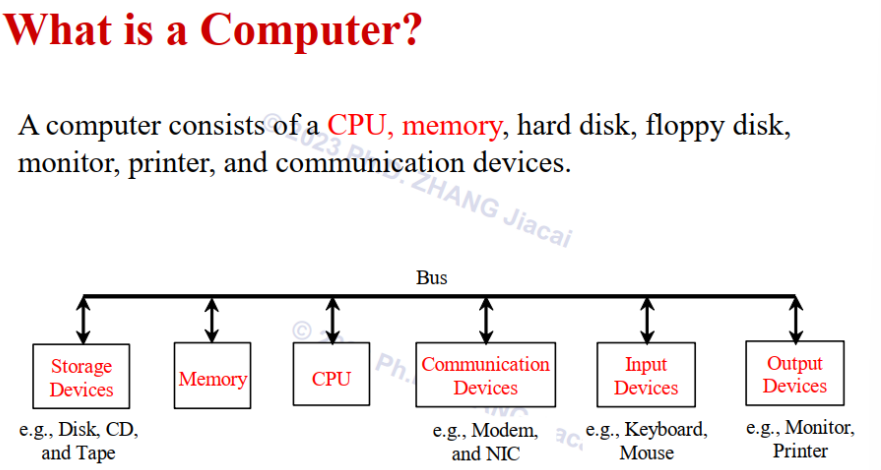
Java概述

方法:
对于现实世界的复杂系统，如何分析，如何设计，如何建模，统一建模语言(UML)
面向对象方法是运用**类、对象、继承、封装、重载、多态、关联、消息**等概念来构造系统的软件开发方法
设计范式：经过反复使用的代码设计经验，每种模式描述了一个在我们周围不断重复发生的问题，以及该问题的核心解决方案
技术:
用编程语言来实现面向对象的系统建模的知识与技能;
Java编程是技术层面，掌握了一门语言是掌握了一种技术，Java是纯面向对象的编程语言

概论

计算机基本组成



CPU

计算机核心(central processing unit)
从内存中提取并处理命令
衡量处理速率：megahertz(MHz，兆赫兹)， 1 MHz = 1 million pulses/second

Memory(内存)

存储数据并编写指令(store data and program instructions)
一个字节串规划了一个内存单元，**每个字节包含8比特(bits)**
在所有程序和数据被处理前必须被加入内存

> A memory unit is an ordered sequence of *bytes*, each holds *eight bits*. A program and its data must be brought to memory before they can be executed.
> A memory byte is *never empty*, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it

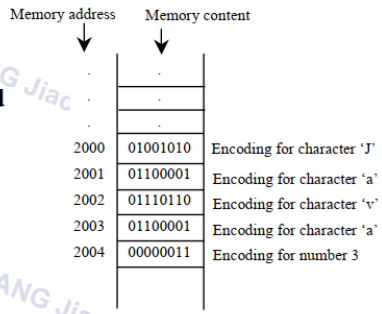
数据的存储方式(二进制代码)

Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (zeros and ones).

The programmers need not to be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme.

The encoding scheme varies. For example, character 'J' is represented by 01001010 in one byte

If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.



存储设备(Storage Devices)

Memory is volatile, because information is lost when the power is off.

Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them.

There are three main types of storage devices:Disk drives (hard disks and floppy disks), CD drives (CD-R and CD-RW), and Tape drives.

内存易变，因为当电源关闭时里面的数据就消失了
程序和数据往往*永久性*存储在存储设备中并且当计算机要调用时才会移动到内存中

> 内存（RAM）和硬盘（硬盘驱动器）是计算机系统中的两种不同类型的存储设备，它们在功能和工作方式上有很大的区别。
>
> **1. 内存（RAM - Random Access Memory）：**

- **作用：** 内存是计算机用于*临时存储数据和程序*的地方。当计算机启动时，操作系统和正在运行的程序都会加载到内存中。
  - **速度：** 内存的读写速度非常快，可以迅速存取数据，是CPU直接访问的地方。
- **易失性：** 内存是*易失性存储*，也就是说，当计算机关闭或重启时，内存中的数据将被清除。

> **2. 硬盘（硬盘驱动器）：**

- **作用：** 硬盘是计算机中用于长期存储数据和程序的设备。它保存操作系统、应用程序、用户文件等。
  - **速度：** 相对于内存，硬盘的读写速度较慢。硬盘通常是机械设备，需要一定的时间来寻找和读取数据。
- **非易失性：** 硬盘是*非易失性存储*，即使在关机时数据也会被保留。

