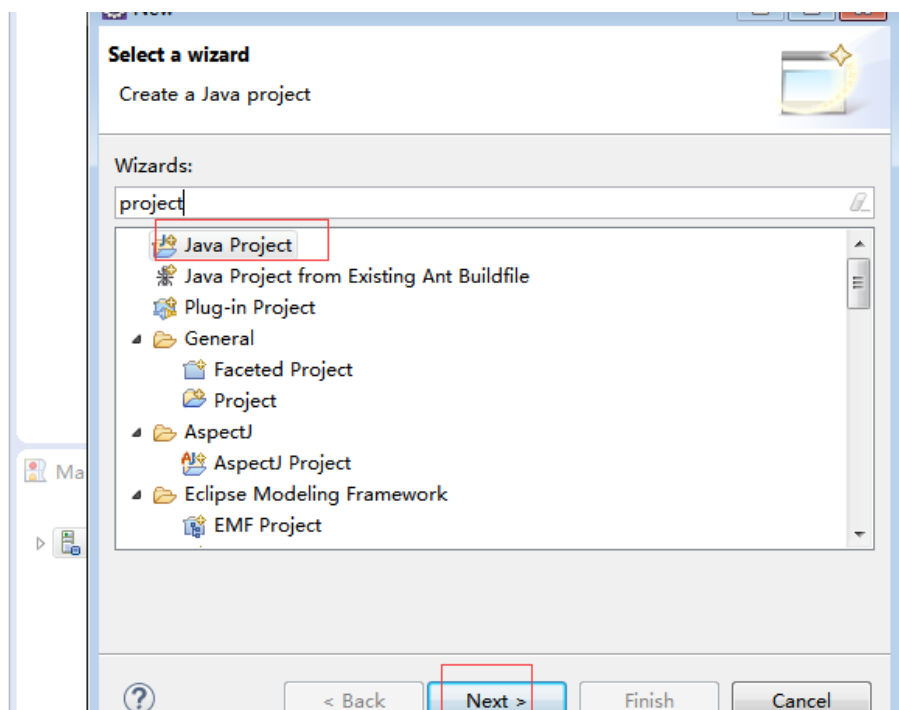




编写代码开发工具给的输入提示，26个英文字母，按顺序。

## 快速创建一个简单的java类

### 1. 创建java project工程



## Create a Java Project

Create a Java project in the workspace or in an external location.



Project name:

☒ Use default location

Location:

[Browse...](#)

### JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jre7') [Configure JREs...](#)

### Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

### Working sets

☐ Add project to working sets

Working sets: [Select...](#)

