✠ 1 ✠ Java 应用与开发课程教学体系

很高兴同学门能够选修 Java 应用与开发课程。

希望我们一起通过这门课程的学习，建立 Java 语言编程的初步知识体系，掌握 Java 应用系统开发的方式、方法。更重要的，能够对编程这个事情、这项技能有更加深刻的认知，对未来的职业化发展有所促进。

Java 应用与开发课程的教学体系如图1.1所示，包括了 Java SE 和 Java EE 两个部分，每部分都涉及一些验证性实验，另外，会开展两次稍微大一点的集成开发项目。同时，在学习的过程中会穿插一些开发工具、设计模式、应用服务器和数据库的基本应用。

在课程学习的过程中，希望同学们要有足够的求知欲，养成良好的学习态度，具备不断探索的精神，多尝新、多实践、多总结。我想这是计算机专业人士应该具备的基本素养。

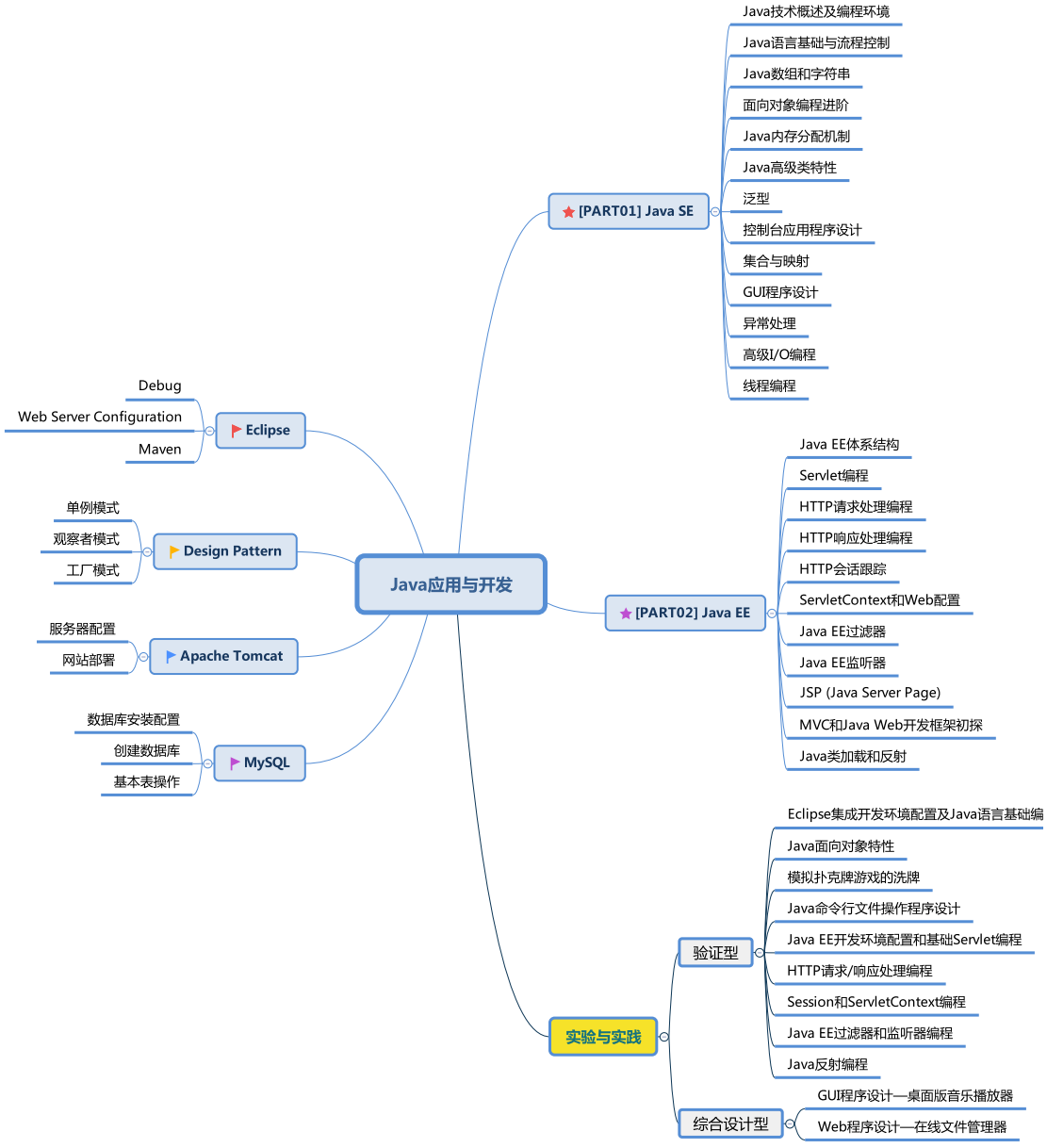

图 1.1: Java 应用与开发课程教学体系

✠ 2 ✠ Java 技术概述及开发环境

基本信息

课程名称： Java 应用与开发

授课教师： 王晓东

授课时间： 第一周

参考教材： 本课程参考教材及资料如下：

- 陈国君主编，Java 程序设计基础（第 5 版），清华大学出版社，2015.5
- Bruce Eckel, Thinking in Java (3rd)

教学目标

1. 讲解 Java 的发展历程，从 Java 的视角回顾 OOP；
2. 理解 Java 平台的相关概念和机制；