> **区别：**

- **速度：** 内存速度快，适合需要快速访问的临时数据；硬盘速度慢，适合长期存储。

- **易失性：** 内存是易失性存储，关机时数据丢失；硬盘是非易失性存储，数据持久保存。

- **用途：** 内存用于存放正在运行的程序和临时数据；硬盘用于存放操作系统、应用程序和用户文件等长期存储的数据。

- **成本：** 一般情况下，内存的价格比硬盘高。
  总的来说，内存和硬盘在计算机系统中扮演不同的角色，相互协同工作以实现计算机的正常运行。

输出设备(Output Devices)

> The monitor displays information (text and graphics). The resolution and dot pitch determine the quality of the display

通信设备(Communication Devices)

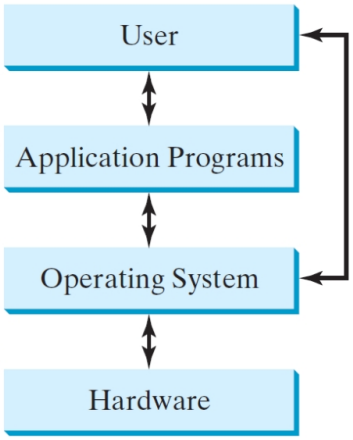
操作系统(Operation Systems)

Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities.

The popular operating systems for general-purpose computers are Microsoft Windows, Mac OS, and Linux.

Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.



机器语言、汇编语言与高级语言

> Computer programs, known as *software*, are instructions to the computer.
> You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to *use computer languages to communicate with them*.
> Programs are written using *programming languages*.

机器语言(Machine Language)

- 一系列原始指令
- 二进制代码形式

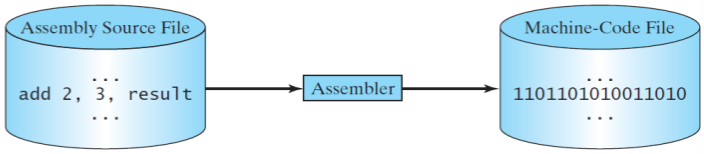
> Program with native machine language is a *tedious process*. Moreover the programs are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this: 1101101010011010

汇编语言(Assembly Language)

通过编译器将汇编语言转化为二进制代码

Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADD F3 R1, R2, R3



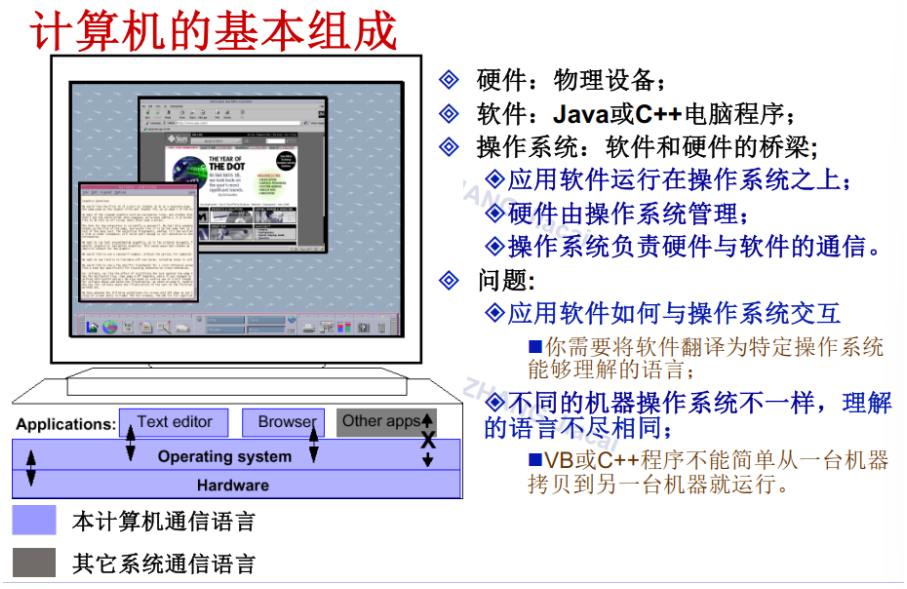
高级语言

易懂

Java语言特性

- Java是鲁棒的程序设计语言
- 易于学习
- 可以独立于平台运行小程序和应用程序
  主要靠Java虚拟机（JVM, Java Virtual Machine)在目标及实现平台无关性
  JVM解释执行
  “一处编译多处执行”
- 面向对象

