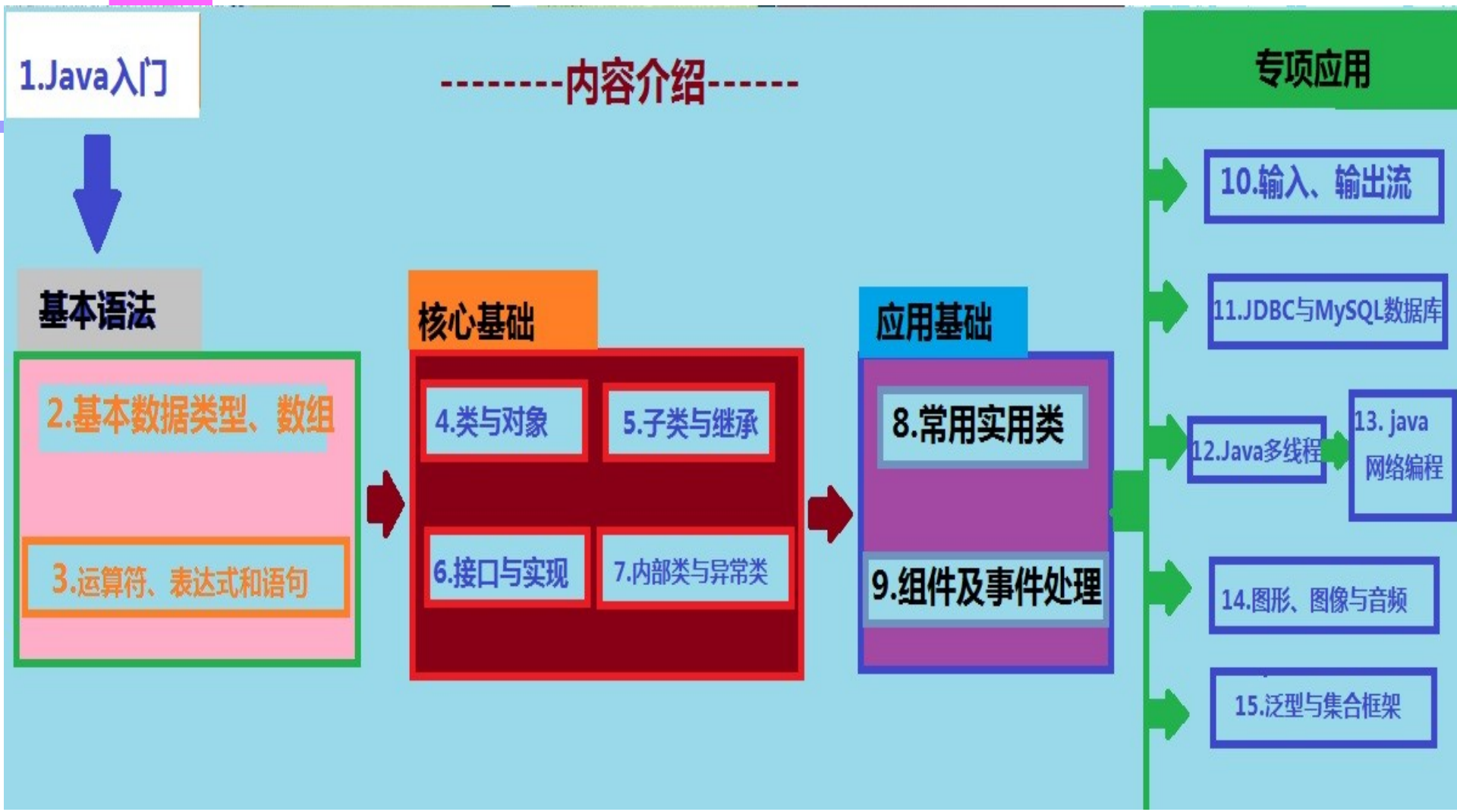
计算机科学技术学院
School of Computer Science and Technology

Java 程序设计

第 1 章　Java 入门


1.Java入门
--------内容介绍------
专项应用
基本语法
2.基本数据类型、数组
3.运算符、表达式和语句
核心基础
4.类与对象
5.子类与继承
6.接口与实现
7.内部类与异常类
应用基础
8.常用实用类
9.组件及事件处理
10.输入、输出流
11.JDBC与MySQL数据库
12.Java多线程
13. java 网络编程
14.图形、图像与音频
15.泛型与集合框架