3. 掌握基本的 Java 开发环境配置方法。

授课方式

理论课： 多媒体教学、程序演示

实验课： 上机编程

教学内容

2.1 Java 技术概述

2.1.1 Java 发展简史

Java 的发展过程中伴随着多个伟大公司的起起落落。

1982 Sun 公司成立（安迪 · 贝托谢姆和麦克尼利）。

1986 Sun 公司上市。

1985 Sun 公司推出著名的 Java 语言。

2001 9.11 事件前，Sun 市值超过 1000 亿美元；此后，由于互联网泡沫的破碎，其市值在一个月内跌幅超过 90%。

2004 Sun 公司和微软在旷日持久的 Java 官司中和解，后者支付前者高达 10 亿美元的补偿费。

2006 共同创始人麦克尼利辞去 CEO 一职，舒瓦茨担任 CEO 后尝试将 Sun 从设备公司向软件服务型公司转型，但不成功。

2010 Sun 公司被甲骨文公司收购。

Java 语言的版本迭代历程如图2.1所示。

2.1.2 Java 技术的特点

Java 具备以下技术特点：

面向对象 Java 是一种以对象为中心，以消息为驱动的面向对象的编程语言。

平台无关性 分为源代码级（需重新编译源代码，如 C/C++）和目标代码级 (Java) 平台无关。

分布式 可支持分布式技术及平台开发。

可靠性 不支持直接操作指针，避免了对内存的非法访问；自动单元回收功能防止内存丢失等动态内存分配导致的问题；解释器运行时实施检查，可发现数组和字符串访问的越界；提供了异常处理机制。