计算机基本组成



Interpreting/Compiling Code

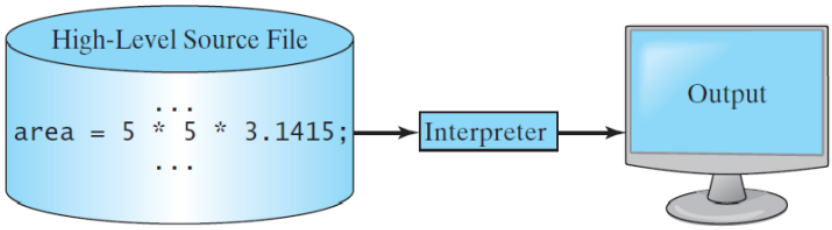
> A program written in a high-level language is called a source program or *source code*.
> Because a computer cannot understand a *source program*, a source program must be translated into *machine code* for execution.
> The translation can be done using another programming tool called an *interpreter* or a *compiler*.

解释器(Interpreter)

每次从源码解释一条语句(statement)为机器码并立即执行，一条源码可能被解释成多条机器指令
缺点：比较慢
优点：容易跨平台，只需要对应平台安装对应解释器即可

An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away, as shown in the following figure.

Note that a statement from the source code may be translated into several machine instructions.



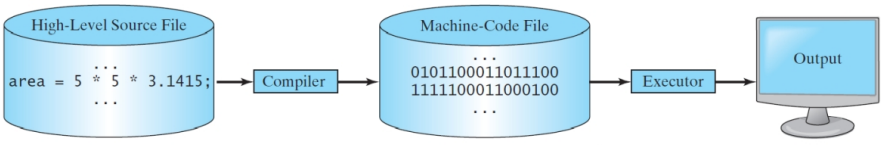
编译器(Compiler)