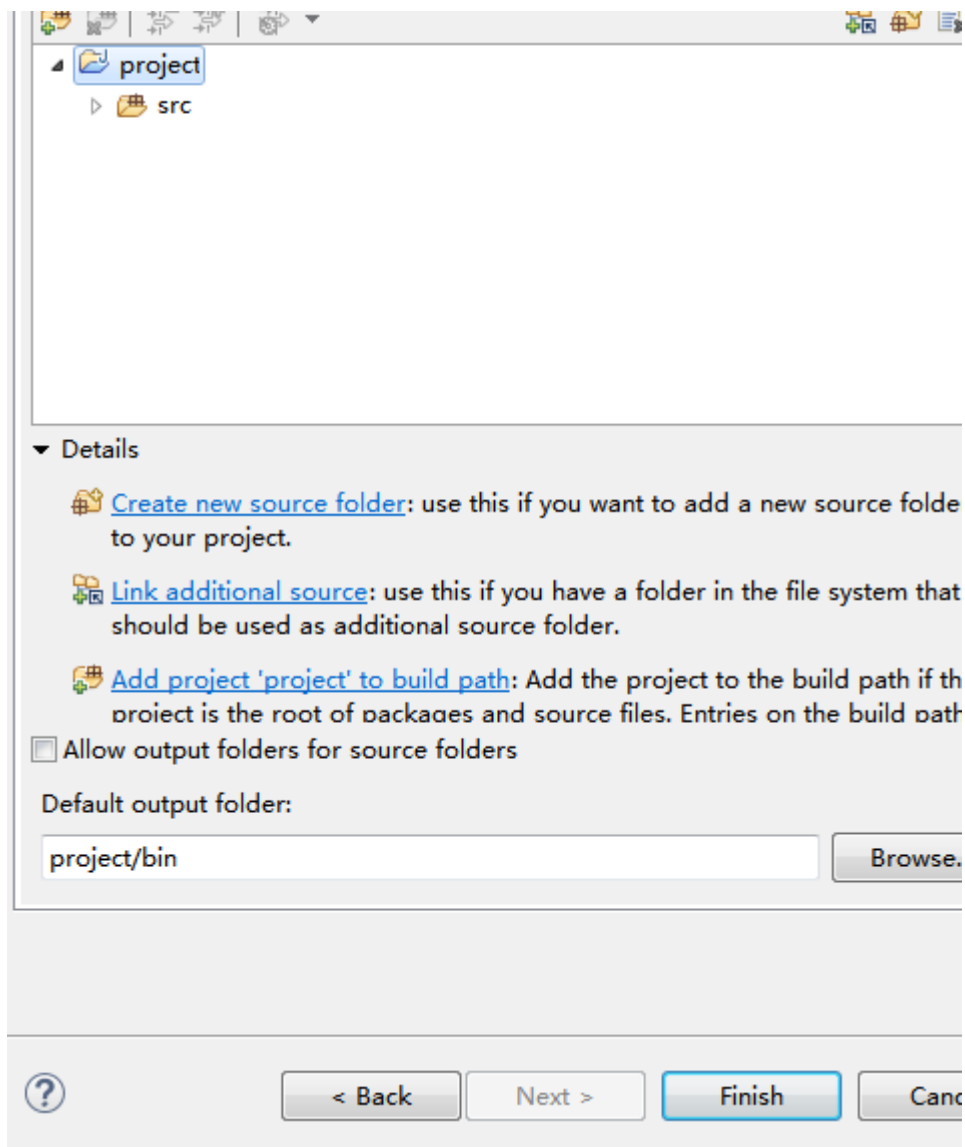


< Back

Next >

Finish

Cancel



## 2. 创建java类

创建寻找路径：src->new->class

Source folder:

Package:

☐ Enclosing type:

---

Name:  定义java名称

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

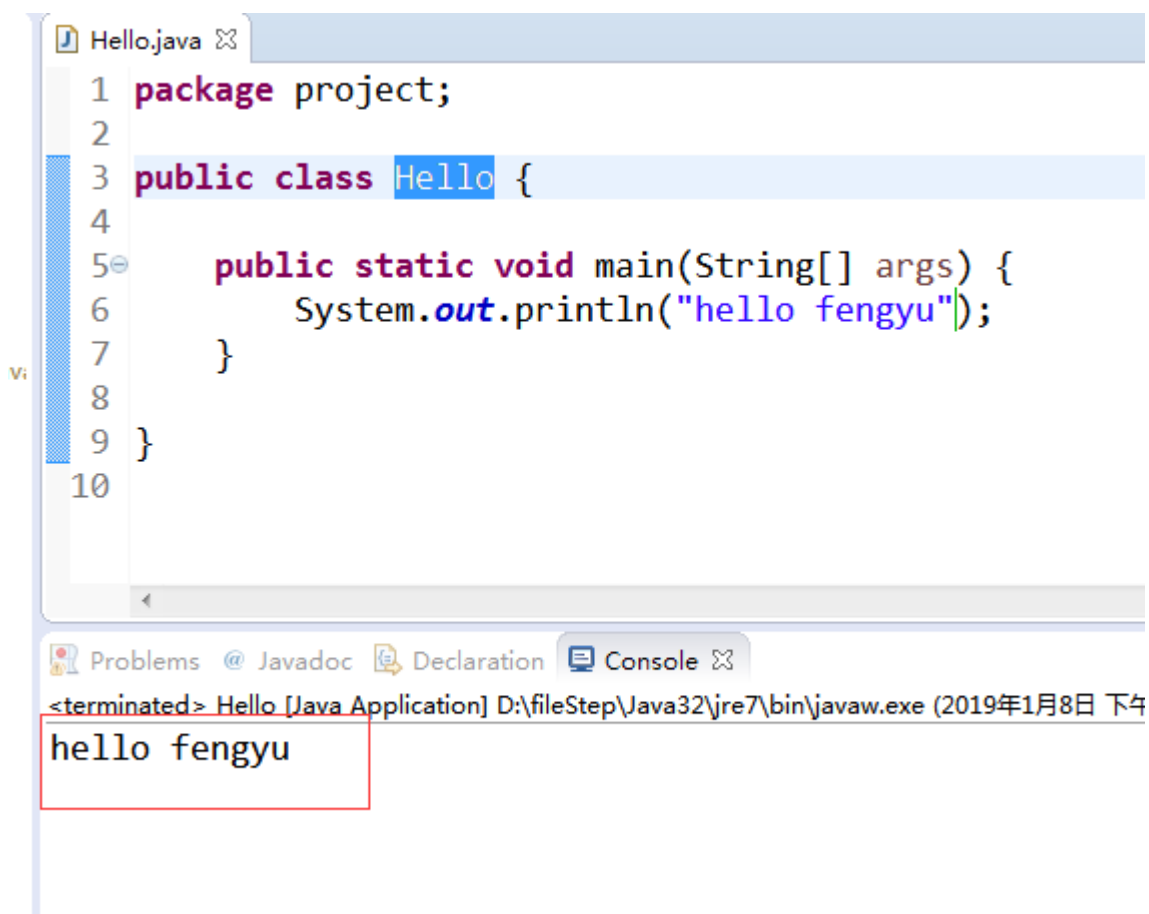
☒ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