多线程 C++ 没有内置的多线程机制，需调用操作系统的多线程功能来进行多线程程序设计；Java 提供了多线程支持。

网络编程 Java 具有丰富的网络编程库。

编译和解释并存 由编译器将 Java 源程序编译成字节码文件，再由运行系统解释执行字节码文件（解释器将字节码再翻译成二进制码运行）。

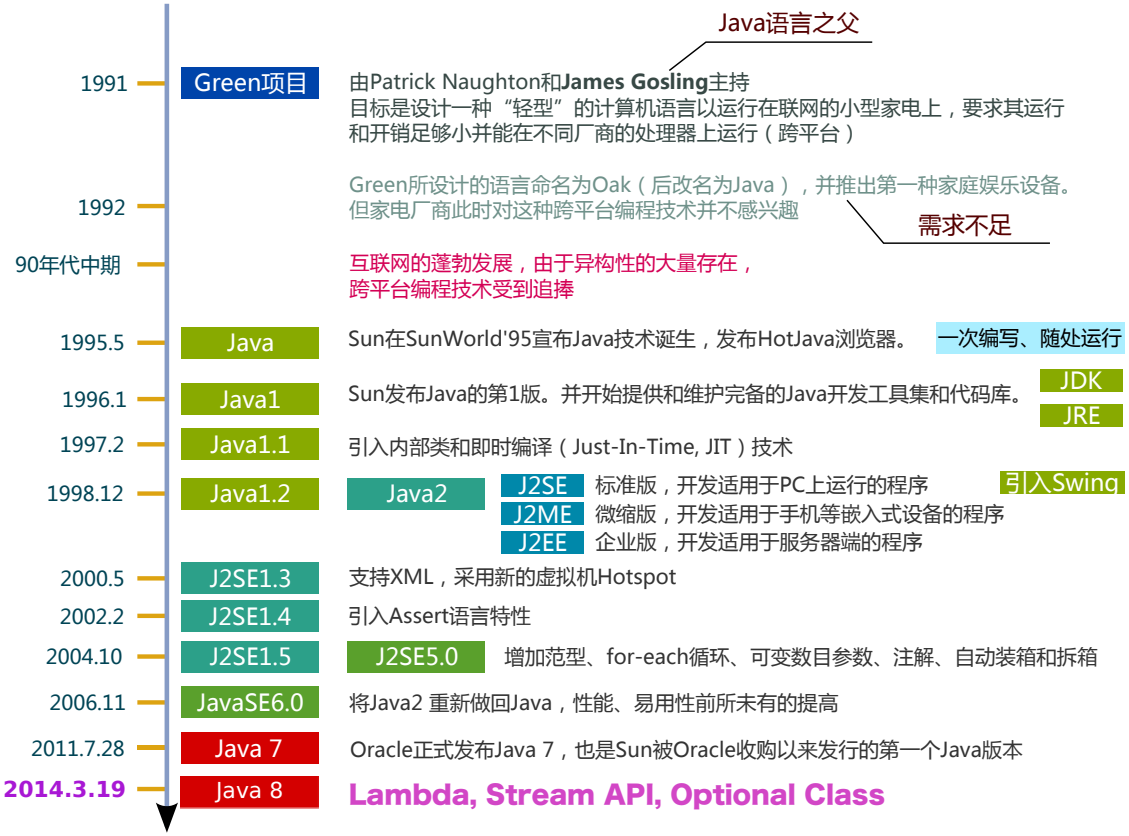

图 2.1: Java 版本迭代

2.2 Java 平台核心机制

Java 技术栈如图2.2所示，程序的编译运行过程如图2.3所示。需要了解以下几个核心概念：

- Java 虚拟机
- 垃圾回收机制
- Java 运行时环境（Java Runtime Environment, JRE）
- JIT, Just-In-Time 传统解释器的解释执行是转换一条，运行完后就将其扔掉；JIT 会自动检测指令的运行情况，并将使用频率高（如循环运行）的指令解释后保存下来，下次调用时就无需再解释（相当于局部的编译执行），显著提高了 Java 的运行效率。

2.3 Java 开发环境

构建 Java 开发环境，需要首先获取和安装 Java 开发工具集，可以从 Oracle 官方网站链接 http://www.oracle.com/technetwork/java/javase/downloads/index.html 获取。下载完成后解压放入合适的磁盘目录下。

