# 第一章 认识Java语言

## 1. 认识Java语言了解（了解）

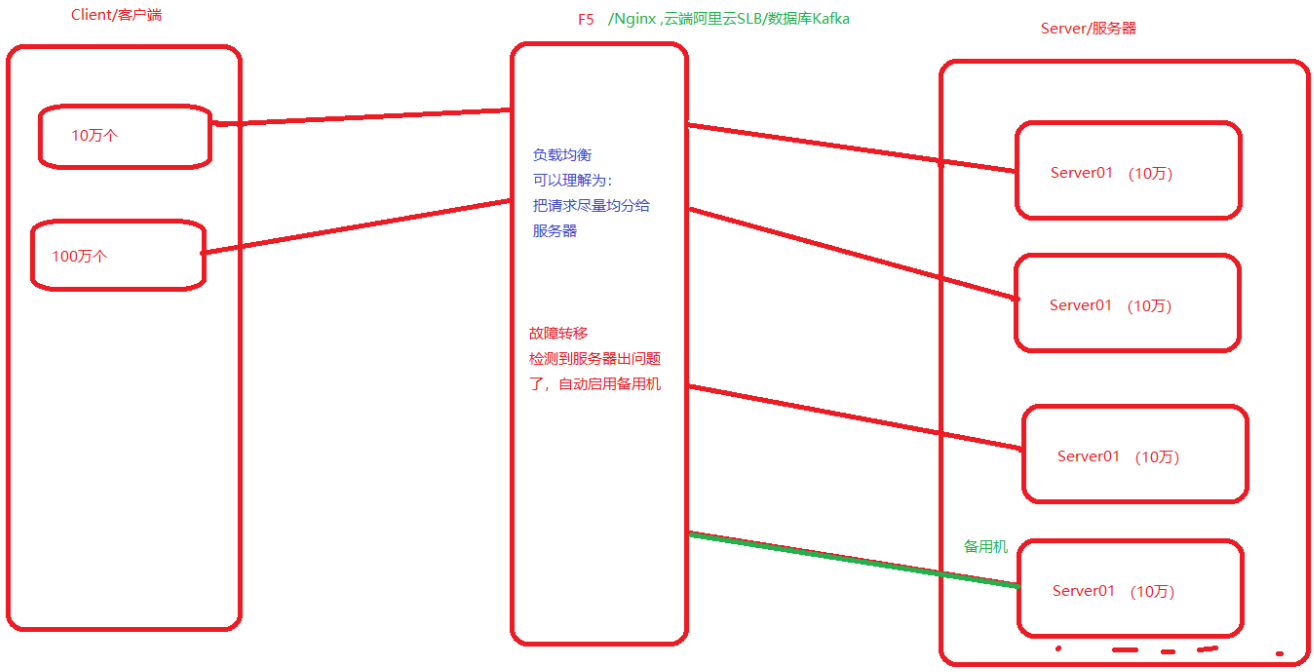
### 1.1 Java编程语言的主要特征

1. 简单性
    a. 语法简单
    b. 针对复杂的操作做了一些封装，比如：内存管理和垃圾回收，都是封装好的
    c. 舍弃了C语言的指针和多继承
    d. ...
2. 面向对象
    a. 结构比较清晰，适合做大型项目
    b. 面向对象和面向过程

```
面向过程的公司架构:
   老板 = 财务 + 销售 + 售后 + 保洁 + 。。。（光杆司令）
面向对象的公司架构:
   老板（负责管理）
   财务-找人来做
   销售-找人来做
   售后-找人来做
   保洁-找人来做
   。。。

```

3. 分布式




4. 多线程
    a. 进程：就是运行在内存中的一个软件：比如：360，百度网盘
    b. 线程：软件的执行路径，比如百度网盘：一边下载，一边上传，这就是多线程
5. 安全
    a. 字节码文件传输：使用的非对称加密机制，不容易被篡改
    b. 四种安全机制：字节码校验器，类加载器，内存空间管理和访问权限
6. 丰富的API
    a. API（Application Programming Interface）：应用程序接口
    b. 银行转账开发：调用的是银行的接口（银行程序员封装好的，我们拿来直接使用）
    c. 很多复杂的应用，Oracle公司程序员都写好了，我们可以直接拿来用
7. 跨平台移植（Write once, Run Anywhere）


## 1.2 Java语言的三大平台（了解）

1. JavaSE（Java Standard Edition）:Java平台标准版
    - C/S架构
    a. C/S：Client/Server :客户端/服务器
        i. QQ下载后安装的桌面应用程序，找个QQ就是客户端
    b. B/S：Browser/Server：浏览器/服务器
        i. 比如：京东，百度，淘宝。。。
2. JavaEE(Java Enterprise Edition): Java平台企业版
    a. 适合开发，大中小型企业应用
3. JavaME（Java Micro Edition）:Java微型版
    - 手机端开发，现在几乎不用

## 1.3 Java语言开发的应用程序

1. 操作系统（Solaris）
2. 开发工具（NetBeans）
3. 数据库（Derby）
4. 浏览器（Hotjava）
5. 服务器（GlassFish）【Tomcat，Jboss，WebLogic(Oracle),WebSphere(IBM)】
6. 电商平台（淘宝，天猫，京东）
7. 企业级应用（EPR-NC,PMS-SG186）
8. ...

## 1.4 Java语言的专有名称

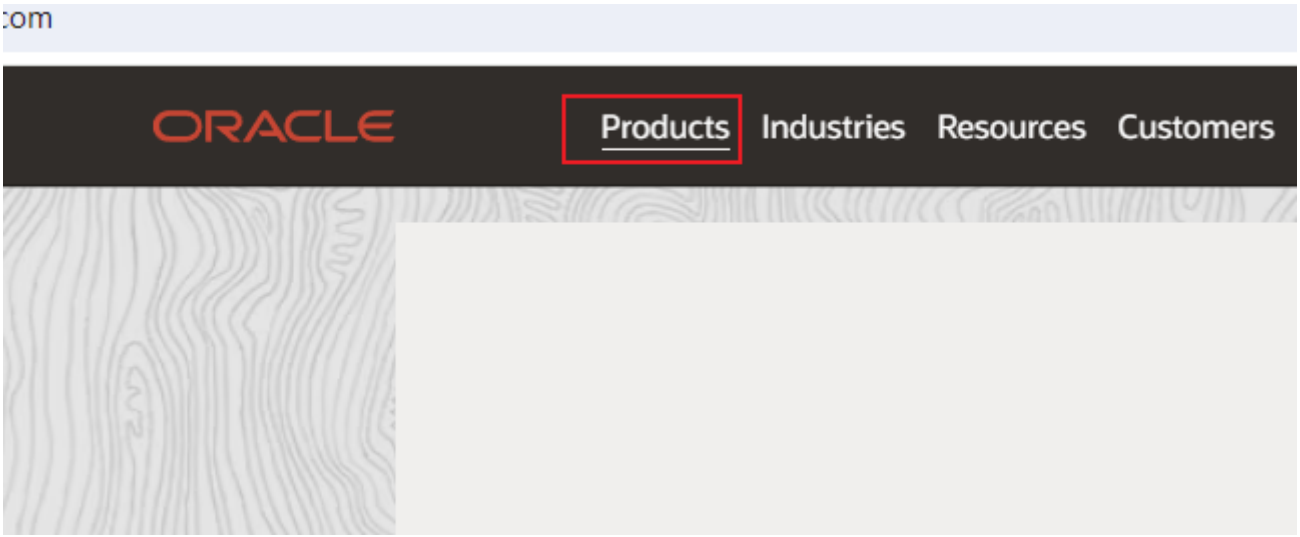
1. JDK（Java Development Kit）:Java开发工具包
2. JRE （Java Runtime Environment）:Java运行时环境
3. JVM（Java Virtual Machine）:Java虚拟机
4. SDK（Software Development Kit）： 软件开发工具包