导读

主要内容

- Java 诞生、地位
- Java 的特点
- 安装 JDK
- 简单的 Java 应用程序

重点与难点：

- **重点**： Java 平台无关性、 Java 程序的结构
- **难点**： Java 程序的开发过程

Java 的诞生

oak

Java 的先导知识与后继技术

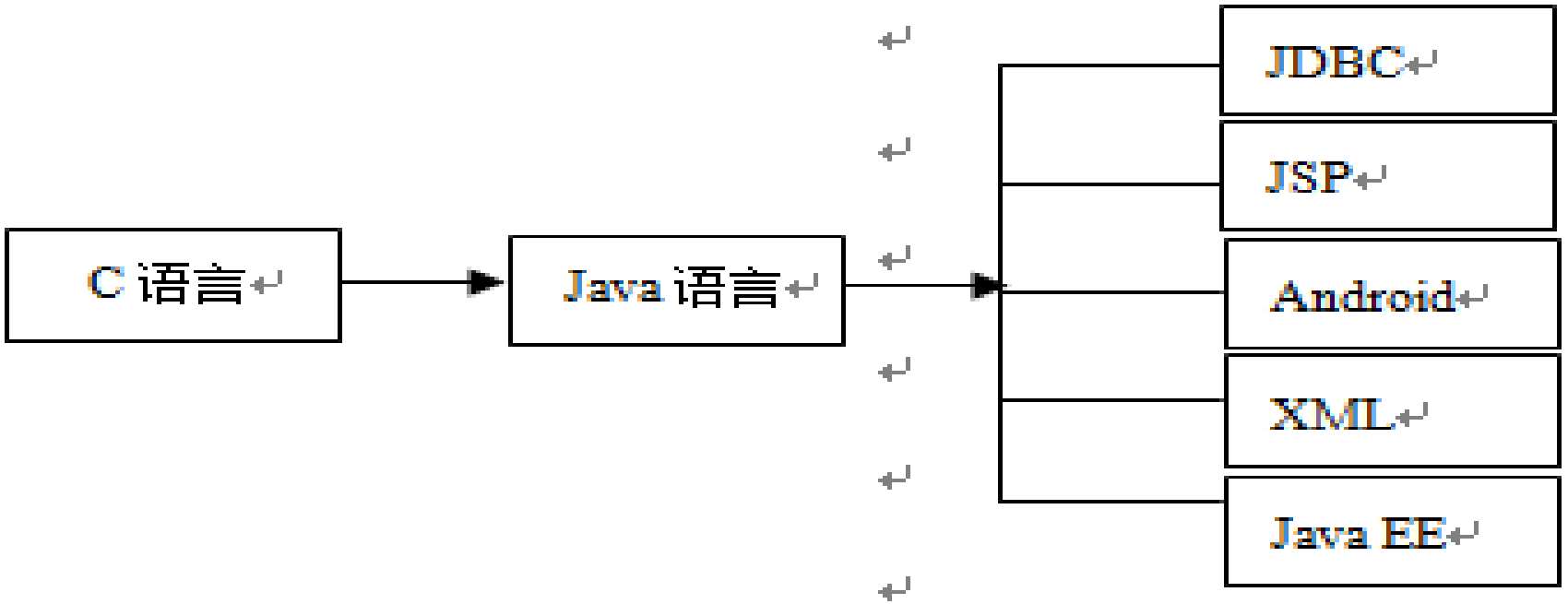


图 1.1　Java 的先导知识与后继技术

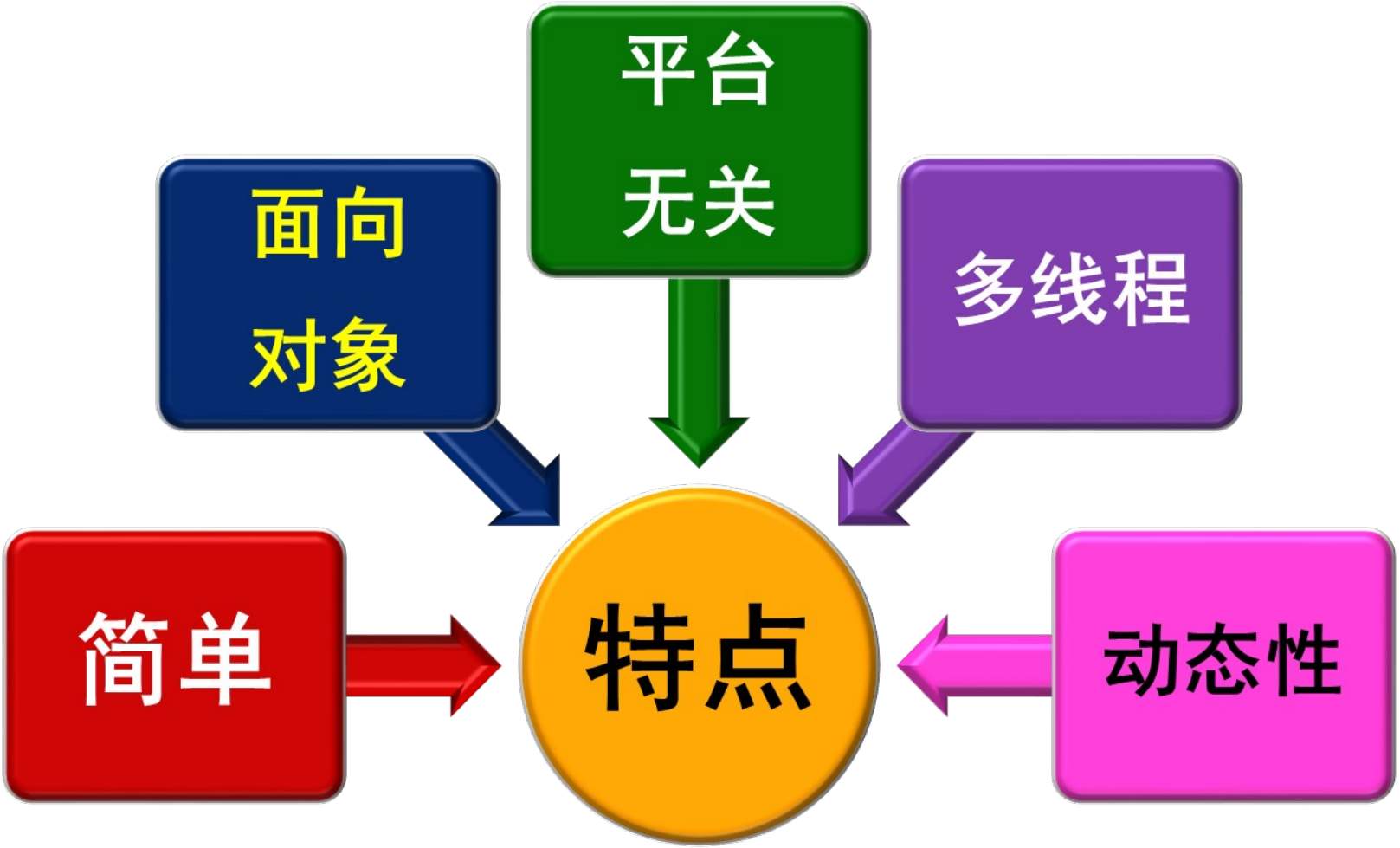
1.1 Java 的地位

- 网络地位
  - Java 的平台无关性使得 Java 特别适合于网络应用软件的设计与开发
- 语言地位
  - 很好的面向对象语言，通过学习 Java 语言可以学习怎样使用对象来完成某些任务、掌握面向对象编程的基本思想
- 需求地位
  - IT 行业对 Java 人才的需求正在不断的增长，掌握 Java 语言及其相关技术意味着较好的就业前景和工作酬金

Jul 2020	Jul 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.45%	+2.24%
2	1	↓	Java	15.10%	+0.04%
3	3		Python	9.09%	-0.17%
4	4		C++	6.21%	-0.49%
5	5		C#	5.25%	+0.88%
6	6		Visual Basic	5.23%	+1.03%
7	7		JavaScript	2.48%	+0.18%
8	20	⇈	R	2.41%	+1.57%
9	8	↓	PHP	1.90%	-0.27%
10	13	↑	Swift	1.43%	+0.31%
11	9	↓	SQL	1.40%	-0.58%
12	16	⇈	Go	1.21%	+0.19%
13	12	↓	Assembly language	0.94%	-0.45%
14	19	⇈	Perl	0.87%	-0.04%
15	14	↓	MATLAB	0.84%	-0.24%
16	11	⇊	Ruby	0.81%	-0.83%
17	30	⇈	Scratch	0.72%	+0.35%
18	33	⇈	Rust	0.70%	+0.36%
19	23	⇈	PL/SQL	0.68%	-0.01%
20	17	↓	Classic Visual Basic	0.66%	-0.35%