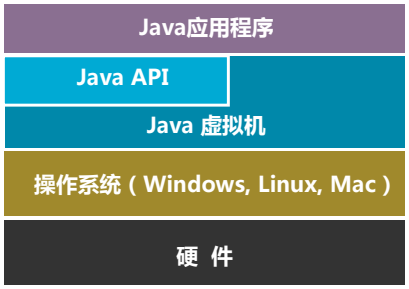

图 2.2: Java 技术栈

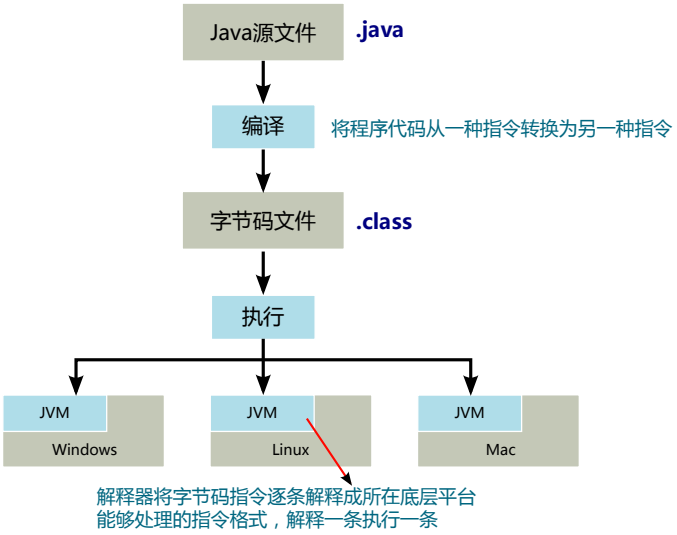

图 2.3: Java 程序编译运行过程

对于 Windows 操作系统，可以采用以下路径：

```
D:\Program Files\Java
```

对于 Linux 操作系统，可以采用以下路径：

```
/opt/jdk1.8.0_172
```

接下来进行环境变量配置，以 Windows 操作系统为例：

变量名 Path

变量值 D:\Program Files\Java\jdk1.8.0_172\bin

配置完成后，可以看到 JDK 目录中包含以下子目录和文件：

```
bin   COPYRIGHT db include javafx-src.zip
jre   lib   LICENSE man README.html release src.zip
THIRDPARTYLICENSEREADME-JAVAFX.txt THIRDPARTYLICENSEREADME.txt
```

对子目录的功能简要描述如下：

bin Java 开发工具，包括编译器、虚拟机、调试器、反编译器等；

jre Java 运行时，包括 Java 虚拟机、类库和其他资源文件；

lib 类库和所需支持性文件；

include 用于调试本地方法（底层平台）的 C++ 头文件；

src.zip 类库的源代码；

db Java DB 数据库，JDK6.0 新增项目，一种纯 Java 的关系型数据库。

2.4 Java 开发工具

业界普遍采用 Eclipse 或 IntelliJ IDEA 等集成开发环境进行 Java 大型工程开发，当然也可以采用文本编程工具 Vim 或 Emacs 等进行 Java 小型程序的开发。

本课程采用 Eclipse 作为首选集成开发环境。

2.5 Java 基本开发流程

本部分使用文本编程工具编写一个简单的 Java Hello World 程序，演示 Java 的基本开发和代码编译运行流程。首先，我们需要使用文本编程工具编写一个 Java 源文件 HelloWorld.java，文件命名必须与类名相同。

```
public class HelloWorld {
    public static void main(String[] args) {
  System.out.println(”Hi, Java!”);
    }
}
```

然后，使用 javac 工具将源文件编译为字节码文件，编译完成后，我们可以看到生成了 HelloWord.class 这一字节码文件。

```
> javac HelloWorld.java && ls
HelloWorld.class HelloWorld.java
```

接下来，我们使用 java 工具运行该程序，在终端正确打印了“Hi, Java”字符串。

```
> java HelloWorld
Hi, Java!
```

说明

编写 Java 应用程序需掌握的几条规则如下：

1. Java 语言拼写是大小写敏感的（Case-Sensitive）；
2. 一个源文件中可以定义多个 Java 类，但其中最多只能有一个类被定义为 Public 类；
3. 如果源文件中包含了 public 类，则源文件必须和该 public 类同名；
4. 一个源文件包含多个 Java 类时，编译后会生成多个字节码文件，即每个类都会生成一个单独的“.class”文件，且文件名与类名相同。

✠ 3 ✠ Java 语言基础与流程控制

基本信息

课程名称： Java 应用与开发

授课教师： 王晓东

授课时间： 第一周

参考教材： 本课程参考教材及资料如下：

- 陈国君主编，Java 程序设计基础（第 5 版），清华大学出版社，2015.5
- Bruce Eckel, Thinking in Java (3rd)

教学目标

1. Java 语言基础包括：数据类型、常量和变量、关键字与标识符、运算符与表达式、从键盘输入数据。
2. Java 流程控制包括：语句和复合语句、分支结构（选择结构）、循环结构、跳转语句。

授课方式

理论课： 多媒体教学、程序演示

实验课： 上机编程

教学内容

3.1 Java 语言基础

3.1.1 数据类型

Java 数据类型分为两大类：基本数据类型和引用数据类型。基本数据类型是由程序设计语言系统所定义、不可再划分的数据类型。所占内存大小固定，与软硬件环境无关，在内存中存放的是数据值本身。Java 的基本数据类型包括：整型（byte、short、int、long）、浮点型（float、double）、逻辑型（boolean）和字符型（char）。引用数据类型（复合数据类型）在内存中存放的是指向该数据的地址，不是数据值本身。引用数据类型包括类、数组、接口等。

数据类型的基本要素包括：

- 数据的性质（数据结构）
- 数据的取值范围（字节大小）
- 数据的存储方式
- 参与的运算

整型

Java 整型类型的数据位数及取值范围如表3.1所示。

表 3.1: 整型数据类型

类型	数据位数	取值范围
byte（字节型）	8	$-128 \sim 127$，即 $-2^7 \sim 2^7 - 1$
short（短整型）	16	$-32768 \sim 32767$，即 $-2^{15} \sim 2^{15} - 1$
int（整型）（默认）	32	$-2147483648 \sim 2147483647$，即 $-2^{31} \sim 2^{31} - 1$
long（长整型）（l 或 L）	64	$-2^{63} \sim 2^{63} - 1$