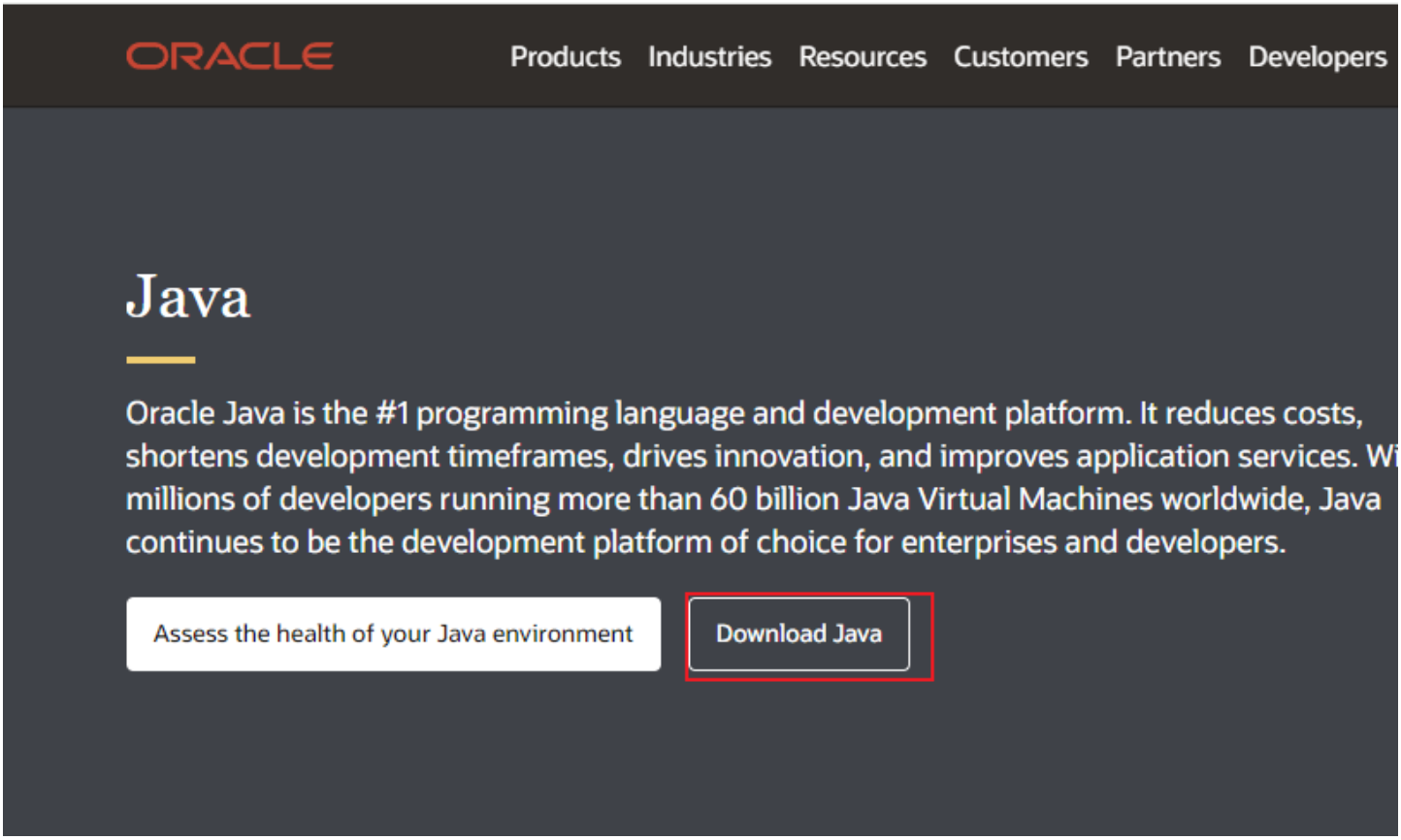
## 1.5 JDK,JRE和JVM之间的关系（掌握）

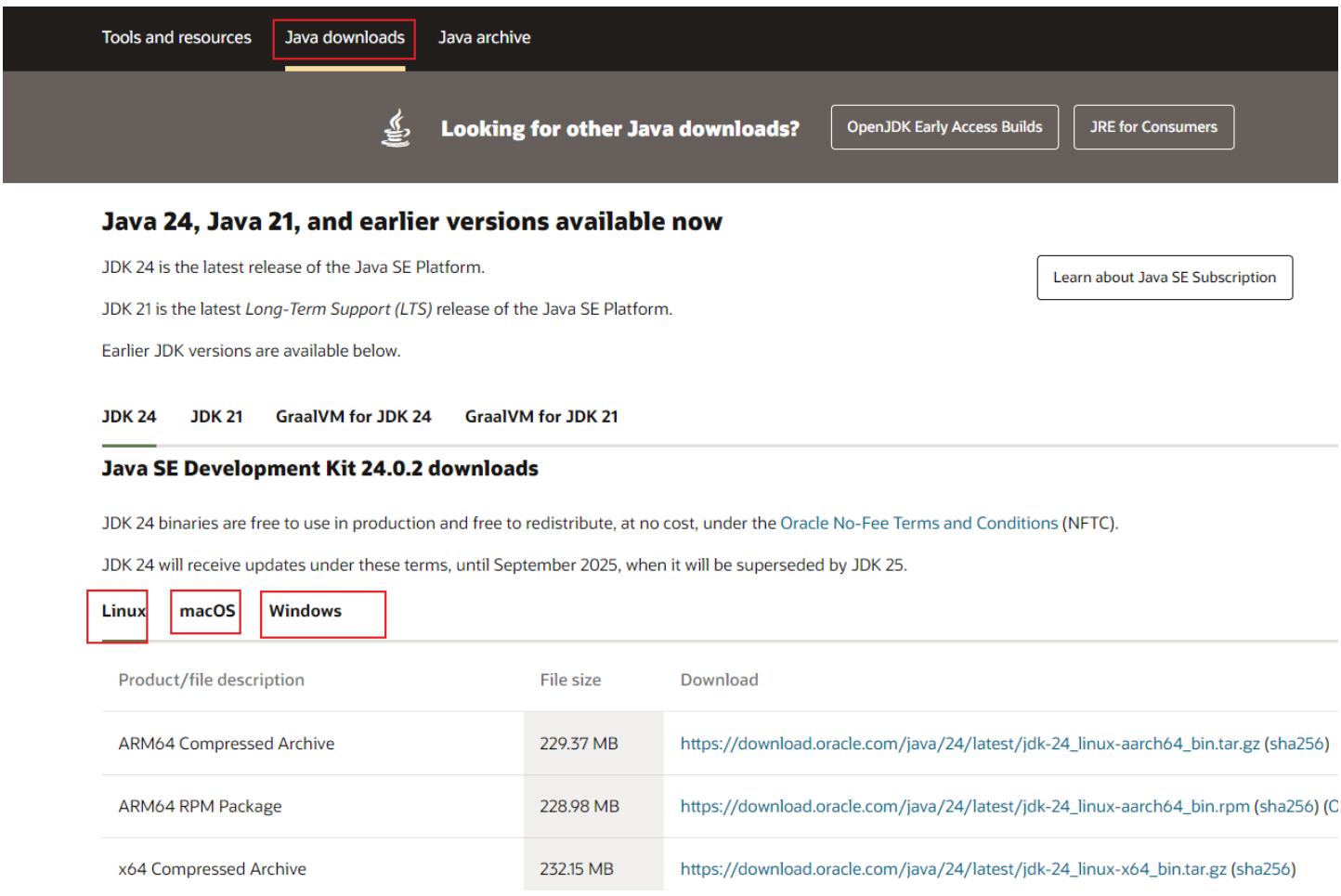
1. JRE=JVM + JavaSE核心类库（Oracle公司程序员写好的功能或者成为写好的代码）
2. JDK=JRE + Java开发工具集（javac.exe,java.exe,javap.exe,javadoc.exe...） + 完整的JavaSE类库
3. 想运行Java程序：装JRE
4. 想开发Java程序：装JDK

## 1.6 JDK的下载与安装（掌握）

1. 下载地址：www.oracle.com



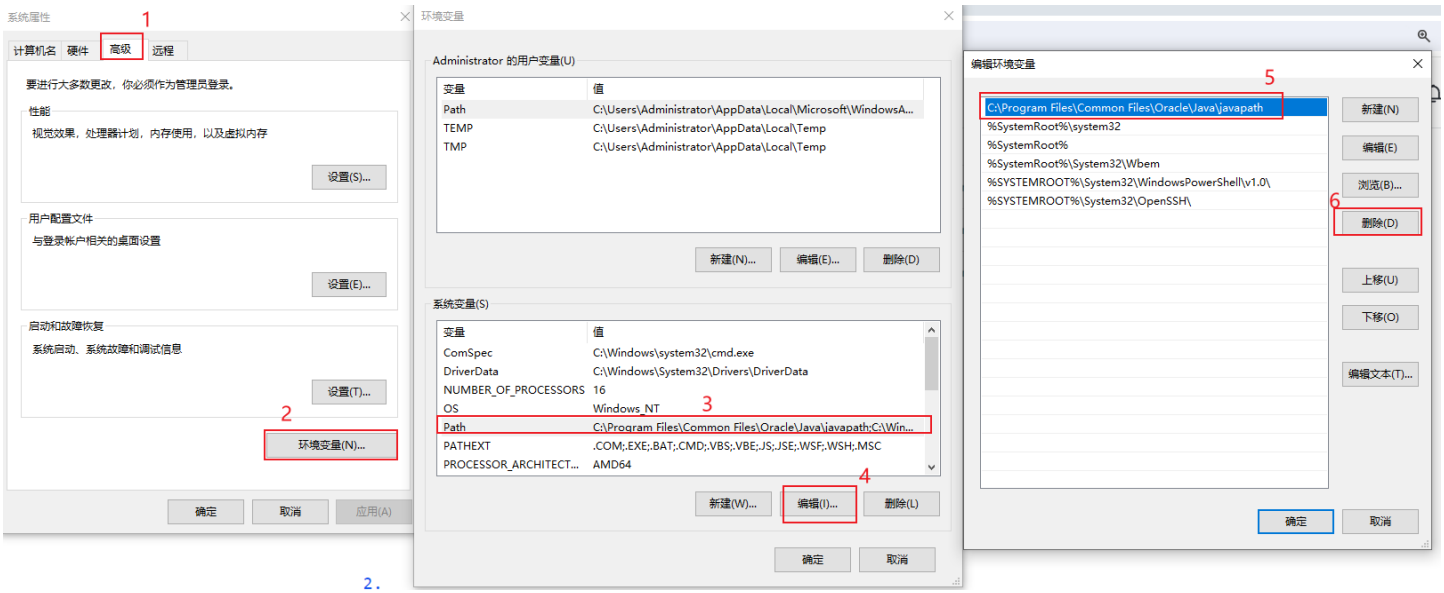




2. 我们需要装的版本JDK21

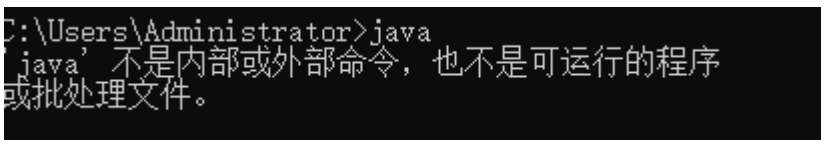
# 1.7 配置环境变量

## 1.7.1 删除默认的配置

1. 我的电脑--右键--属性--高级系统设置

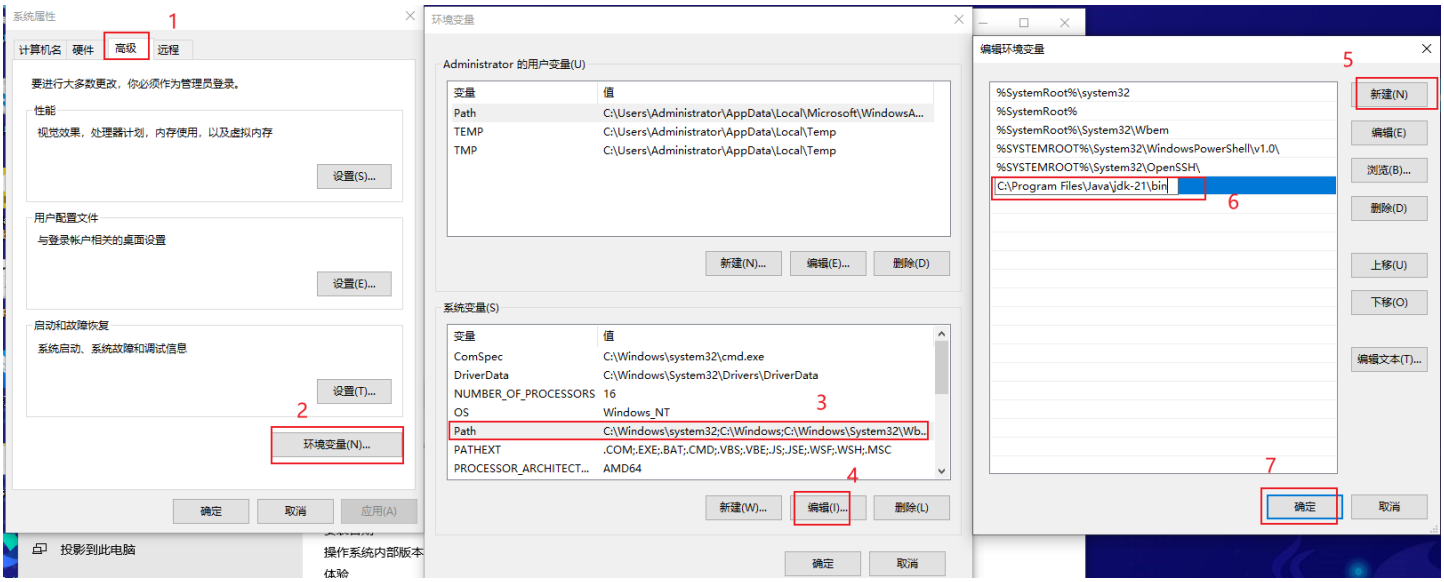



2. cmd命令的寻址原理
    a. 当前路径下寻址java.exe可执行文件
        i. 找到了--去执行
        ii. 没找到
            1. 去环境变量找
                a. 找到了--去执行
                b. 没找到




## 1.7.2 环境变量配置

1. **找到JDK的安装路径**：C:\Program Files\Java\jdk-21\bin
2. **把路径配置到环境变量中**：




注意：配置环境变量后，一定好重启DOS窗口

# 1.8 配置QQ

# 1.9 开发一个简单的应用程序（掌握）

## 1.9.1 编写阶段

1. 创建一个.java的文件
2. 编写一个类

代码块

```
class 类名称{
    类体;
}

class Hello{

}
```

- class 是定义类的关键字，必须这么写（大小写区分）
- Hello是类的名称，可以自定义（遵循命名规范，后面详细讲解，现在使用英文，单词首字母大写）
- {}，包含的部分chegnw