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	23	21	-	-
C++	4	4	4	3	2	1	3	9
C#	5	5	5	7	9	-	-	-
JavaScript	6	8	8	8	6	-	-	-
PHP	7	6	3	4	25	-	-	-
SQL	8	-	-	96	-	-	-	-
Swift	9	181	-	-	-	-	-	-
Ruby	10	11	9	23	31	-	-	-
Objective-C	12	3	15	37	-	-	-	-
Lisp	27	21	14	13	8	5	6	2
Fortran	29	28	21	14	16	4	2	12
Ada	34	26	24	15	15	6	7	3
Pascal	235	13	12	53	13	3	8	5

1.2 Java 的特点 _1



1．平台与机器指令

- **这里所指的平台 = 操作系统 + 处理器 CPU**

  - 每个平台都会形成自己独特的机器指令，所谓平台的机器指令就是可以被该平台直接识别、执行的一种由 0， 1 组成的序列代码
  - 相同的 CPU 和不同的操作系统所形成的平台的机器指令可能是不同的
  - 如某个平台可能用 8 位序列代码 **00001111** 表示加法指令，以 **10000001** 表示减法指令，而另一种平台可能用 8 位序列代码 **10101010** 表示加法指令，以 **10010011** 表示减法指令。

2． C/C++ 程序依赖平台

C/C++ 针对当前 C/C++ 源程序所在的特定平台对其源文件进行编译、链接，生成**机器指令**，即根据当前平台的机器指令生成**可执行文件**

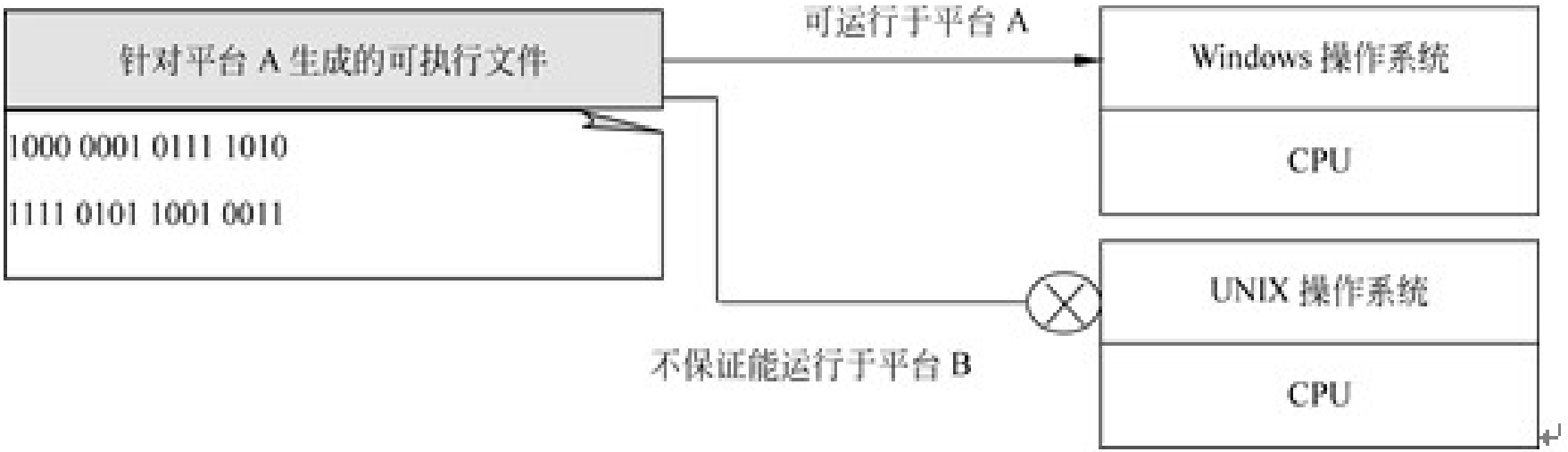


图 1.2 C/C++生成的可执行文件依赖于平台

3． Java 虚拟机与字节码

- Java 虚拟机把 Java 源程序编译成称为字节码的“中间代码”(0，1 组成的序列代码)，字节码并不是机器指令，与平台无关
- Java 虚拟机负责将字节码翻译成虚拟机所在平台的机器码，并让当前平台运行该机器码