将所有源码一次性转化为机器语言(二进制代码)文件并执行
优点：执行速度快
缺点：依赖具体的机器

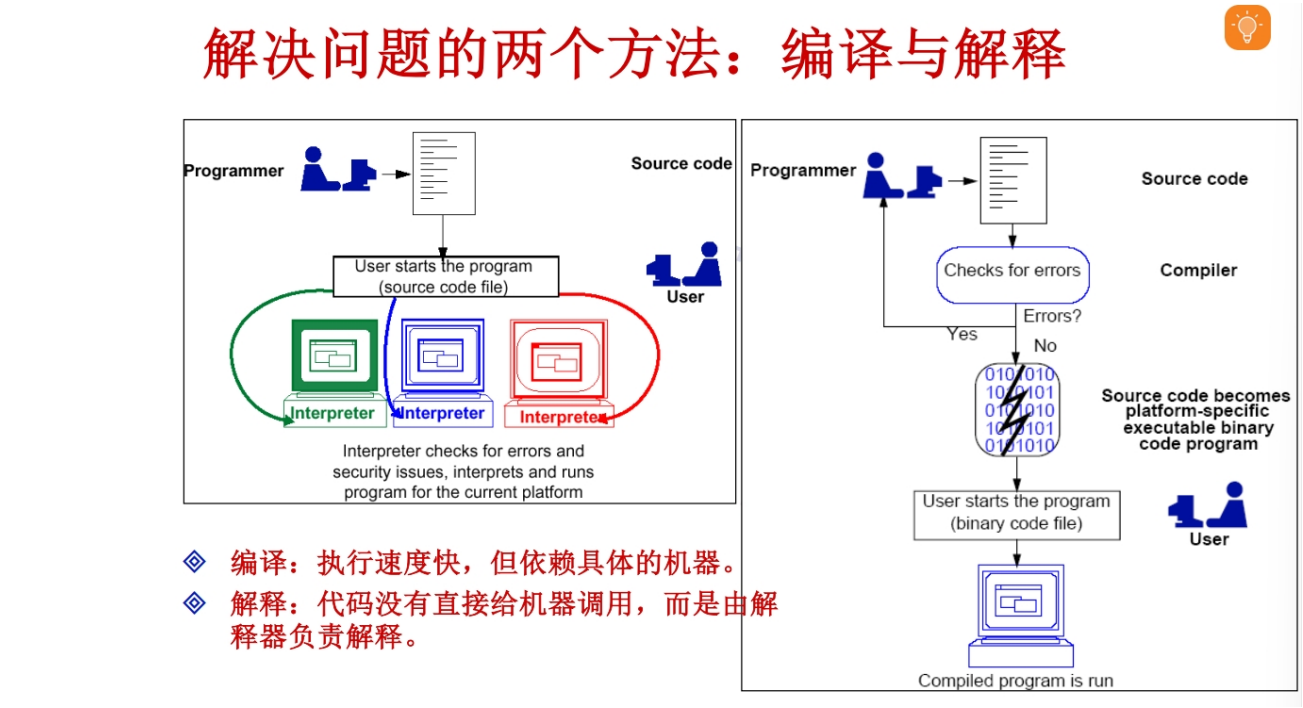
解释执行用户得到的是源码，编译执行用户得到的是exe文件

Compiling Source Code

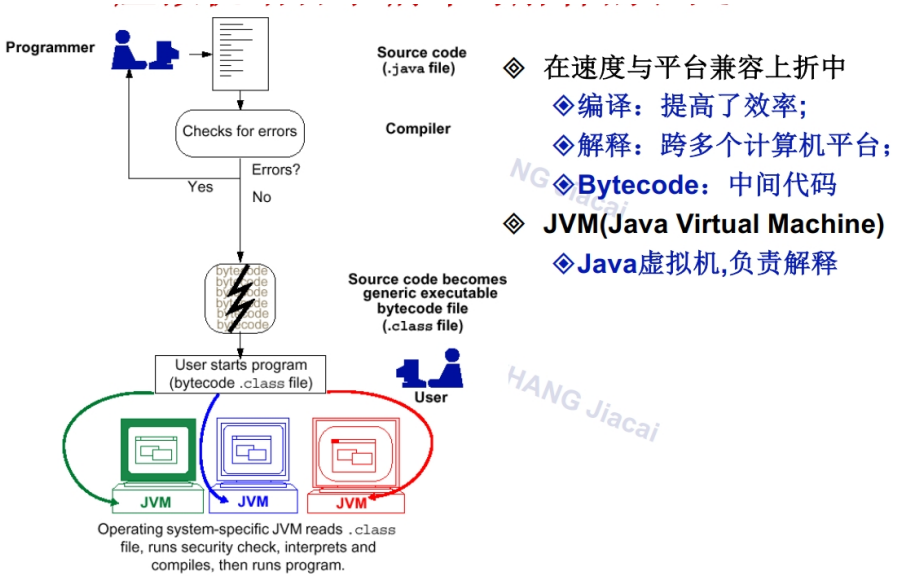
A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