3. 在类中编写main方法

a. 程序运行的时候，先执行main方法，不写main方法会报如下错误：

代码块

```
D:\powernode\02-JavaSE\03-code>java Hello
错误: 在类 Hello 中找不到 main 方法, 请将 main 方法定义为:
   public static void main(String[] args)
否则 JavaFX 应用程序类必须扩展javafx.application.Application
```

b. 编写main方法

代码块

```
class Hello{
    public static void main(String[] args){
        方法体;
    }
}
```

i. main方法不可以随便写，要写在类体中

ii. main方法{}包含的部分是方法体

4. 编写输出语句

代码块

```
class Hello{
    public static void main(String[] args){
        System.out.println("我会输出");
    }
}
```

a. 输出语句要写在方法体中

b. 输出语句（） 中能写什么

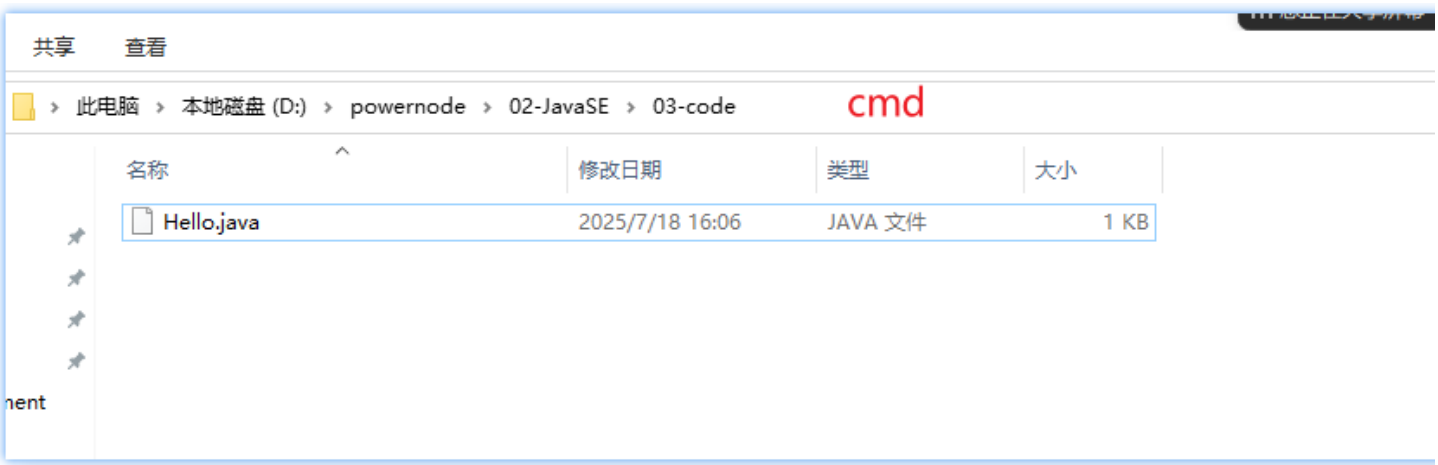
i. 数字，不用写双引号

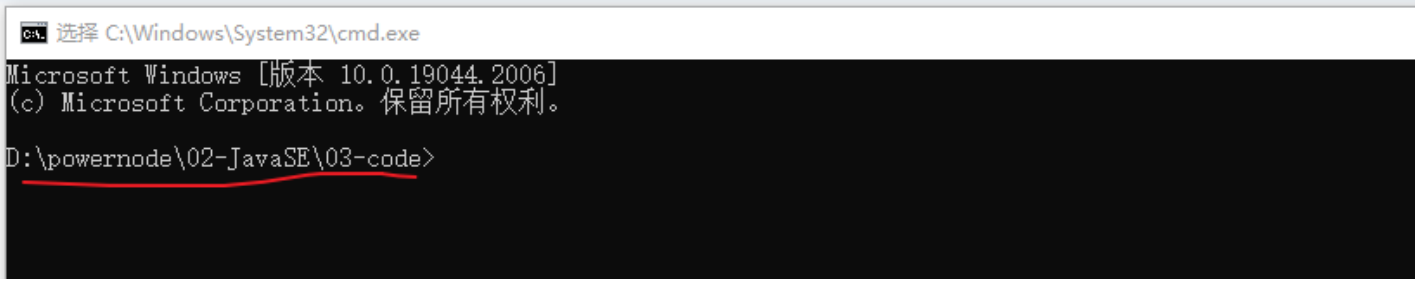
ii. 英文，中文和特殊符号，必须要添加双引号

c. **注意:标点符号都是英文输入法下的**

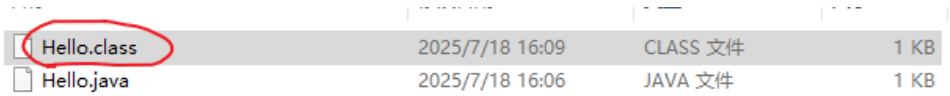
### 1.9.2 编译阶段

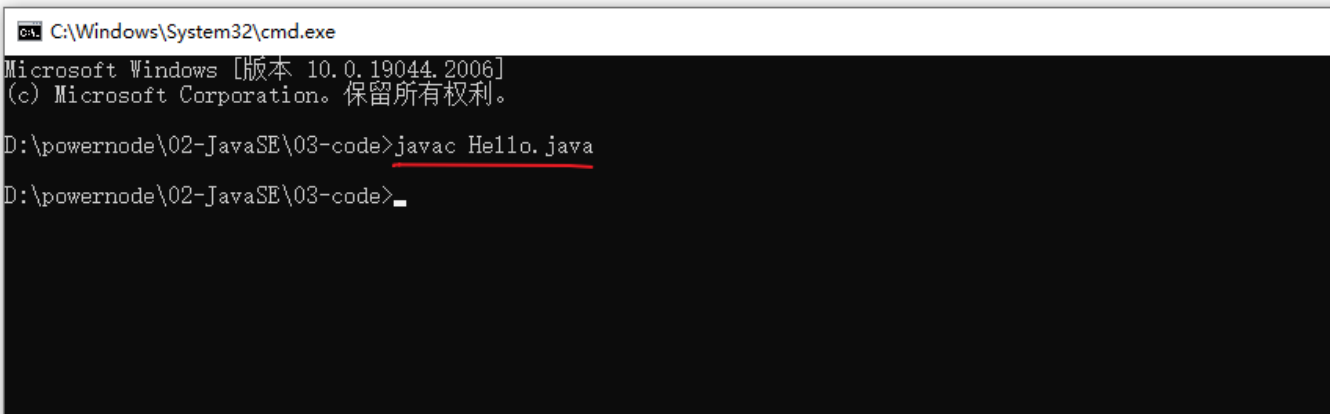
1. 找到.java的文件路径，cmd打开命令窗口




2. 编译.java文件
    a. 语法：javac 文件名称.java
    b. 举例：




### 1.9.3 运行阶段

1. 语法：java 类名称(.class文件的名称)
2. 举例：java Hello

## 1.10 总结编写步骤

1. 编写阶段
    a. 创建一个.java文件

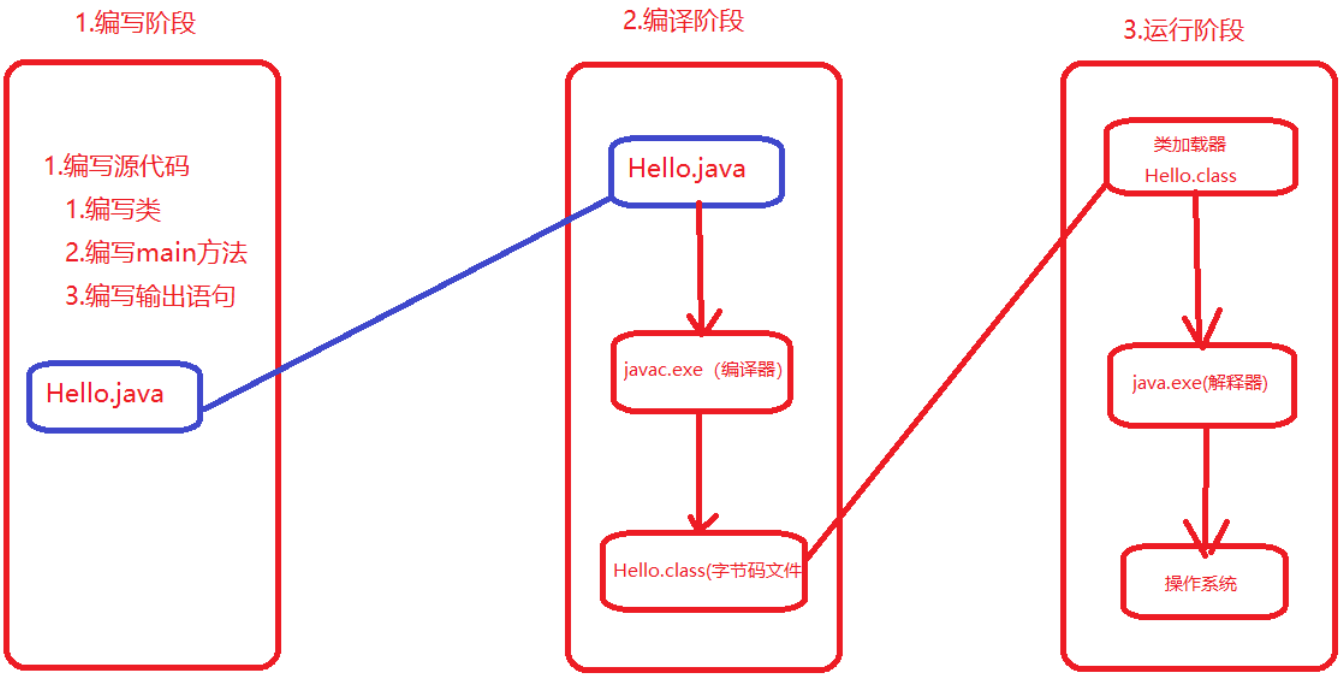
   b. 编写一个类
   c. 在类中编写main方法
   d. 在main方法中编写输出语句
2. 编译阶段
   a. 把.java文件编译为.class文件
   b. 语法：javac 文件名称.java
3. 运行阶段
   a. 语法：java 类名称 （运行是.class文件）

## 作业

1. 理解环境变量，配置QQ，cmd命令下启动
2. 编写一个Triangle类，打印输出三角形如下：

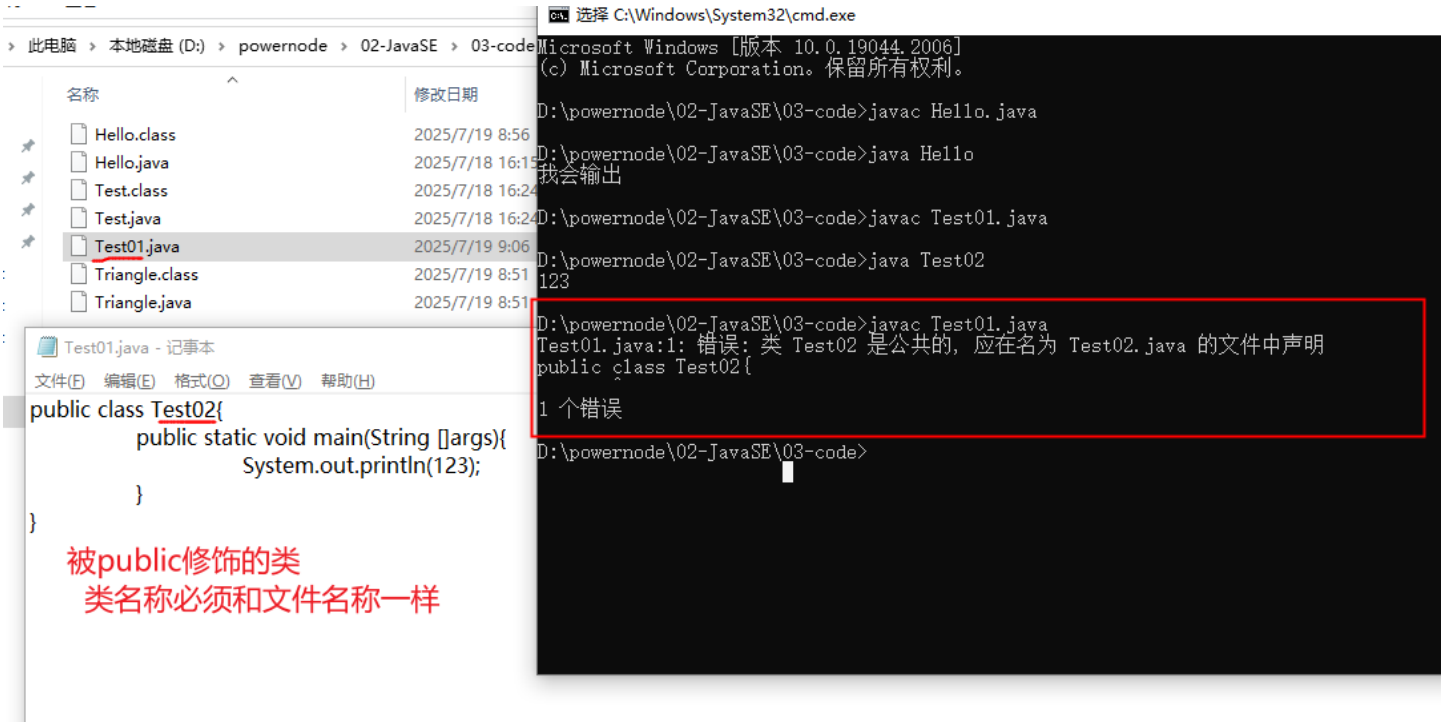
```
*
**
***
****
*****
******
*******
```