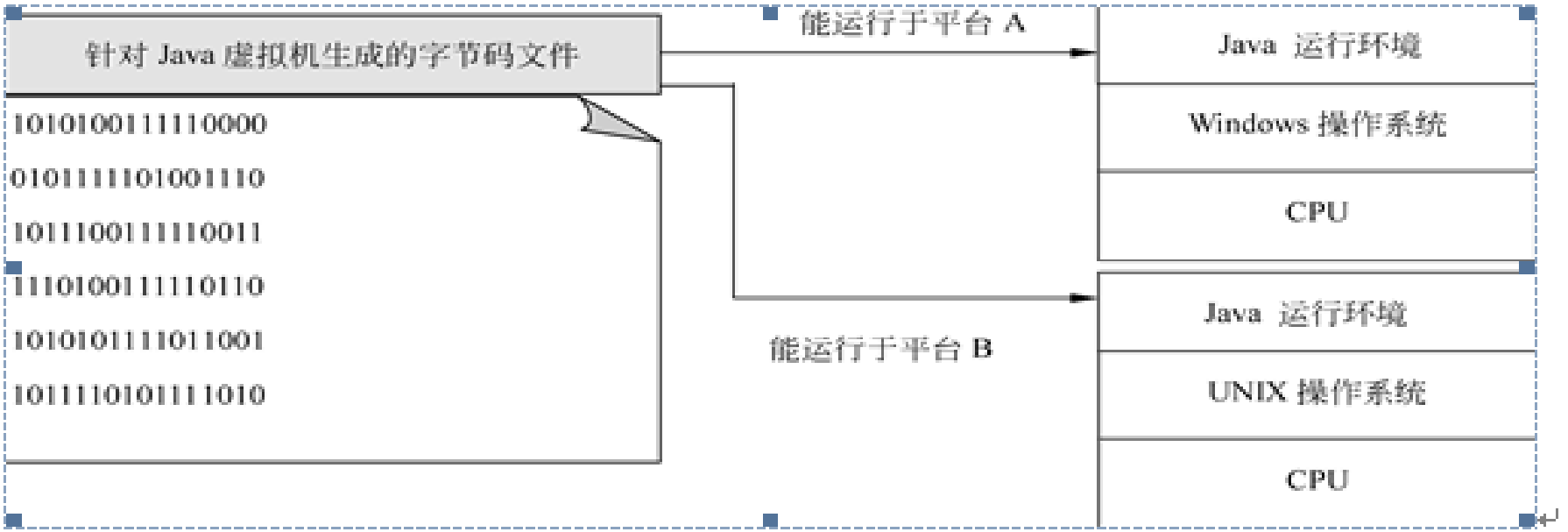
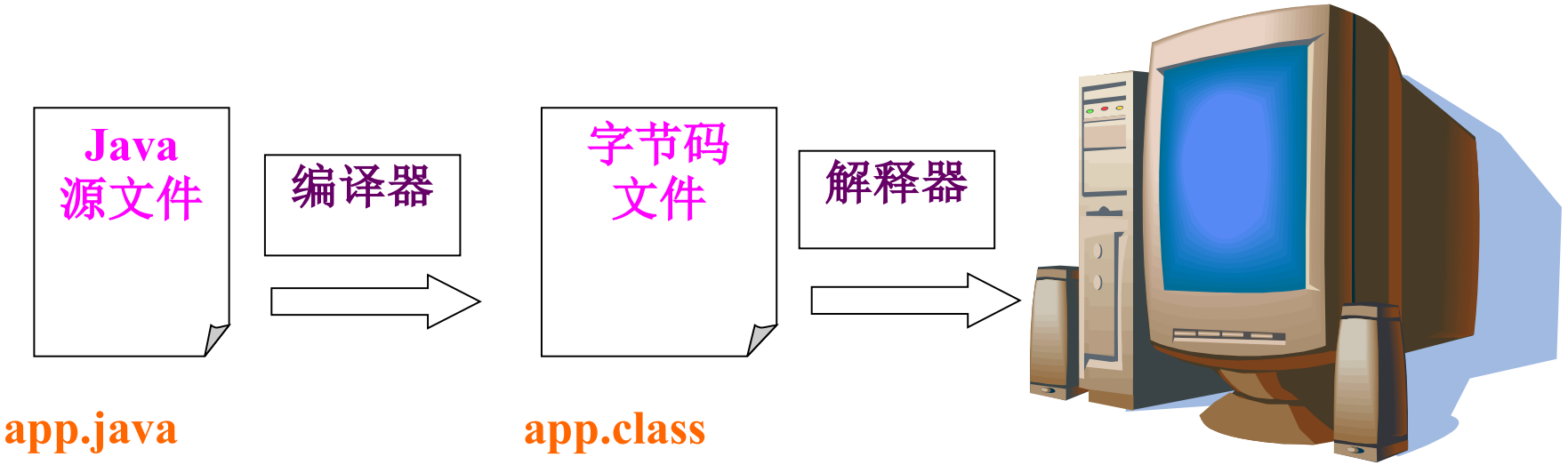