浮点型

Java 浮点型类型的数据位数及取值范围如表3.2所示。

表 3.2: 整型数据类型

类型	数据位数	取值范围
float（单精度）（f 或 F）	32	$1.4E-45 \sim 3.4E+38$
double（双精度）（默认）	64	$4.9E-324 \sim 1.8E+308$

逻辑型

逻辑型又称为布尔型（boolean），布尔型数据类型的特性如下：

- 布尔型数据只有 true（真）和 false（假）两个取值。
- 布尔型数据存储占 1 个字节，默认取值为 false。
- 布尔型数据 true 和 false 不能转换成数字表示形式。

字符型

- 字符型数据类型用来存储单个字符，采用的是 Unicode 字符集编码方案[1]。
- 字符声明用单引号表示单个字符。
- 字符型数据可以转化为整型。

Code: 字符数据类型示例

```
public class CharDemo {
  public static void main(String[] args) {
    char a = 'J';
    char b='Java'; //会报错
  }
}
```

3.1.2 数据类型转换

数值型不同类型数据的转换

数值型不同类型数据之间的转换，**自动类型转换**需要符合以下条件：

1. 转换前的数据类型与转换后的类型兼容。
2. 转换后的数据类型的表示范围比转换前的类型大。
3. 条件 2 说明不同类型的数据进行运算时，需先转换为同一类型，然后进行运算。转换从“短”到“长”的优先关系为：
   byte → short → char → int → long → float → double

如果要将较长的数据转换成较短的数据时（不安全）就要进行**强制类型转换**，格式如下：

```
(预转换的数据类型) 变量名;
```

[1]建议搜索理解什么是字符集和字符编码规则。

字符串型数据与数值型数据相互转换

Code: 字符串数据转换为数值型数据示例

```
String myNumber = "1234.56";
float myFloat = Float.parseFloat(MyNumber);
```

字符串可用加号“+”来实现连接操作。若其中某个操作数不是字符串，该操作在连接之前会自动将其转换成字符串。所以可用加号来实现自动的转换。

Code: 数值型数据转换成字符串数据示例

```
int myInt = 1234;                    //定义整形变量MyInt
String myString = "" + MyInt;  //将整型数据转换成了字符串
```

3.1.3 常量和变量

常量

整型常量 八进制、十六进制、十进制长整型后需要加 l 或 L。

浮点型常量 单精度后加 f 或 F，双精度后加 d 或 D 可省略。

逻辑型常量 true 或者 false。

字符型常量 单引号。

字符串常量 双引号。

Code: 常量的声明

```
final int MAX = 10;
final float PI =3.14f;
```

变量

变量的属性包括变量名、类型、值和地址。Java 语言程序中可以随时定义变量，不必集中在执行语句之前。

Code: 变量声明、初始化和赋值

```
int i, j = 0;
i = 8;
float k;
k = 3.6f;
```

表 3.3: Java 语言的关键字（保留字）

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

3.1.4 关键字与标识符

Java 的关键字（Java 保留字）如表3.3所示。

标识符是用来表示变量名、类名、方法名、数组名和文件名的有效字符序列。Java 语言对标识符的规定如下：

- 可以由字母、数字、下划线 (_)、美元符号 ($) 组合而成。
- 必须以字母、下划线或美元符号开头，不能以数字开头。
- 关键字不能当标识符使用。
- 区分大小写。

建议遵循驼峰命名，类名首字母大写，变量、方法及对象首字母小写的编码习惯。

3.1.5 运算符与表达式

按照运算符功能来分，Java 基本的运算符包括以下几类：

算术运算符	+、-、*、/、%、++、--
关系运算符	>、<、>=、<=、==、!=
逻辑运算符	!、&&、||、&、|、^
位运算符	>>、<<、>>>、&、|、^、~
赋值运算符	=、扩展赋值运算符，如+=、/=等
条件运算符	? :
其他运算符	包括分量运算符 . 、下标运算符[ ]、实例运算符 instanceof、内存分配运算符 new、强制类型转换运算符(类型)、方法调用运算符()等