# 1.11 Java程序的执行原理




## 1.12 public 修饰符

- 用法一
- 使用**public修饰的类，类名称和文件名称必须一样**，否则报如下错误：




## 1.13 Java简单的DOS命令（了解）

1. 切换盘符
   a. 语法：盘符 + 冒号（英文输入法）
   b. 举例：d:

2. 列出当前盘符下的文件
    - dir
3. 进入文件夹
    a. 语法：cd 空格 文件名称
    b. 举例：cd  02-JavaSE
4. 返回上一级目录
    - cd .. (注意..前面添加空格)
5. 返回根目录
    - cd /(注意/前面添加空格)

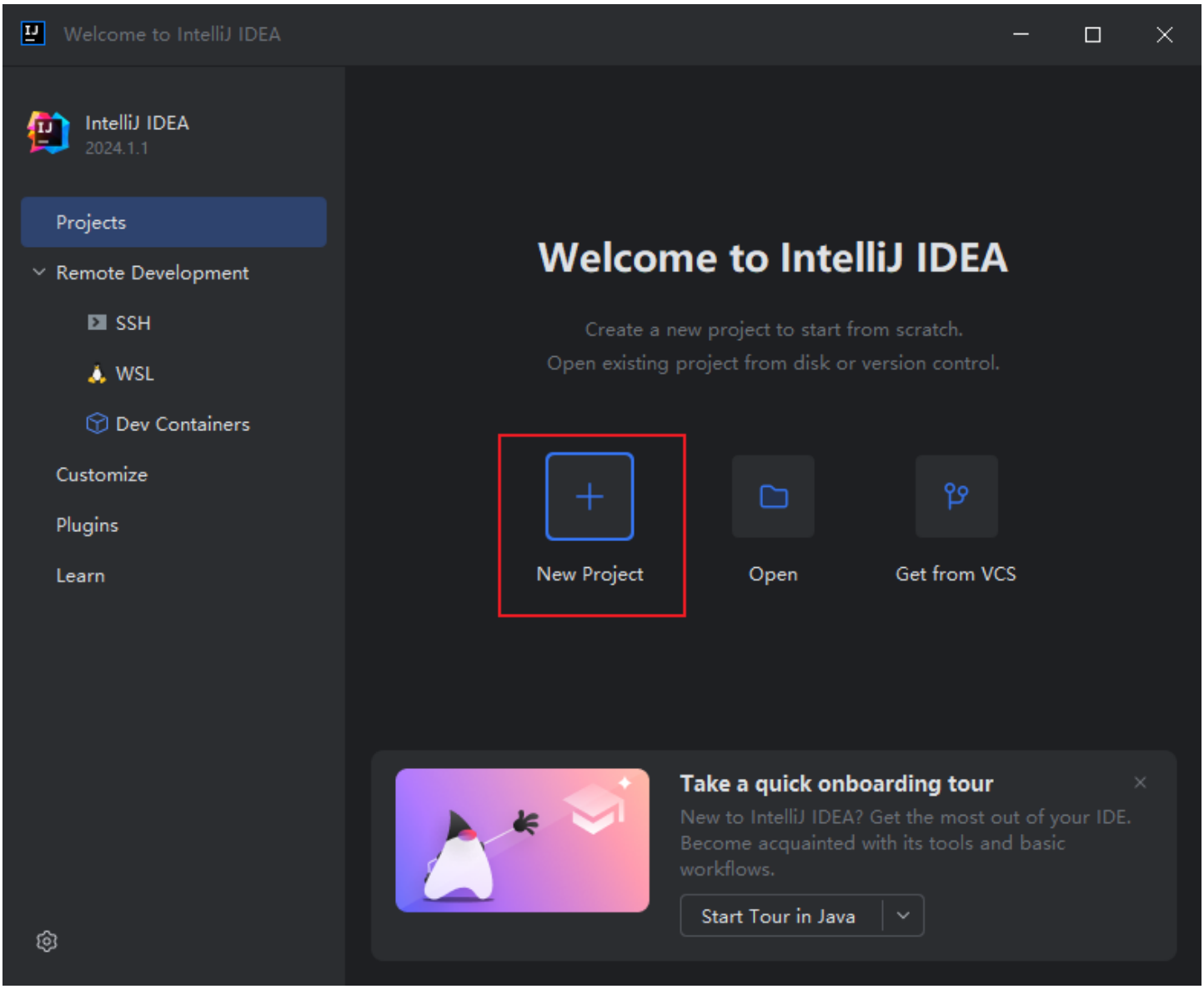
### 1.14 文件夹和文件的管理（了解）

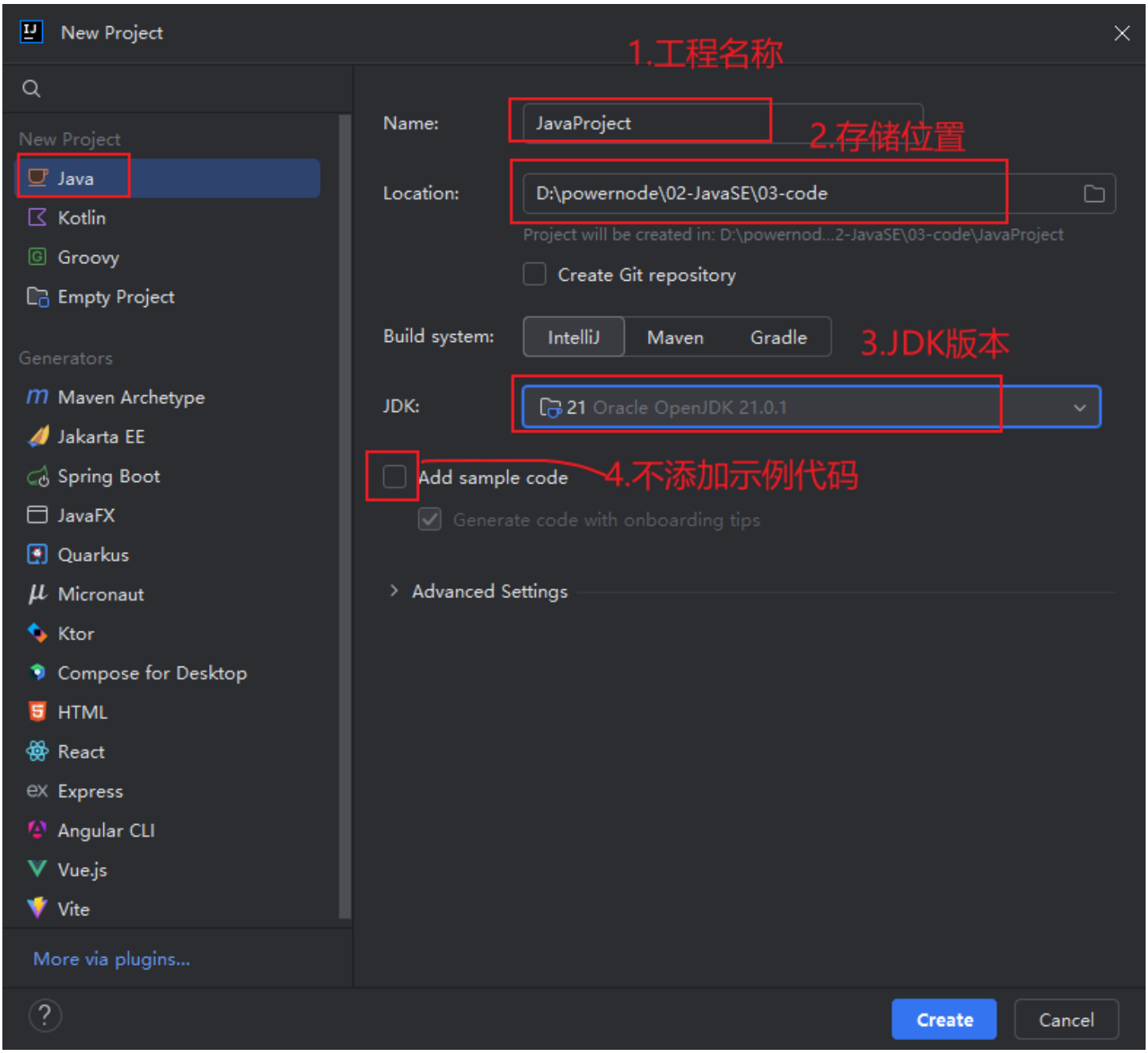
1. 文件夹管理
    a. 创建文件夹：md test
    b. 打开文件夹：start  test
    c. 删除文件夹：rd test
2. 文件管理
    a. 创建文件：echo >hello.java
    b. 创建文件并写入内容：echo abc123 > Hello11.java
    c. 显示文件内容：type Hello11.java
    d. 删除文件：del Hello11.java
    e. 删除后缀为.class的文件 ： del *.class