两种方式比较



Java虚拟机结合了编译与解释的长处



将源码编译成八进制码(class文件)，再在不同平台上解释运行

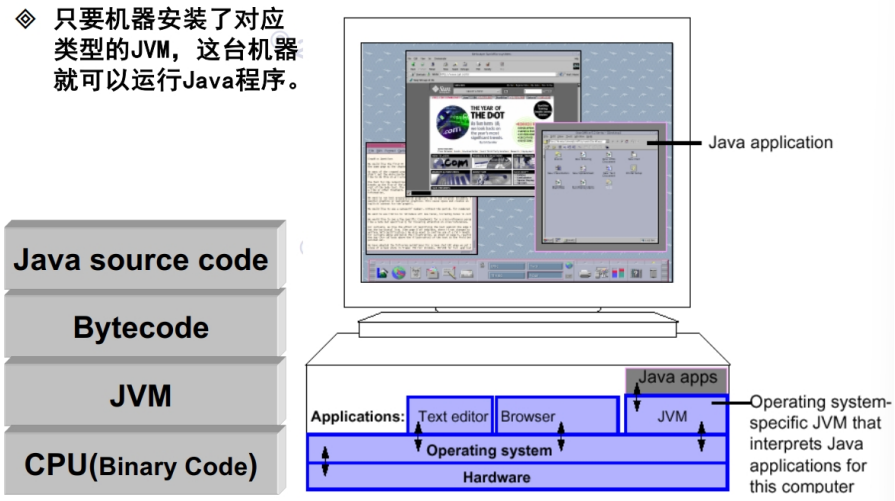
在速度与平台兼容上折中：

- 编译：提高了效率
- 解释：跨平台
- Bytecode：八进制码，机器读不懂，需要解释器

JVM(Java VIrtual Machine)：

- Java虚拟机，负责**解释**
- 虚拟机有多个版本，不同平台安装不同虚拟机
- 只要机器安装了对应类型的JVM，这台机器就可以运行Java程序

Java程序如何运行



Java语言的特性

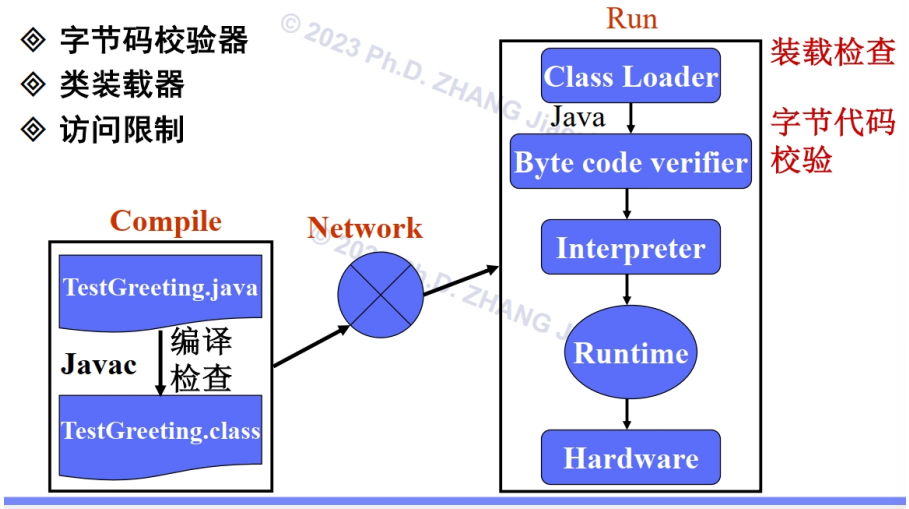
可靠性

- 强类型规定
- 摒弃指针
- 自动垃圾收集
- 运行时检查
- 异常处理机制

安全性

- 字节码校验器
- 类装载器
- 访问限制(封装)