图 3.1: Java 运算符

3.1.6 从键盘获得输入

由键盘输入的数据，不管是文字还是数字，Java 皆视为**字符串**，若是要由键盘输入获得数字则必须再经过类型转换。

Code: 获得键盘输入字符串并转换为数字

```
import java.io.*;
public class MyClass {
  public static void main(String[] args) throws IOException {
    int num1, num2;
    String str1, str2;
    InputStreamReader in;
    in = new InputStreamReader(System.in);
    BufferedReader buf;
    buf = new BufferedReader(in);
    System.out.print("请输入第一个数：");
    str1 = buf.readLine();          //将输入的内容赋值给字符串变量 str1
    num1 = Integer.parseInt(str1); //将 str1 转成 int 类型后赋给 num1
    System.out.print("请输入第二个数：");
    str2 = buf.readLine();          //将输入的内容赋值给字符串变量 str2
    num2 = Integer.parseInt(str2); //将 str2 转成 int 类型后赋给 num2
    System.out.println(num1 + " * " + num2 + " = " + (num1 * num2));
  }
}
```

为了简化输入操作，从 JavaSE 5 版本开始在 java.util 类库中新增了一个类专门用于输入操作的类**Scanner**，可以使用该类输入一个对象。

Code: 使用 Scanner 获得键盘输入并转换为特定数据类型

```
import java.util.*;
public class MyClass {
  public static void main(String[] args)
  {
    Scanner reader = new Scanner(System.in);
    double num;
    num = reader.nextDouble(); //按照 double 类型读取键盘输入
    ...
  }
}
```

Scanner 对象其他可用的数据读取方法包括：nextByte()、nextDouble()、nextFloat()、nextInt()、nextLong()、nextShort()、next()、nextLine()。

3.2 Java 流程控制

3.2.1 语句与复合语句

- Java 语言中语句可以是以分号“;”结尾的简单语句，也可以是用一对花括号“{}”括起来的复合语句。

- Java 中的注释形式：
  - 单行注释：//
  - 多行注释：/* */
  - 文件注释：/** */

3.2.2 分支结构

if 分支结构 1

if 双路条件结构

```
if (条件表达式){
  语句序列 1
}else{
  语句序列 2
}
```

if 单路条件结构

```
if (条件表达式){
 语句序列
}
```

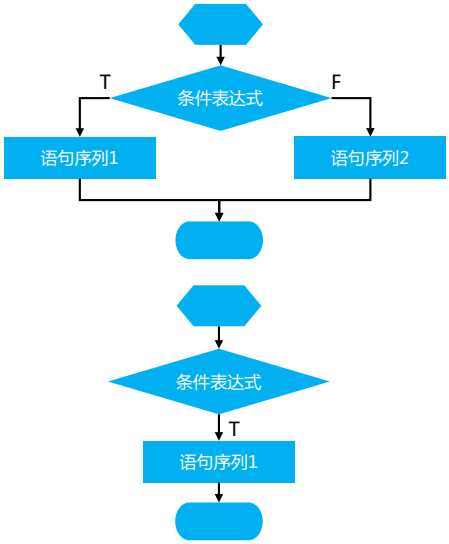

图 3.2: if 分支结构 1

if 分支结构 2

switch 分支结构

说明

在 Java 1.7 版本之后，switch 里表达式的类型可以为 String。

3.2.3 循环结构

while 循环

```
1  while(conditional expression) {
2    statements goes here ...
3  }
```