## 2. 安装 idea

### 2.1 安装idea

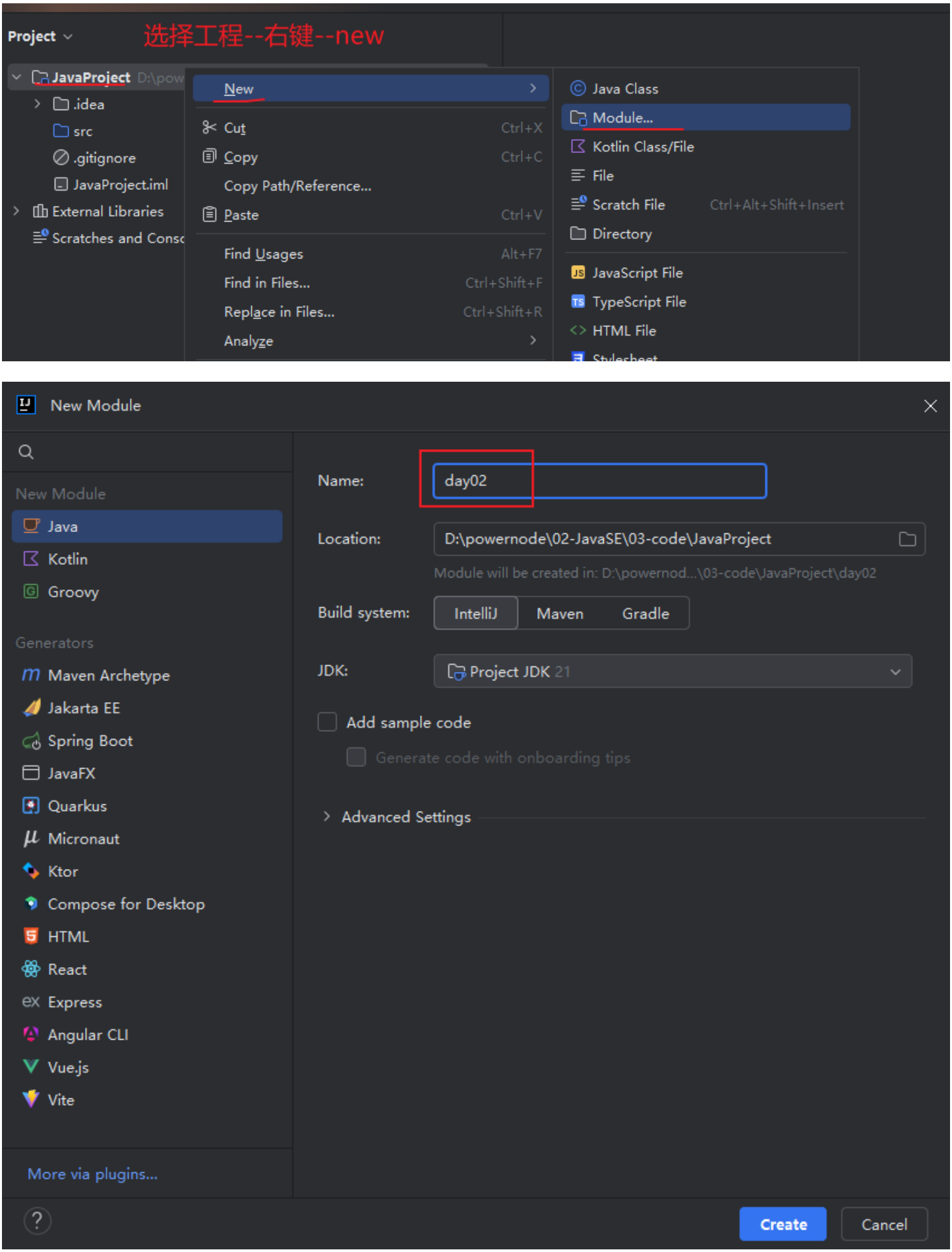
### 2.2 创建工程


Welcome to IntelliJ IDEA
IntelliJ IDEA
2024.1.1
Projects
Remote Development
SSH
WSL
Dev Containers
Customize
Plugins
Learn
Welcome to IntelliJ IDEA
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
New to IntelliJ IDEA? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour in Java

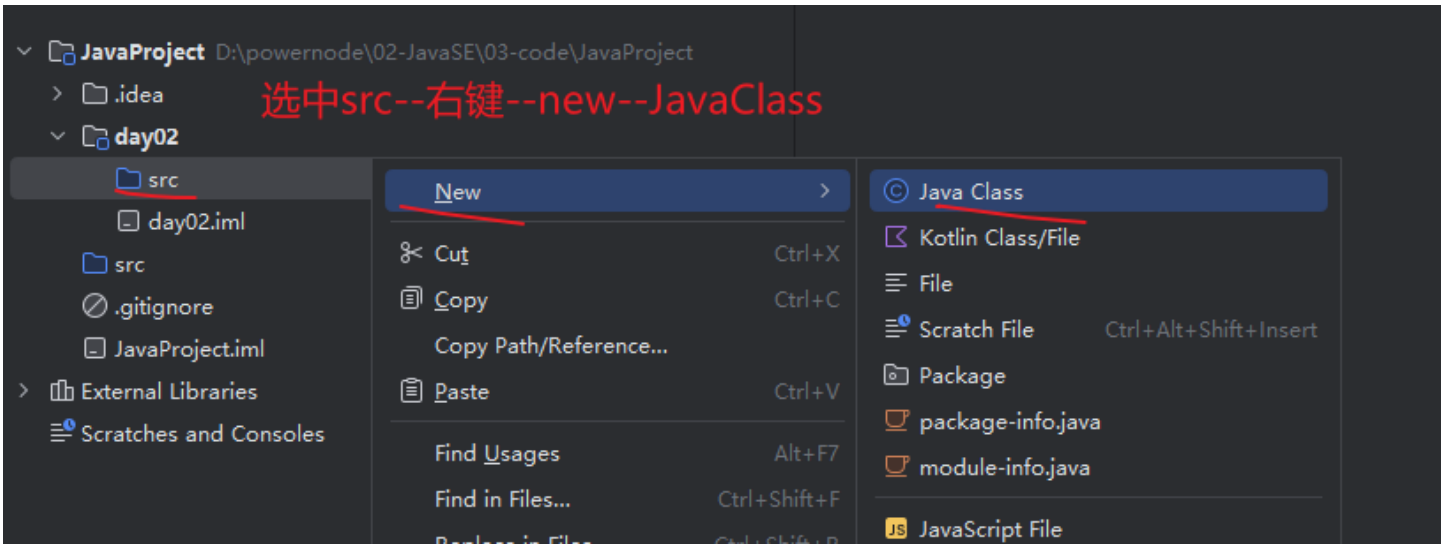



## 2.3 新建模块和类

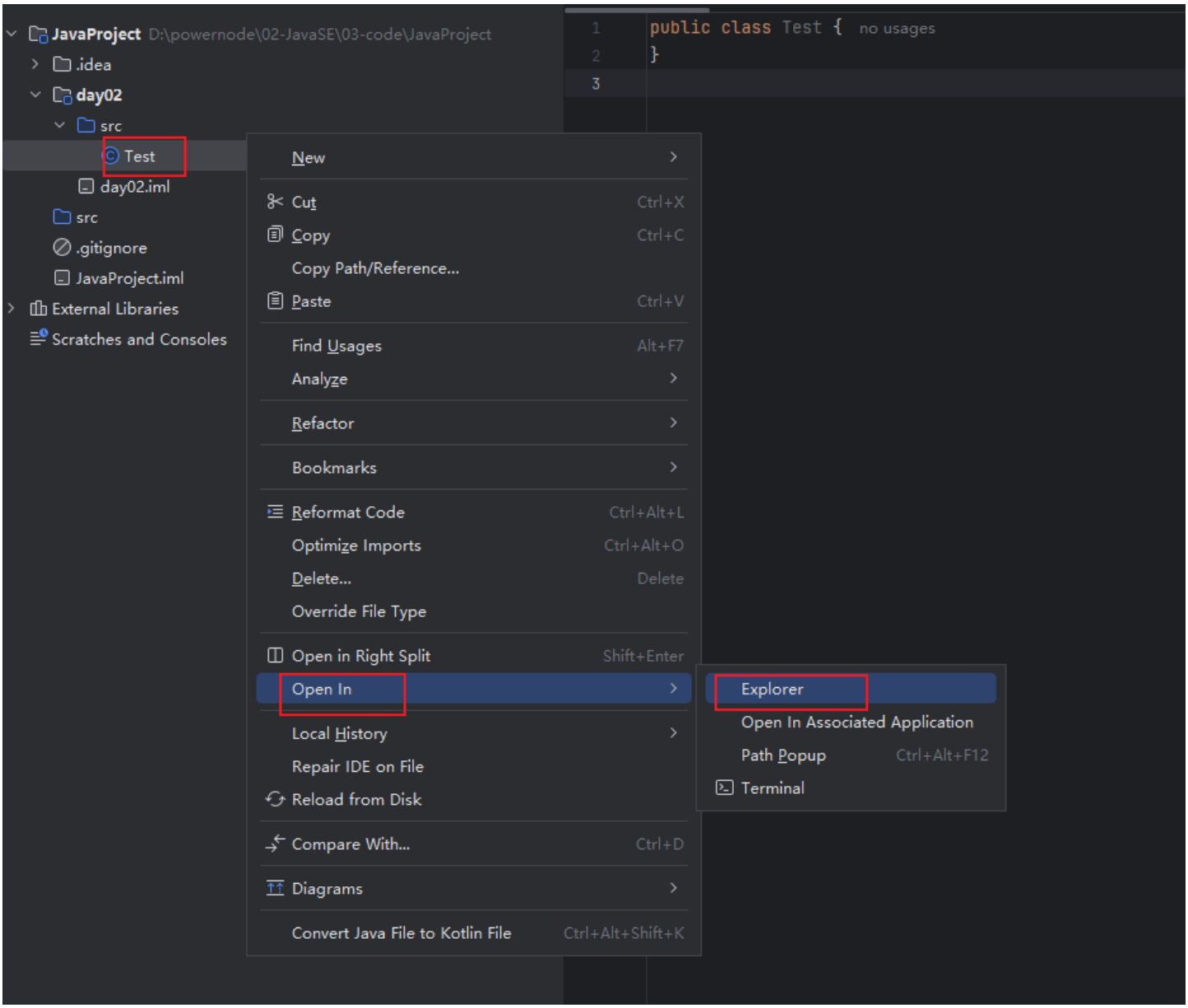
### 2.3.1 新建模块


Project
选择工程--右键--new
JavaProject D:\pow
.idea
src
.gitignore
JavaProject.iml
External Libraries
Scratches and Consc
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Java Class
Module...
Kotlin Class/File
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
JavaScript File
TypeScript File
HTML File
New Module
New Module
Java
Kotlin
Groovy
Generators
Maven Archetype
Jakarta EE
Spring Boot
JavaFX
Quarkus
Micronaut
Ktor
Compose for Desktop
HTML
React
Express
Angular CLI
Vue.js
Vite
More via plugins...
Name:
day02
Location:
D:\powernode\02-JavaSE\03-code\JavaProject
Module will be created in: D:\powernod...\03-code\JavaProject\day02
Build system:
IntelliJ
Maven
Gradle
JDK:
Project JDK 21
Add sample code
Generate code with onboarding tips
Advanced Settings
Create
Cancel