### 3. 测试编码

Run as->java application



The screenshot shows the Eclipse IDE interface. The top editor window displays the code for `Hello.java`:

```
1 package project;
2
3 public class Hello {
4
5     public static void main(String[] args) {
6         System.out.println("hello fengyu");
7     }
8
9 }
10
```

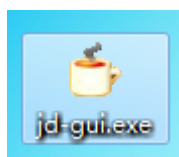
Below the editor, the **Console** tab is active, showing the output of the program:

```
<terminated> Hello [Java Application] D:\fileStep\Java32\jre7\bin\javaw.exe (2019年1月8日 下午)
hello fengyu
```

console控制台输出，如果没有看到控制台，点Window->show view->Console  
eclipse开发工具生成class类文件在项目下的bin文件夹下（查看即可）。

## class文件查看

可以使用jd-gui.exe查看



注：协助每个同学所有java开发工具的安装，idea,eclipse,myeclipse...,jdk等环境，前期先备有

注：无论如何，从最初一开始就要养成**所有编写**（命名，名称...那怕一个单词）**规范**（文档查看，所需要的规范说明的下一），否则过后学习越多，太多同学不按照规范来写，五花八门，到最后养成习惯而更改不过来。

## 程序（了解）

做某事或未来做某项目时，尽可能倾向于从整体上或宏观上去看待一件事情或一个事物，有时间的情况下研究它的起因，变化过程或趋势，然后尝试着去推测其未来走向。

有时细节决定成败，因此从某种意义上讲，需要平时多关注宏观，至少也应该和细节持平。因为“宏观”通常和整体结构对应，“细节”通常和局部处理进行对应。

当整体结构一旦确定下来，后期改起来会特别的麻烦（有时会牵一发而动全身），这是因为牵扯到的方面有时太广。但是局部处理因涉及范围较小，后期更换处理方法会相对变得容易一些。

因此无论是从实践（理论），实现细节是变化最易频繁。应该做的是把整体**结构设计良好**，具体某个地方的实现细节根据**实际情况**而定。

很多人总是会陷入去关注细节，让细节占据大部分思维，而忽视从宏观整体上的把握，或在此上面投入的精力不够，而可能导致后期整个项目会被Pass掉或是干掉（平时学习编写或是工作任务中想到，那些是主要的，那些是次要，分出主次）。