Java语言的特性——安全性

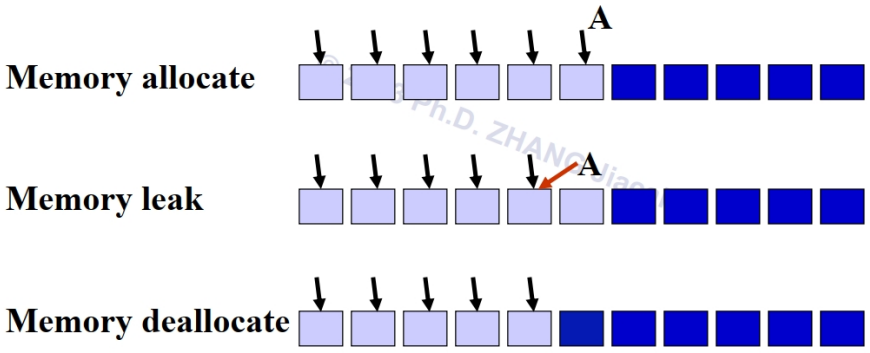


多线程

- Java环境本身就是多线程的
- Java语言内置多线程控制，可大大简化多线程应用程序开发
- Java的多线程支持在一定程度上受到运行时支持平台的限制

自动垃圾回收

- JVM的垃圾回收功能
  - 在其它语言中，程序员要负责分配和回收无用内存空间。
  - JVM将自动负责内存分配与回收



总结

Java语言特点总结

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

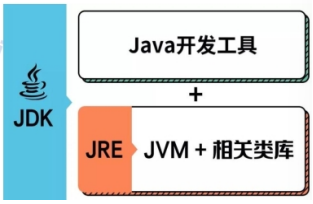
Java开发环境

JDK

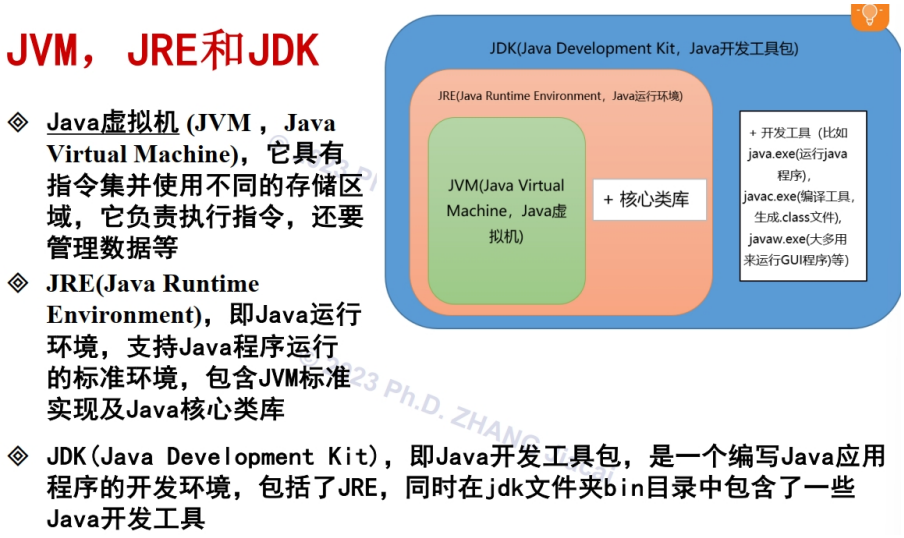
- javac：*Java编译器*，将源代码转换为字节代码，生成*class文件*
- java：*Java解释器*，直接从类文件*执行*Java应用程序代码
- javadoc：根据Java源代码及其说明语句生成的HTML文档
- javah：产生可以调用Java过程的C过程，或建立能被Java程序调用的C过程的头文件

Java集成开发环境(IDE)

- WebSphere Studio Application Developer (WSAD)
- JBuilder
- Visual Age for Java
- Jcreator
- Eclipse



JVM,JRE和JDK



第一个文件: Hello World:

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

cmd窗口指令:
cd: 改变当前目录，一个点到上一级; cd. : 当前目录; cd.. : 切换到副目录
dir: 显示目录文件
DOS: 磁盘操作系统

配置路径:

- 找到JDK安装位置
- 打开环境变量
- 添加JAVA_HOME=C:\...
- path添加%JAVA_HOME%\bin
- 添加CLASSPATH=.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar

另一种方法：使用eclipse执行

eclipse的配置

- 将文件夹设置为原本设好的javacode文件夹中
- new java program - hello
- new class - HelloWorld ， 选择`public static void main(String [] args)`，相当于`main`函数

创建两个class执行

第一个: Greeting:

```java
package hello;

public class Greeting{
        private String sal;
        Greeting(String s){
                sal = s;
        }
        public void greet(String whom){
                System.out.println(sal+" "+whom);
        }
}
```

第二个：TestGreeting:

```java
package hello;

public class TestGreeting{

        public static void main(String[] args){
        Greeting hello = new Greeting ("Hello");
        hello.greet("World");
        }
}
```

Java代码编写

- Java区分大小写，各种括号成对出现
- *常量命名全部大写*，单词间用下划线隔开，力求语义表达完整
- *类名、变量名一般用单词首字母大写的混合编写方法*
- 注释符号"//"，对程序无影响
- 每个source file **最多一个public class**，文件名为公开类名；多个类可以放在一起，只要只有一个public class就可以