图 1.3 Java 生成的字节码文件不依赖于平台

3． Java 虚拟机与字节码

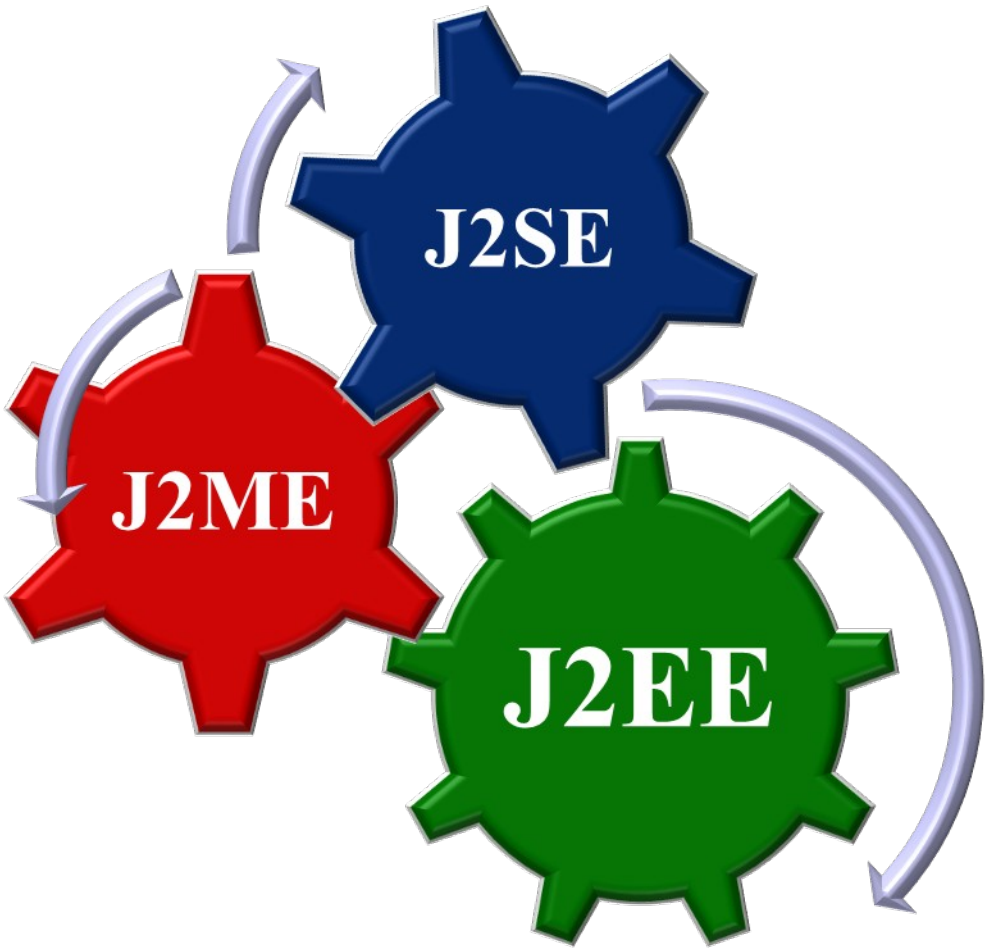
- Java 程序的运行过程如图所示：



一次编写，到处运行！！！

1.3 安装 JDK

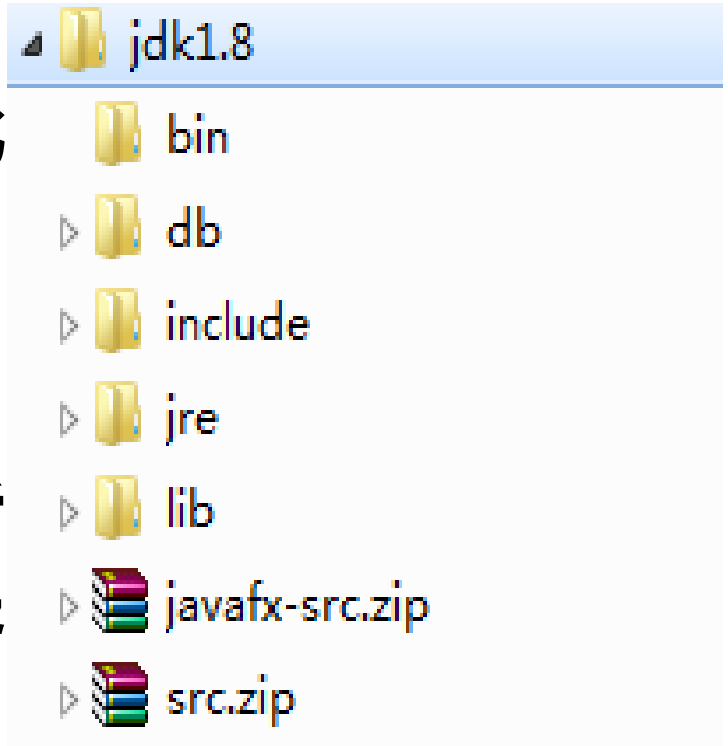
1.3.1 三种平台简介



1.3.2 安装 Java SE 平台

- **下载 JDK** 本书将使用针对 Window 操作系统平台的JDK，因此下载的版本为 jdk-8u102-windows-x64.exe

- **选择安装路径界面** 为了便于今后设置环境变量，建议修改默认的安装路径为: D:\jdk1.8



下载 JDK—jdk-12.0.2_windows-x64_bin.exe

https://www.oracle.com/technetwork/java/javase/downloads/jdk12-downloads-5295953.html

1．系统环境 path 的设置