## 2.3.2 创建类

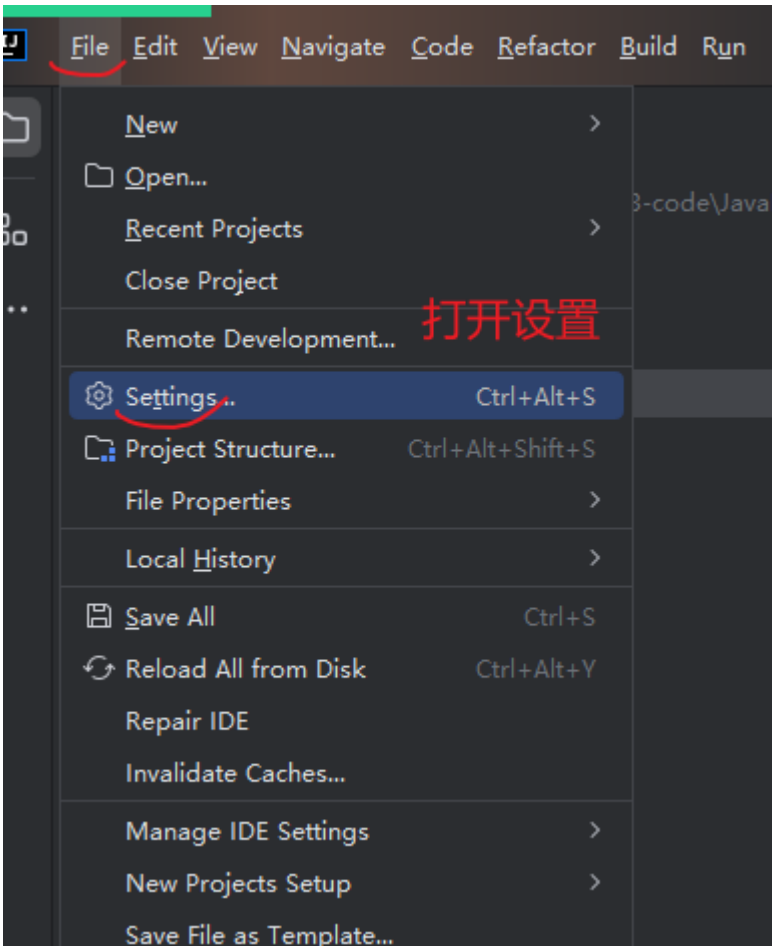


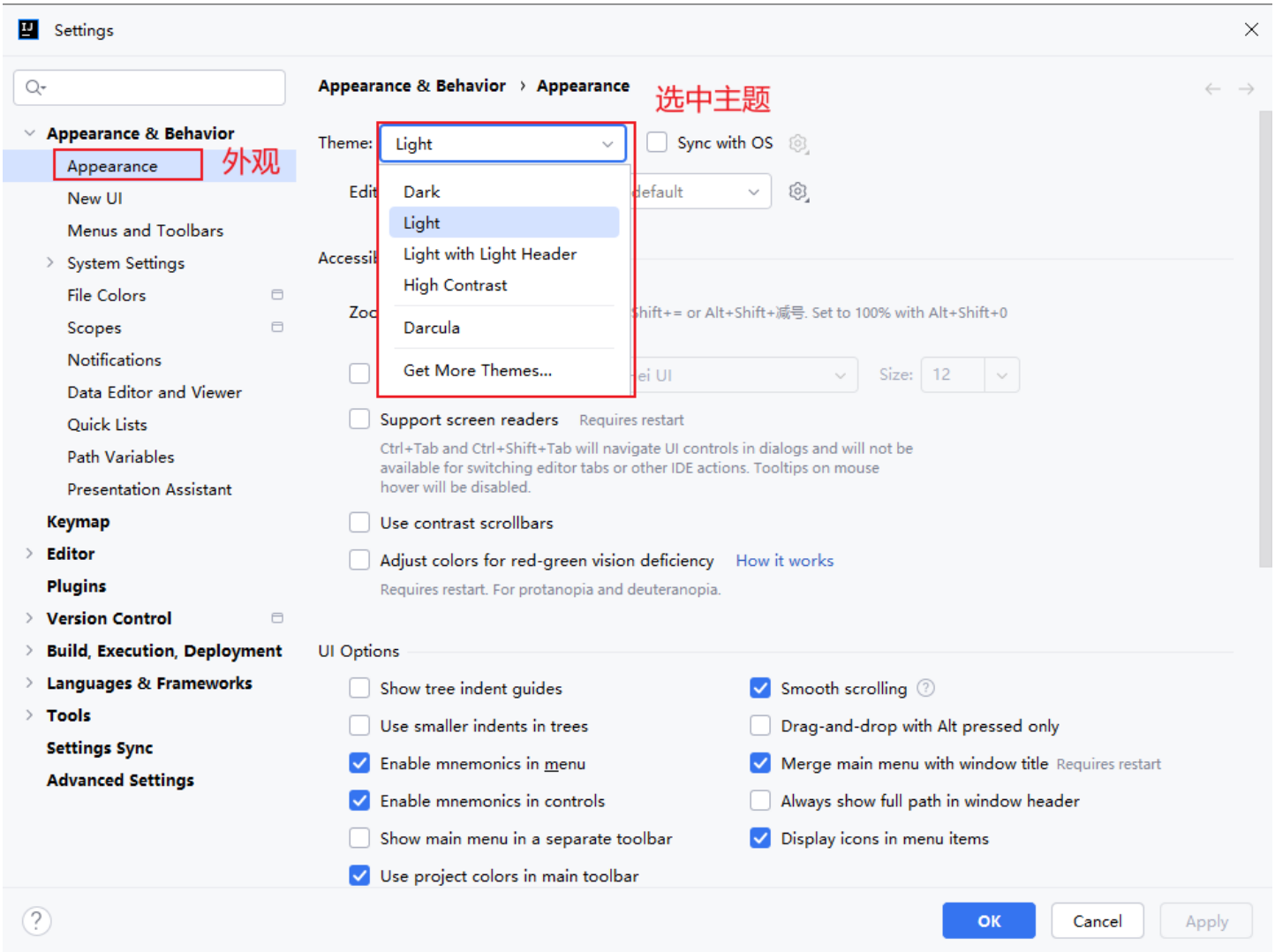

## 2.3.3 创建类的存储位置




## 2.4 idea设置

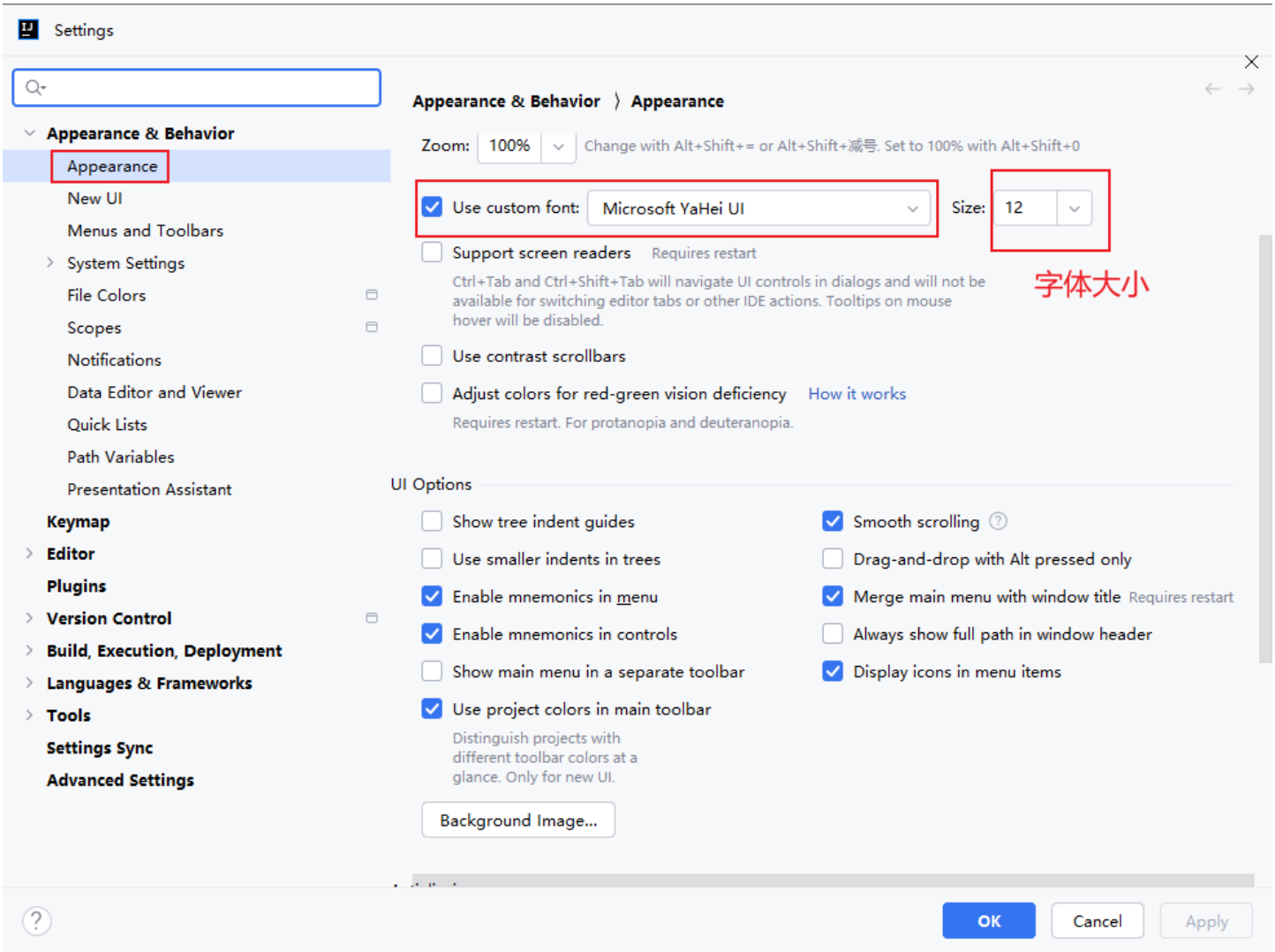
### 2.4.1 设置主题


File Edit View Navigate Code Refactor Build Run
New
Open...
Recent Projects
Close Project
Remote Development...
打开设置
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Save File as Template...