Java程序编译与动行

例如：

JAVA

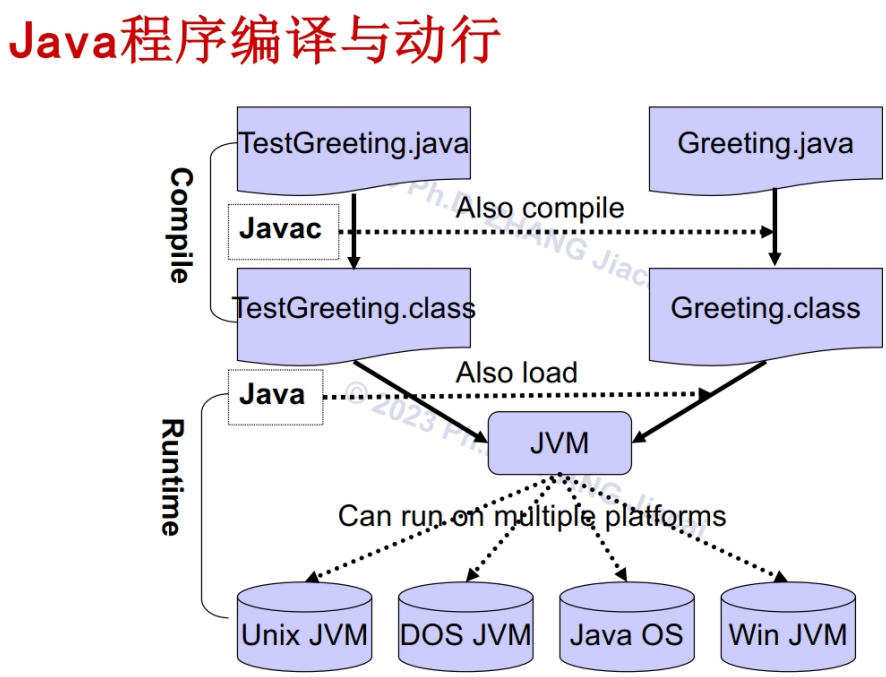
```java
package hello;

public class TestGreeting{

        public static void main(String[] args){
        Greeting hello = new Greeting ("Hello");
        hello.greet("World");
        }
}

class Greeting{
        private String sal;
        Greeting(String s){
                sal = s;
        }
        public void greet(String whom){
                System.out.println(sal+" "+whom);
        }
}
```

- 主类中的`main`函数格式固定，是程序的入口点
  `main(String[] args) / main(String args[])`
- javac编译后在同目录下生成同名的.class文件



Java源文件结构

- Java源文件结构(Source File Layout)：
- BNF(Backus Naur Form)符号
  - “::=“意思为“定义为”；
  - <>里面是描述条款；
  - []代表可选项；
  - *代表有0或多个；+代表有1或多个；
- 这些条目的顺序很重要；
- 如果有public class，源文件名取其名，否则无限制。

```
package shipping.report;

import shipping.domain.*;
import java.util.List;

public class Vehicle{
   private List vehicles;
   ……
}
```

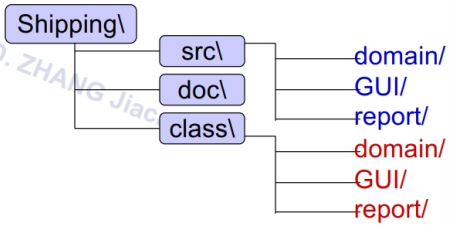
<source_file>::=
[<package_declaration>]
<import_declaration> *
<class_declaration> +

Package语句

Package语句

- Java提供了包机制，把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用
- 如同文件夹一样，包也采用了树形目录的存储方式。
- 同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。
- 包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类

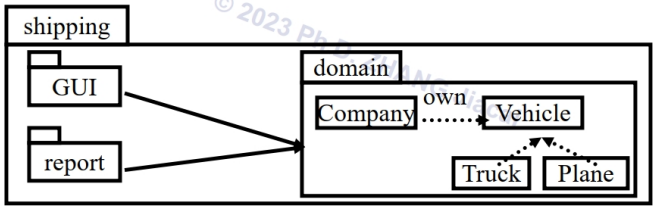
- Java中每个类都可以生成对应的说明文档，放在doc目录
- Java API文档是一组HTML文件。每个包文档都包含指向其它类的超连接
- 每个类文挡包含类的体系结构、描述、成员、构造器等



Package语句

<package_declaration::=
package <top_pkg_name>[.<sub_pkg_name >]*;

- **包（Package）包含类和其它包，这样形成一个包的树形结构，有助于大软件的组织。**
- **一个文件最多一个package语句位于文件的开始（空行或注释除外）；**
- **包层次以“.”来分隔，通常按照意义的类间相关性将多个类组织成包；**
- **如果没有package语句，该文件中所有类都归属于缺省包（无名字）；**
- **包通常使用小写单词，而类名通常是每个单词的首字母大写。**

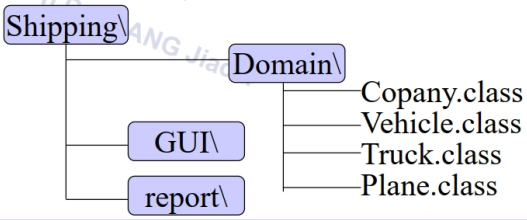


import语句

import语句

```
<import_declaration>::=
import <pkg_name>[.<sub_pkg_name >]*.<class_name|*>;
```