设置系统变量 JAVA_HOME

- 添加系统环境变量 JAVA_HOME，让该环境变量的值是 JDK 目录结构的根目录，例如 E:\jdk1.8

图 1.7 设置系统变量 JAVA_HOME

1．系统环境 path 的设置

设置系统环境 path 的的值

- 编辑“系统变量 (S)” 中的 path， 添加新值 E:\JDK1.8\bin
- 用系统变量 %JAVA_HOME% 代替 E:\JDK1.8

图 1.9 编辑系统变量 path 的取值

2．系统环境 classpath 的设置

- 一般不需要设置环境变量 classpath 的值
- 如果希望使用最新的 Java 运行环境，就重新设置 classpath 的值，如图 1.10 。 (E:\jdk1.8\jre\lib\rt.jar;.;)

Classpath 值中的 (.;) 是指可以加载应用程序当前目录及其子目录中的类

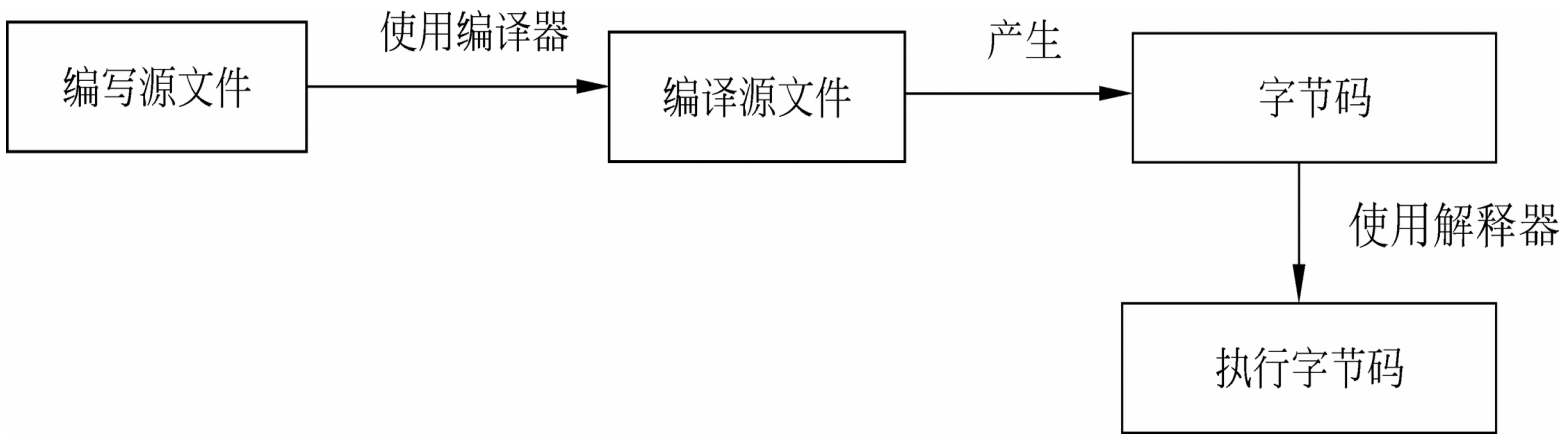
1.4 Java 程序的开发步骤

开发步骤:

① **编写源文件**——扩展名必须是 .java

② **编译 Java 源程序**——用 Java 编译器（javac.exe ）编译源文件，得到字节码文件

③ **运行 Java 程序**——使用 Java 解释器（java.exe ）来解释执行字节码文件



1.5 简单的 Java 应用程序

- Java 程序的结构：
  - 一个复杂的程序可以由一个或多个 Java 源程序文件构成，每个文件中可以有多个类定义

package 语句

import 语句

类定义（1 个或多个）

```
package ch01;          // 定义该程序属于 ch01 包
import java.io.*;      // 导入 java.io 类库中的所有类
public class App1_1    // 定义类： App1_1
{
  public static void main(String[] args)
  {
    char c= ' ';
    System.out.print(" 请输入一个字符： ");
    try{
        c=(char)System.in.read();
      }catch(IOException s){ }
    System.out.println(" 您输入的字符是： "+c);
  }
}
```

1.5 简单的 Java 应用程序

1.5.1 源文件的编写与保存

- Java 应用程序的源文件是由若干个互相独立的类组成

```
public class Hello {
    public static void main (String args[]) {
        System.out.println(" 大家好 !");
        System.out.println("Nice to meet you");
        Student stu = new Student();
        stu.speak("We are students");
    }
}
class Student {
    public void speak(String s) {
        System.out.println(s);
    }
}
```

编写与保存源文件

编写源文件

- 使用编辑工具
- 符合语法规范

保存源文件

- 只能有一个类是public类
- 扩展名是java

1.5.2 编译

1．编译器（javac）

- 使用 javac 编译源文件

C:\chapter1> javac Hello.java

```
C:\chapter1>javac Hello.java

C:\chapter1>
```

图 1.13　使用 javac 编译源文件

1.5.3 运行

1. 应用程序的主类

- 一个程序中可以有多个类，但只能有一个类是主类。在Java应用程序中，这个主类是指包含 main() 方法的类。主类是Java 程序执行的入口点

```
public static void main(String[] args)
```

1.5.3 运行

2. 解释器

- 使用 Java 虚拟机中的 Java 解释器（java.exe）来解释执行其字节码文件。
- Java 应用程序总是从主类的 main 方法开始执行。因此，需进入主类字节码所在目录，然后使用 Java 解释器（java.exe）运行主类的字节码

C:\chapter1\> java Hello

```
C:\chapter1>java Hello
大家好!
Nice to meet you
We are students
```

再看一个简单的 Java 应用程序

```
public class People {
    int height;
    String ear;
    void speak(String s) {
       System.out.println(s);
    }
}
```

```
class A {
    public static void main(String args[]) {
      People zhubajie;
      zhubajie = new People();
      zhubajie.height = 170;
      zhubajie.ear = " 两只大耳朵 ";
                    System.out.println("    身高 :"+zhubajie.height);
      System.out.println(zhubajie.ear);
       zhubajie.speak(" 师傅，咱们别去西天了，改去月宫吧 ");
    }
}
```

```
C:\1000>javac People.java

C:\1000>java A
身高:170
两只大耳朵
师傅,咱们别去西天了,改去月宫吧
```

1.6 Java 反编译

- JDK 提供的反编译器是 javap.exe
- javap 命令： javap Hello.class
  - C:\chapter1\>javap Hello.class

如果想反编译类库中的 Date 类（其包名是 java.util ） Date.class ， 可使用 javap 命令： javap java.util.Date.class ，例如：

C:\chapter1\> javap java.util.Date

1.7 编程风格

- 1.7.1 Allmans 风格 —— “独行”风格
- 1.7.2 Kernighan 风格 —— “行尾”风格

```
class Allmans
{
    public static void main(String args[])
    {
        int sum=0,i=0,j=0;
        for(i=1;i<=100;i++)
        {
            sum=sum+i;
        }
        System.out.println(sum);
    }
}
```

```
class Kernighan {
    public static void main(String args[]) {
        int sum=0,i=0,j=0;
        for(i=1;i<=100;i++) {
            sum=sum+i;
        }
        System.out.println(sum);
    }
}
```

1.7.3 注释

- Java 支持两种格式的注释：单行和多行
  - **单行注释**使用“ //”表示单行注释的开始，即该行中从“ //” 开始的后续内容为注释
  - **多行注释**的使用“ /*” 表示注释的开始，以“ */” 表示注释结束

总结

1 Java 语言是面向对象编程，编写的软件与平台无关

2 开发一个 Java 程序需经过三个步骤：编写源文件、编译源文件、 生成字节码、加载运行字节码

3 编写代码务必遵守行业的习惯风格。

作业

- 习题 1　　1, 3

思考

public static void main(String[] args)

- main 是否可以写成 Main ？
- String -> string ? String -> int
- void -> int, String
- 参数必须是 String 数组吗?