类 对象 包

1.类

- 你写的 Java 程序，最小的单位就是“**类**”（class）
- `.java` 是你写的源代码文件，编译之后会变成 `.class` 文件（给 JVM 执行用的）
- 每个类都像一个“**蓝图**”或“模板”，定义了**对象的属性和行为**

比如要描述学生，就写一个 Student 类：

```
public class Student {
    String name;
    int age;

    void study() {
        System.out.println(name + " is studying");
    }
}

这个类定义了学生的名字、年龄、行为（study）
```

2.类 & 对象

- 类（class）是“模板”
- 对象（object）是“用模板做出来的产品”

就像“汽车”类是模板，每辆车（对象）是基于这个模板生产的

```
Student s1 = new Student();
Student s2 = new Student();
```

`s1` 和 `s2` 是两个对象，它们都是从 `Student` 类里造出来，但它们可以有不同的名字、年龄

3.包 package

- 可以把 **package 理解成文件夹/目录**
- 它用来把代码分门别类地放好，避免**同名类**发生冲突

📦 包的作用：

1. **组织代码，让结构更清晰、可维护**
2. **防止同名类冲突**
3. **控制访问权限**（哪些类能被别人看到/调用）
4. **提供封装边界**（隐藏内部细节，暴露外部接口）

🧠 包名 = “姓”，类名 = “名”

✅ 类的全名 = 包名 + 类名

- 如果有两个类同名，比如两个不同包下的 `Student`：

  com.school.Student
  com.company.Student

  编译器就通过“姓”区分说的是哪个 Student

✅ 如果有这样的文件结构：

src/main/java/com/example/Hello.java

那么这个 Java 文件的开头就应该是：

```
package com.example;
```

> 包名就是这个 Java 文件所在的文件夹路径（从 src/main/java 开始）👉**包名是由目录结构决定的**

🧾 **如果某个 class 文件直接写在 src/main/java 下面，就是默认包（没有包名）**

📌 总结：

> Java 程序是由 类（class） 组成的，每个类就是一个蓝图，可以创建多个对象
> 为了防止类名冲突和更好地组织代码，Java 使用package来分类这些类，就像用文件夹整理文件一样

📦 类是一种“数据类型”

```
int x = 10;      // int 是基础数据类型
Dog d = new Dog(); // Dog 是类，也是一个“自定义数据类型”
```

🔧 类可以定义两类内容：

1. **属性**（也叫字段、状态） ➜ 比如 `name`, `age`
2. **方法**（也叫行为、函数） ➜ 比如 `bark()`, `eat()`

```
src/main/java/my/cute/a/Main.java
```

📌 这个类的包名就是：`my.cute.a`

📌 类名是 `Main`

全限定类名 ➡️ my.cute.a.Main

- **全限定类名 Fully Qualified Class Name（FQCN）**

📥 import 简化使用

🔁 如果你不 import：

你每次用 `Cat` 类都要写全名，很麻烦：

```
com.github.hcsp.pet1.Cat c1 = new com.github.hcsp.pet1.Cat();
```

✅ import：

```
import com.github.hcsp.pet1.Cat;
```

就可以直接写：

```
Cat c1 = new Cat();
```

👉 `import` 的作用：**引入类，简化代码书写**

📦 java.lang 包默认自动导入

有一个叫 `java.lang` 的包，里面放的是最基础、最常用的类，例如：

- `String`
- `System`
- `Object`

这些类你在 Java 里可以直接用，**不用 import**：

```
String s = "hello";    // 不用 import java.lang.String
System.out.println(s); // 不用 import java.lang.System
```

🔷 基本结构：

- Java 中一个 `.java` 文件一般只定义**一个类**（当然可以定义多个类，但**只允许一个 `public` 类**）
- `class` 是**定义“说明书”**，告诉 Java：“我怎么造某种东西”
- 用 `new` 去“造”出来的才叫**对象（object）**

🔷 每个对象都是独立的副本

> 每一个对象都是彼此独立的个体，它们虽然来自同一个 class（模板），但它们拥有自己的属性值和内存空间

更贴近实际运行机制的说法：

> 每一个对象是内存中分配的一块独立空间，用于存储它的成员变量数据

🔷 现实世界的类比 ：

✅

> 创建一个 computer 的类，keyboard 的类，screen 的类，mouse 的类
>
> 把 keyboard、screen、mouse 放到 computer 里
>
> computer 的构造器定义了如何创建这些组件
>
> 你 new computer 的时候就自动 new 了这些组件

✨ 组合类（类中包含其他类）：

```
class Keyboard {}
class Screen {}
class Mouse {}

class Computer {
    Keyboard keyboard;
    Screen screen;
    Mouse mouse;

    // 构造函数里自动创建键盘、屏幕、鼠标
    Computer() {
        keyboard = new Keyboard();
        screen = new Screen();
        mouse = new Mouse();
    }
}
```

当你：

```
Computer myPC = new Computer();
```

你并不是只创建了一个 computer，而是：

➡️ 同时在 computer 的构造函数中自动 **new** 了 keyboard、screen、mouse

🔷 你 new 的电脑，可以有默认配置 or 自定义参数

✅ 原文：

> 你买 (new) 了一个 computer
>
> 里面所有东西默认就有
>
> 也可以自己指定一些鼠标、键盘配置（参数）

✨ 构造函数支持传参配置

```
class Computer {
    Keyboard keyboard;
    Screen screen;
    Mouse mouse;

    Computer(Keyboard k, Screen s, Mouse m) {
        keyboard = k;
        screen = s;
        mouse = m;
    }
}

// 自定义配置
Computer myCustomPC = new Computer(new Keyboard("机械"), new Screen
("4K"), new Mouse("无线"));
```