- import语句告诉编译器去哪里寻找类，事实上包的名字也是类名字的一部分。
- 如果要访问包中所有的类，使用*；否则指明要访问的类名。
- 同一个包中的任何类或子包可以直接使用，无须import。
- import用于将其它包的内容引入当前可用的名字空间。
- import语句位于package语句和类声明之间，它只是定义路径并不装载类；
- 包对应着同名的目录。



Java语言元素

- **标识符**
  - 标识符是用于标识Java程序中的类、方法、变量等元素的名称。
  - 以字母、__、$ 开始作为首字符
  - 其他字符可以是上面三种字符和数字
  - Java关键字不是标识符，但可以作为其中的一部分
  - 标识符大小写相关，且无长度限制

合法标识符	
identifier	userName
user_Name	_sys_var
$change	$classCost

非法标识符	
7group	try#
#roomNo	user.Name
open-door	1_class
boolean	public

- 关键字
  - Java语言本身使用的标识符

◈ 关键字是Java语言本身使用的标识符

◈ Java关键字（Keywords）对于Java编译器有特殊意义：

- ◈ 数据类型名字；
- ◈ 代表程序结构；

◈ Java关键字请注意。

- ◈ true，false，null都是小写，这与C++不同；
- ◈ 没有sizeof操作符，所有类型的大小都固定不依赖与具体实现；
- ◈ 没有byvalue，byref等关键字；
- ◈ goto和const是关键字，但在Java中不再使用

Java关健词

abstract	do	implements
private	throw	boolean
double	import	protected
throws	break	else
instanceof	public	transient
byte	extends	int
return	true	case
false	interface	short
try	catch	final
long	static	void
char	finally	native
super	volatile	class
float	new	switch
while continue	for	null
synchronized	default	if
package	this	

- 数据类型
- 运算符
- 分隔符
  - 用来确认代码在何处分割
  - 如" "，' ；"：' 都是Java语言分隔符

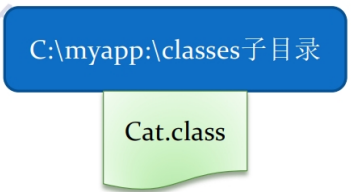
补充

◈ 在用javac命令编译Java源程序时，以下哪个选项用来指定编译生成的类文件的存放路径？

A) -classpath
B) -sourcepath
C) -path
D) -d

```
javac -sourcepath C:\myapp\src
      -d C:\myapp\classes
      C:\myapp\src\Cat.java
```

答案：D



北京师范大学

课堂小问题(3)

◈ 运行“java Cat”命令时，java命令可能从哪里寻找Cat.class类文件？

A) classpath系统环境变量
B) -classpath选项
C) -d选项
D) path系统环境变量

```
java -classpath C:\myapp\classes Cat
```

答案：A,B

C:\myapp\classes\Cat.class

关键字（Keywords）在Java中具有特殊含义，与标识符和变量名有显著的区别：

1. **关键字（Keywords）：**
   - 关键字是Java语言预定义的、具有特定意义的单词。每个关键字在Java语言中都有固定的、预先定义的含义，且这个含义不能被改变。
   - 关键字用于执行特定的功能和任务，如定义数据类型、控制程序流程等。例如，`if`、`class`、`public`、`static`、`void`、`new` 等。
   - 关键字不能被用作标识符或变量名。也就是说，你不能将一个关键字用作类名、方法名或变量名。
2. **标识符：**
   - 标识符是程序员定义的名字，用于标识变量、方法、类、接口等。它们遵循特定的命名规则，但可以自由地由程序员选择。
   - 标识符可以是任何合法的名称，只要它不是Java的关键字，并且遵循命名规则。
3. **变量名：**
   - 变量名是标识符的一种，用于标识存储数据的内存位置。变量名的命名也必须遵守Java的标识符命名规则，并且不能使用关键字。

区别：

- **预定义与自定义：** 关键字是Java语言预定义的，具有特定的含义和用途，不能更改；而标识符和变量名是由程序员自定义的，可以自由命名。
- **用途：** 关键字用于指定Java语言的结构和控制程序流程；标识符用于为变量、方法、类等提供自定义名称。
- **命名限制：** 关键字是固定的单词，不能用作标识符或变量名；标识符和变量名有较大的自由度，但需要遵守命名规则且不得使用关键字。