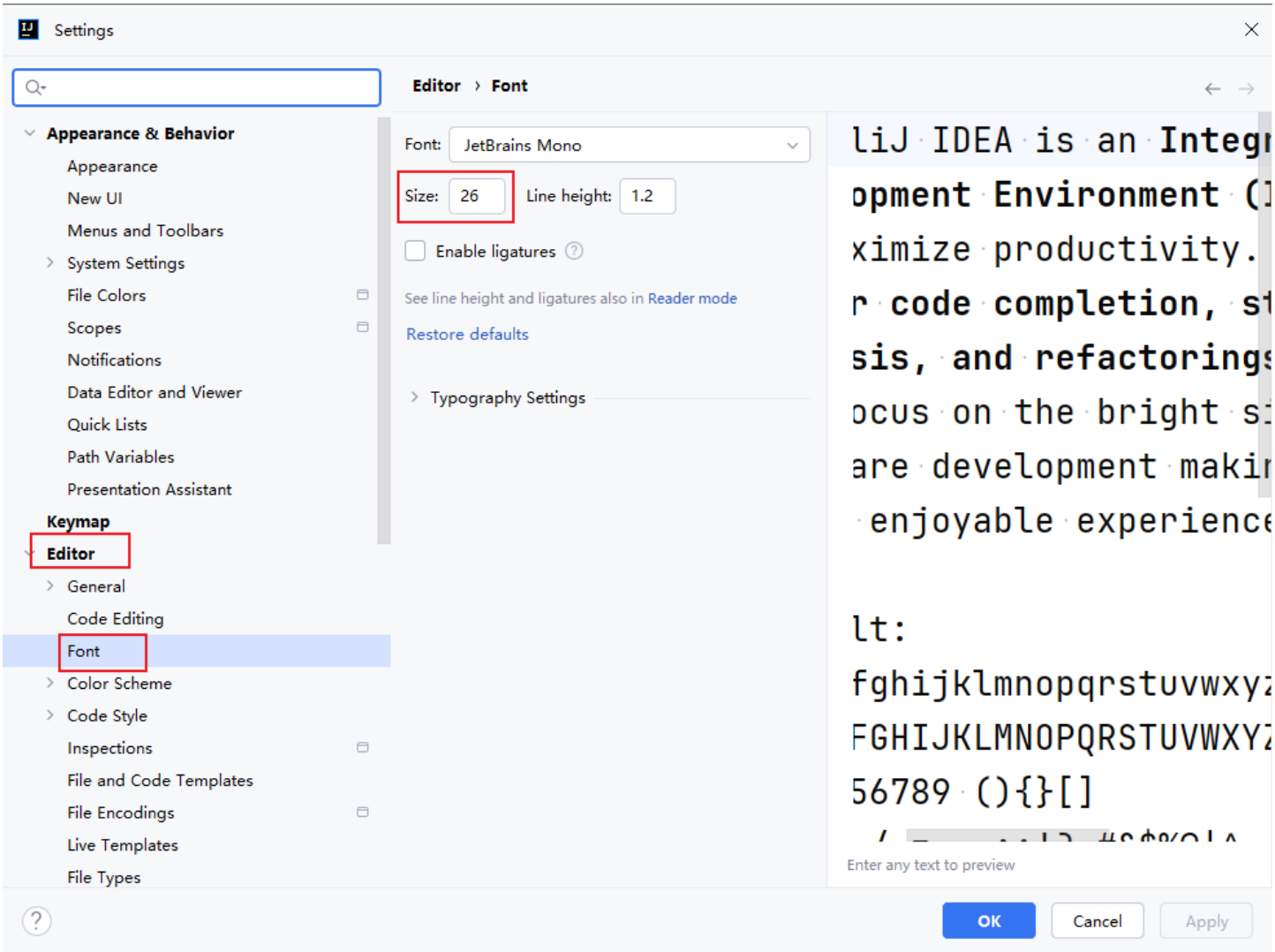



## 2.4.2 设置字体

### 2.4.2.1 外观字体

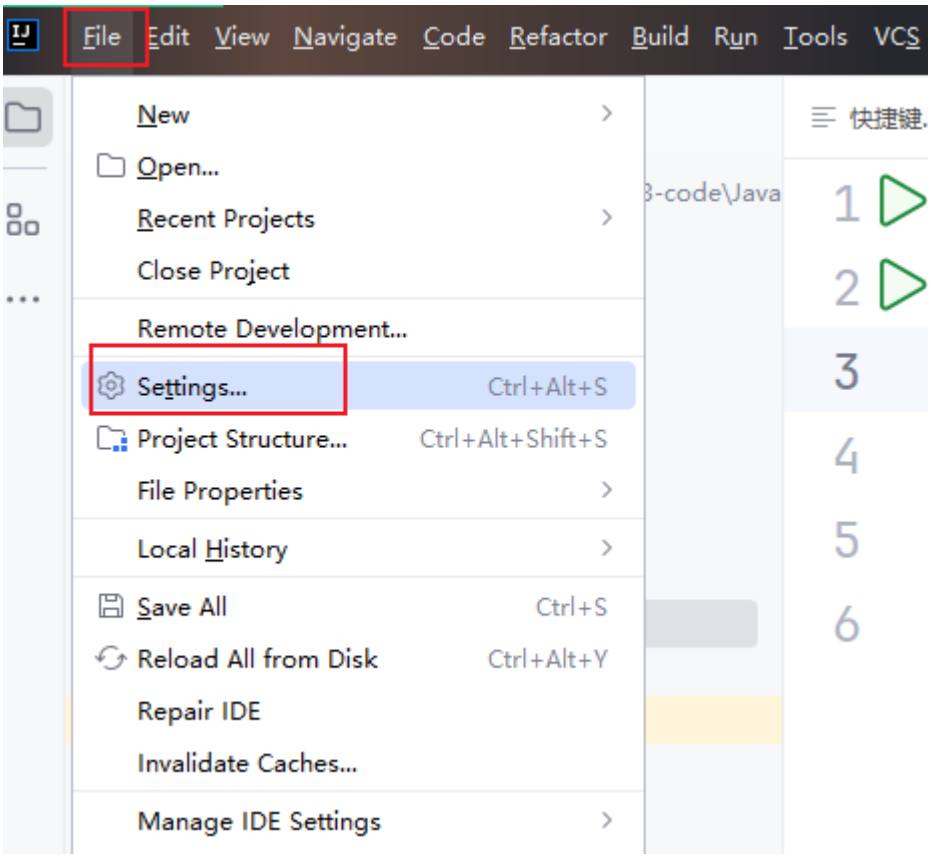

Settings
Appearance & Behavior
Appearance
New UI
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Data Editor and Viewer
Quick Lists
Path Variables
Presentation Assistant
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Settings Sync
Advanced Settings
Appearance & Behavior 〉 Appearance
Zoom: 100% Change with Alt+Shift+= or Alt+Shift+减号. Set to 100% with Alt+Shift+0
Use custom font: Microsoft YaHei UI
Size: 12
字体大小
Support screen readers Requires restart
Ctrl+Tab and Ctrl+Shift+Tab will navigate UI controls in dialogs and will not be available for switching editor tabs or other IDE actions. Tooltips on mouse hover will be disabled.
Use contrast scrollbars
Adjust colors for red-green vision deficiency How it works
Requires restart. For protanopia and deuteranopia.
UI Options
Show tree indent guides
Use smaller indents in trees
Enable mnemonics in menu
Enable mnemonics in controls
Show main menu in a separate toolbar
Use project colors in main toolbar
Distinguish projects with different toolbar colors at a glance. Only for new UI.
Background Image...
Smooth scrolling
Drag-and-drop with Alt pressed only
Merge main menu with window title Requires restart
Always show full path in window header
Display icons in menu items
OK
Cancel
Apply

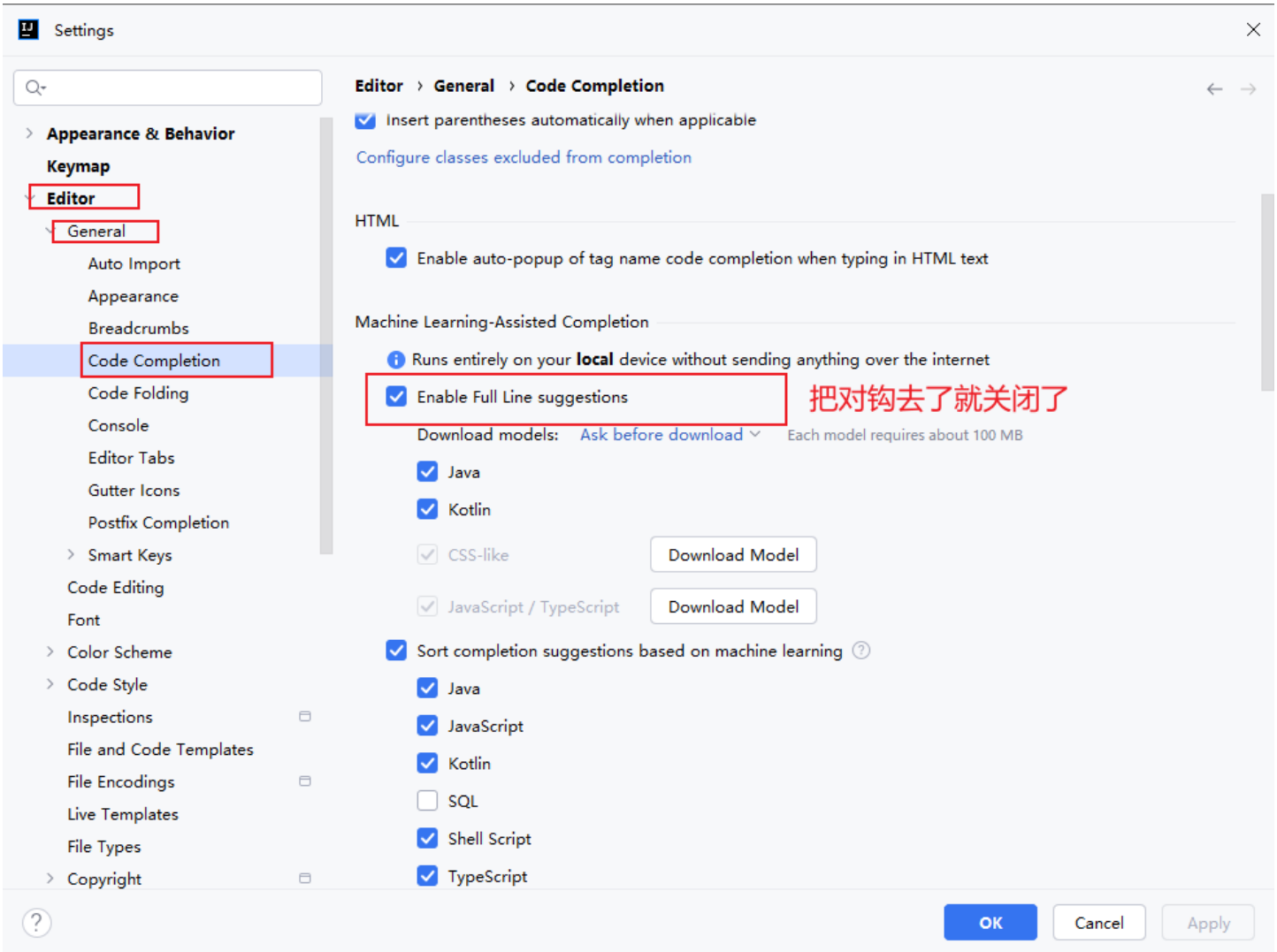
### 2.4.2.2 编辑区字体




### 2.4.3 关闭全行建议

```
public class Test05 {  no usages
    public static void main(String[] args) {}  Choose key for completion: Tab ▾
}
```


File Edit View Navigate Code Refactor Build Run Tools VCS
New
Open...
Recent Projects
Close Project
Remote Development...
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
快捷键.
1
2
3
4
5
6


Settings
Editor › General › Code Completion
Insert parentheses automatically when applicable
Configure classes excluded from completion
HTML
Enable auto-popup of tag name code completion when typing in HTML text
Machine Learning-Assisted Completion
Runs entirely on your local device without sending anything over the internet
Enable Full Line suggestions
把对钩去了就关闭了
Download models: Ask before download
Each model requires about 100 MB
Java
Kotlin
CSS-like
Download Model
JavaScript / TypeScript
Download Model
Sort completion suggestions based on machine learning
Java
JavaScript
Kotlin
SQL
Shell Script
TypeScript
Appearance & Behavior
Keymap
Editor
General
Auto Import
Appearance
Breadcrumbs
Code Completion
Code Folding
Console
Editor Tabs
Gutter Icons
Postfix Completion
Smart Keys
Code Editing
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
OK
Cancel
Apply