## 程序 = 数据结构 + 算法

国外很早提出的计算机专业一个公式：程序 = 数据结构 + 算法

国外程序员说过，数据结构和算法这两者中，**数据结构**要更重要一些，它的重要性是要大于算法的

如，这样一道题目，给一个单链表，逆向输出，拿到这个题目后，不管最终如何实现，至少要去**想一想**。

把这个题目改一下，给一个双向链表，也逆向输出一下。拿到这个题目后，根本就不用想，直接从尾部向前输出即可。

可看到，数据结构变了之后，实现方法一下子就简单了很多。所以数据结构的重要性是要大于算法的。数据结构决定算法。

如人们常说，条条道路通罗马，但有些人一出生就在罗马。就算你的排序算法再快，不可能比已经有序根本就不用排序的还快。这属于是极限思维运用。

数据结构指的是**数据存储方式**或描述方式，自己定义的接口啊、类啊这些都叫数据结构，并不只是List或Map这些。

算法就是指**解决问题的方法**，平常写的一些代码也可以称为算法，并不只是像排序算法、哈希算法（折半算法...）这些才是。

想一想要写的程序代码,最主要的就是**定义数据，获取数据，传递数据，操作数据，存储数据**。

定义数据就是类，获取数据就是查询数据库或从客户端提交，传递数据就是本地方法的参数或远程调用时数据的协议传输，操作数据就是各种运算/转换/排序等，存储数据就是类对象或容器对象或数据库等。

**定义数据**和**存储数据**就是数据结构（数据结构=定义数据+存储数据），**操作数据**就是算法，因此，程序 = 数据结构 + 算法。

如果数据结构经过**精心设计**，算法就会变得很简单，如再处理好数据的获取与传递，那最终写出来的程序，一定是非常好且有价值的代码。

## 软件（了解）

公式：

**软件 = 逻辑抽象 + 合理实现**

程序角度而言，软件的实现都是从逻辑抽象开始，无论是横向的分模块还是纵向的分层，或者说分子系统（项目大时，分而治之，就是一个大项目拆分成好多个小项目），只不过是不同的抽象方法运用而已。这个逻辑抽象是非常重要的，凡是存活时间长的软件，都是经过**良好**逻辑抽象的。

随着时间的推移，事物都会变化，良好的抽象能抵抗变化，或更能适应变化，因此活的时间就会更久一些。

逻辑抽象是一个很复杂的问题，涉及很多哲学思想或**权衡**问题。如，自动化程度高的软件，定制性不强，不容易满足用户的个性化需求。个人定制化强的软件，自动化程度则不高，又会造成用户难以上手，不易普及推广。

（后期学习mybatis回头看看这段）以前的Hibernate的消亡及后来Mybatis的兴起，就是一个定制化大于自动化的结果，Linux用于服务器操作系统，也需要专人维护。Windows用于日常办公系统，每个人都会用。平板电脑等不管几岁的娃娃都会很易上手使用，没有所谓好坏，只是定位的不同。

因此抽象是一个综合问题，充满哲学、权衡取舍。没有特别统一的标准，也没有严格意义对错。只有在使用编写过程中，看你更关注什么，或更期望什么，学会复盘，学会积累。

抽象完之后，一定要能合理实现才行（否则编写了逻辑抽象，却不使用，相当于也是白费的代码）。也不能为了抽象而抽象，最后无法实现，一切不能落地的实用东西，都是空谈。如抽象出一个大脑与电脑进行一个意识交流使用的接口，可就无法落地实现。

理想设计过程：

- 1.合理抽象，划分好子系统/模块，定义好功能边界、交互方式，让项目的整体结构清晰
- 2.精心设计数据结构，定义好类或接口，使代码写起来变得简单，后期容易修改及维护
- 3.其次多考虑把握好宏观的整体，又多关注于具体的实现细节

则称之为最优的理想设计过程

前面属于所有人想要的理想状态，但可能项目开发实际过程不是如此（客户需求，人性最不可以测试），实际工作场景大部份是会变化的，如下的场景：

第一天工作上班：

BOSS：来来来，有个需求给你说

小李：好的

第二天工作上班：

BOSS：昨天的那种方式不好，按这种方式实现吧

小李：好的

第三天工作上班：

BOSS：昨天的那种方式好像还有点问题，按这种新的方式实现吧

小李：好的

第四天工作上班：

BOSS：昨天的那种方式好是好，可能别人一时不太好接受，要不还是按最开始的方式实现吧

小李：好的

第五天工作上班：

BOSS：多长时间能做好

小李：投入5个人，大概2个月吧

BOSS：我给你20个人，半个月能弄好吧

小李：这...

因此，实际软件开发中，客户提出的需求不可能不会变动，一切以实际需求进行开发项目，前期时，框架设计是否合理，需求是否多变，人员的技术能力等（如同一条船，只有成或败）