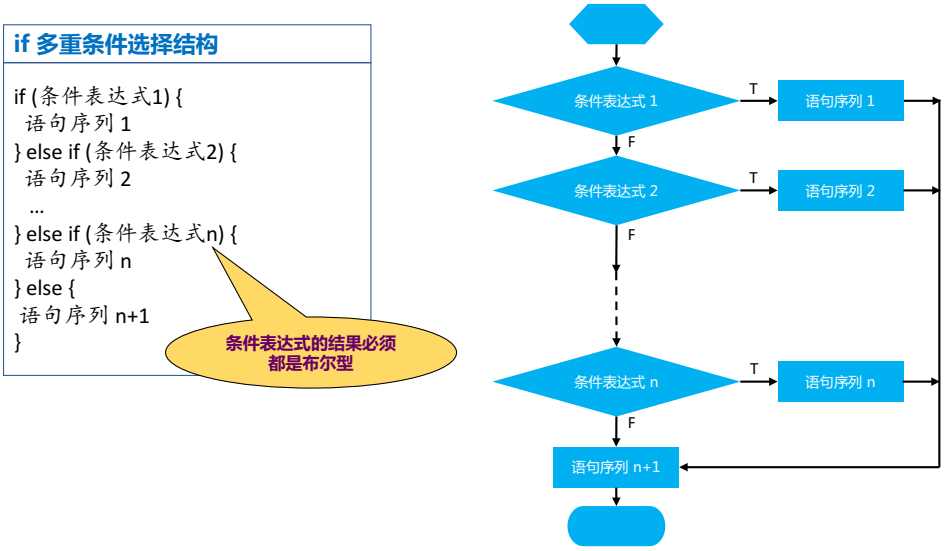

图 3.3: if 分支结构 2

do-while 循环

```
do {
  statements goes here ...
}
while(conditional expression);
```

for 循环 1

```
int [] integers = {1, 2, 3, 4};

for (int j = 0; j < integers.length; j++) {
  int i = integers[j];
  System.out.println(i);
}
```

for 循环 2

```
int [] integers = {1, 2, 3, 4};

for (int i : integers) {
  System.out.println(i);
}
```

循环中的跳转

break 语句 使程序的流程从一个语句块（switch 或循环结构）内跳出。

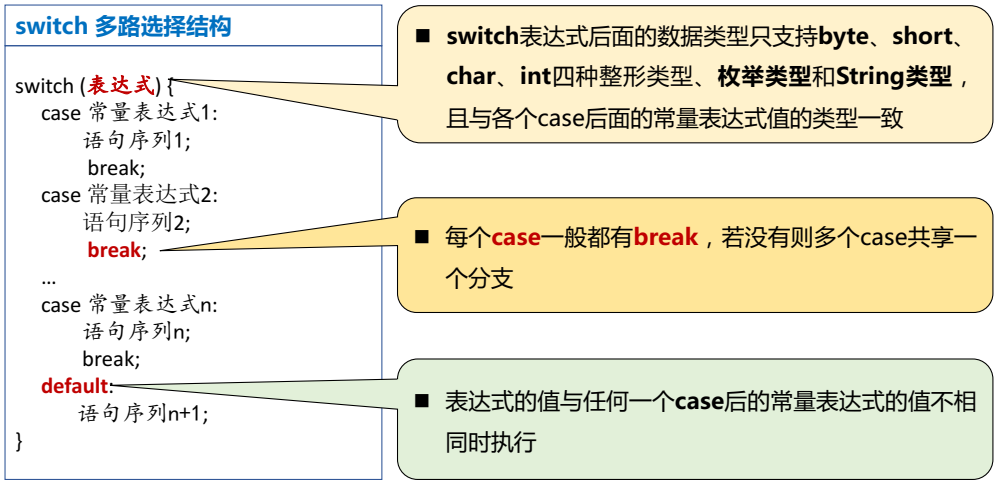

图 3.4: switch 分支结构

continue 语句 终止当前这一轮（次）的循环，进入下一轮（次）循环。

return 语句 用来使程序从方法（函数）中返回，可返回一个值。

3.3 课后习题

3.3.1 简答题

1. Java 语言定义类哪些基本数据类型？其存储结构分别是什么样的？
2. 自动类型转换的前提是什么？转换时的优先级顺序如何？
3. 数字字符串转换为数值类型数据时，可以使用的方法有哪些？

3.3.2 小编程

1. 编写程序，从键盘输入一个浮点数，然后将该浮点数的整数部分输出。
2. 编写程序，从键盘输入 2 个整数，然后计算它们相除后得到的结果并输出，注意排除 0 除问题。

实验设计

实验名称： Eclipse 集成开发环境配置及 Java 语言基础编程练习

上机时间： 第一周

实验手册： 无（参照实验内容完成）

实验内容： 本次实验需要完成以下内容：

1. 使用文本编辑器完成 Java Hello World 程序编写，使用 javac 和 java 编译运行该程序；
2. 熟悉 Eclipse 集成开发环境，学习创建 Java 工程，使用 Maven 创建 Java 工程；
3. 根据授课幻灯片和讲义，尝试实现其中所有的示例代码。

实验要求： 本次实验不需要提交实验报告。