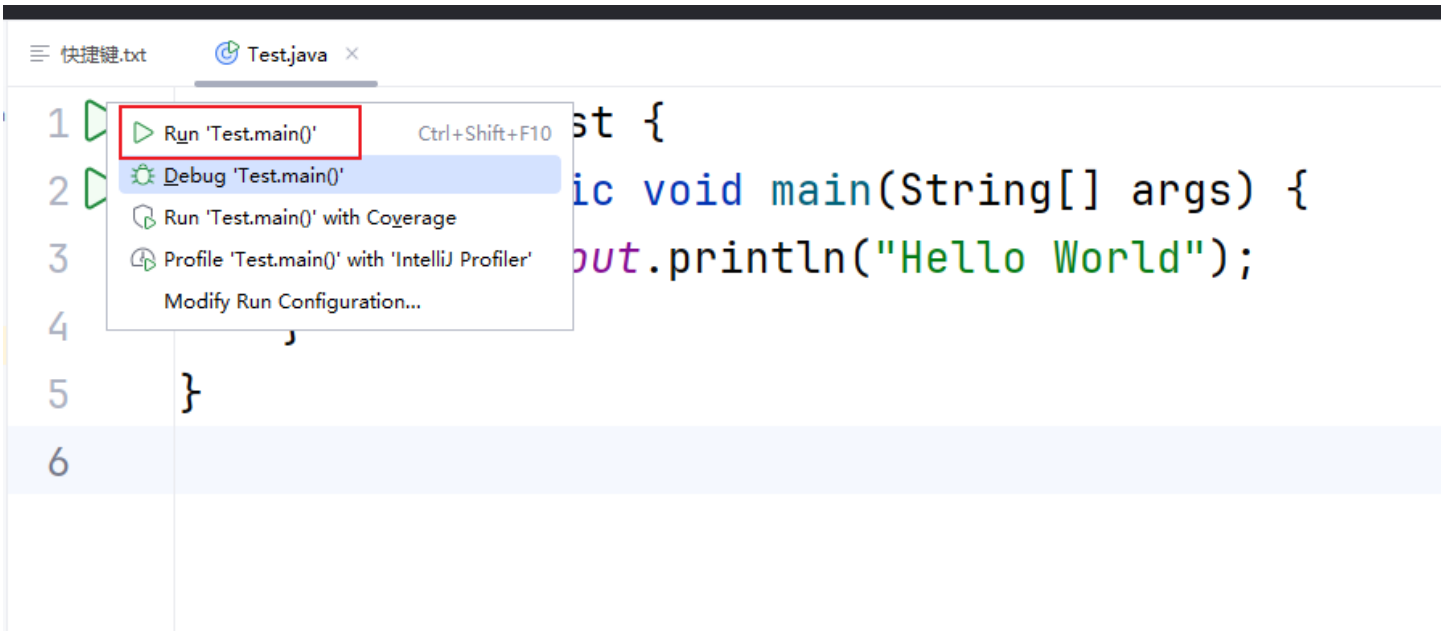
# 3. 使用idea开发

## 3.1 编写阶段

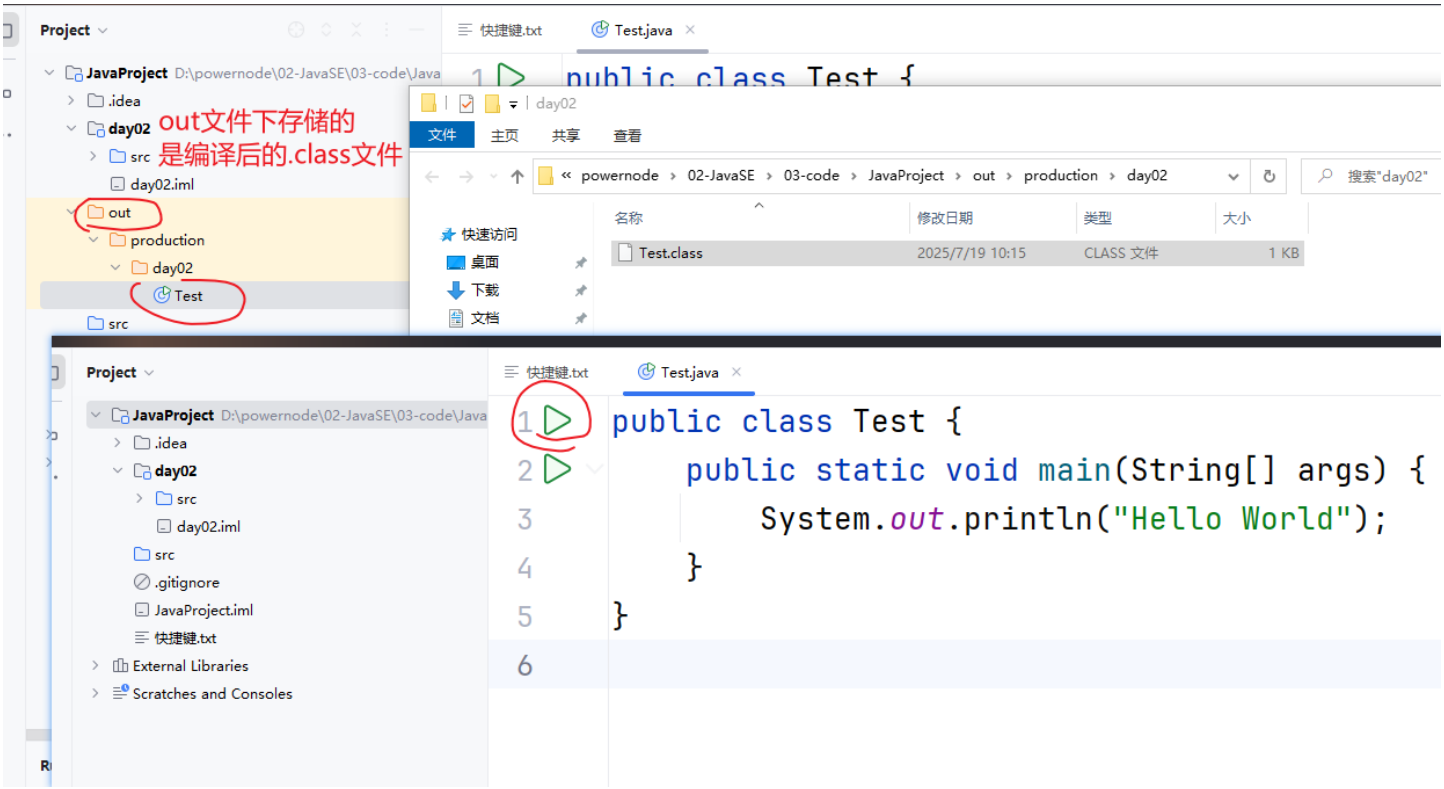
代码块

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
1.shift + enter :光标切换到下一行
2.main 按 enter :编写main方法
3.sout 按 enter :编写输出语句
```

## 3.2 运行阶段




- 在运行之前，idea会有一个自动编译的过程




## 3.3 print和println的区别

代码块

```
public class Test01 {
    //main方法是程序的入口方法，程序执行先执行main方法
    public static void main(String[] args) {
        System.out.println("abc");
        System.out.print(123);
        System.out.println("def");
    }
}

1.print 没有换行
2.println有换行
```

# 4. 注释（掌握）

- 是什么？有什么用？怎么用？

## 4.1 注释是什么

- 注释：是程序员对所写代码的解释说明，主要是给程序员看，JVM并不解释它

## 4.2 注释有什么用

- 提高可读性，程序有解释说明，其他程序员更容易看懂

- 暂时无用的代码进行注释

## 4.3 注释怎么用

### 4.3.1 单行注释

1. 语法：// 需要写的说明文字，注释的代码
2. 举例：

代码块

```
public class Test02 {
    // main方法是一个程序的入口方法，程序执行的时候先从main方法开始
    public static void main(String[] args) {
        //System.out.println("下班去哪里玩? ");
        System.out.println("去后海吧! ");
    }
}
```

### 4.3.2 多行注释

1. 语法：/* 需要写的说明文字，注释的代码 */
2. 举例：

代码块

```
public class Test03 {
    /*
        1.main方法是一个程序的入口方法
        2.程序执行的时候先执行main方法
        3.main不写，编译不会报错
        4.运行会报错，提示找不到main方法
     */
    public static void main(String[] args) {
        //System.out.println("下班去哪里玩? ");
        System.out.println("去后海吧! ");
    }
}
```

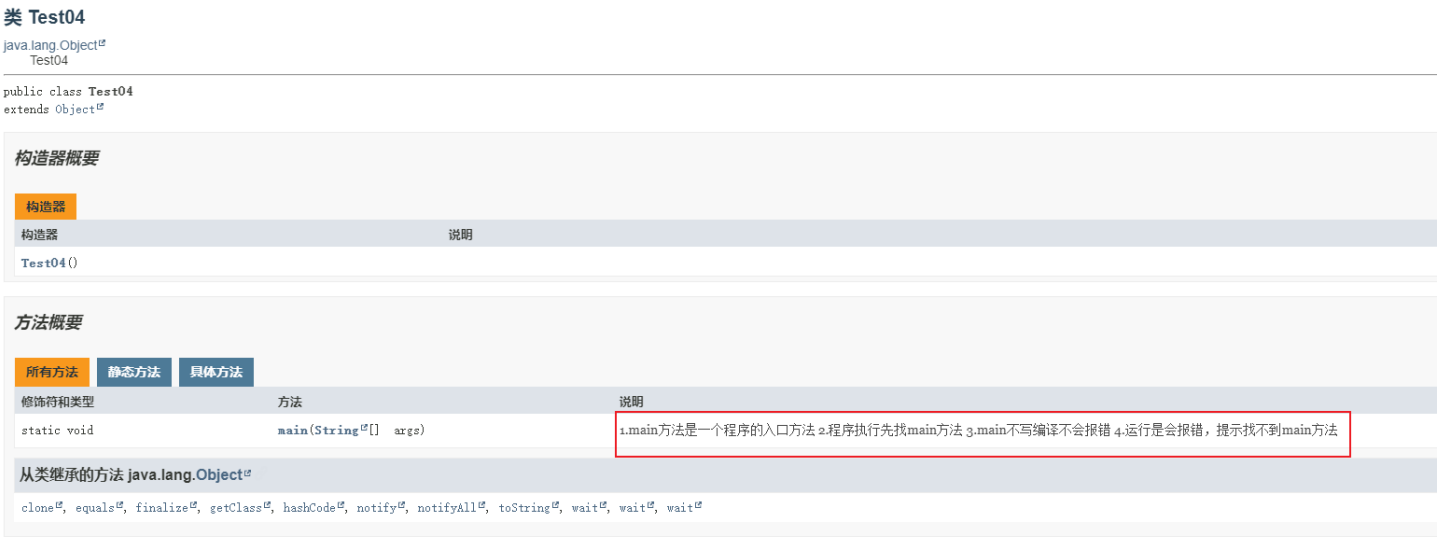
### 4.3.3 文档注释

1. 语法：/** 需要写的说明文字，注释的代码 */
2. 举例：

```java
public class Test04 {
    /**
     * 1.main方法是一个程序的入口方法
     * 2.程序执行先找main方法
     * 3.main不写编译不会报错
     * 4.运行是会报错，提示找不到main方法
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

### 4.3.4 多行注释和文档注释的区别

1. 文档注释和多行注释都可以注释多行
2. 文档注释：使用工具（javadoc.exe）生成文档时，注释存在，多行注释不存在
3. 使用javadoc工具生成文档
   a. 语法：javadoc 文件名称.java
   b. 举例：javadoc Test04.java



**类 Test04**

java.lang.Object
  Test04

public class **Test04**
extends Object

***构造器概要***

构造器

| 构造器 | 说明 |
| --- | --- |
| Test04() | |

***方法概要***

所有方法 | 静态方法 | 具体方法

| 修饰符和类型 | 方法 | 说明 |
| --- | --- | --- |
| static void | main(String[] args) | 1.main方法是一个程序的入口方法 2.程序执行先找main方法 3.main不写编译不会报错 4.运行是会报错，提示找不到main方法 |

从类继承的方法 java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

4. 在实际的开发过程中，一般情况下
   a. 在类，字段和方法上使用文档注释
   b. 在方法内部使用多行注释

# 5. 标识符命名规范（掌握）

## 5.1 标识符是什么

- 标识符是程序员自定义的一些名称

## 5.2 标识符有什么用

- 标识符是用来做识别
  - 现实生活中的标识符，人名，班级名
  - 计算机中的标识符：类名，方法名，变量名等

## 5.3 标识符怎么用

1. 规则
   a. 数字不可以开头
   b. 不能使用Java关键字和保留字
2. 规范
   a. 尽量使用英文，数字，_和$命名
      - 不建议使用$,因为内部类生成的.class文件，默认添加了$
   b. 尽量做到见名知意
   c. 尽量遵循驼峰标识
      i. 类，接口命名：
         1. 单词，首字母大写，其余字母小写
         2. 比如：UserDao,UserService
      ii. 变量和方法命名
         1. 首单词小写，其余单词首字母大写
         2. 比如：getUserNameById();

## 5.4 阿里的命名规约