第四章：流程控制

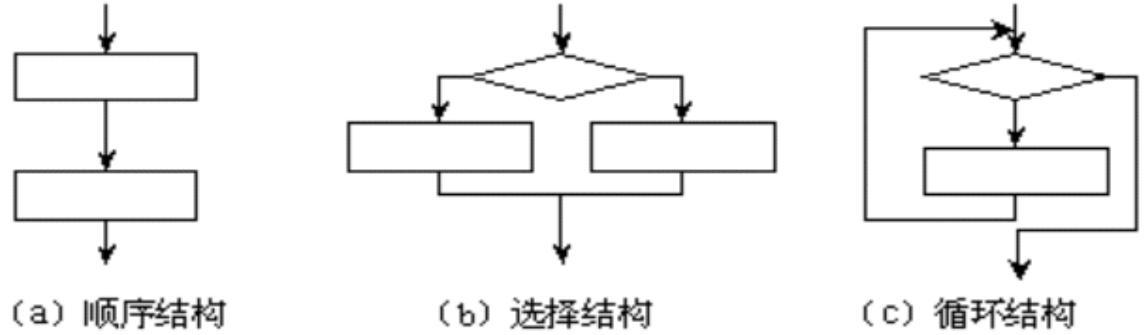
教学内容：

1. 顺序结构
2. 分支语句
3. 循环语句
4. 特殊循环语句
5. 注释

引言：

- Java 虽然是面向对象的语言，但在局部的语句块内部仍然需要借助结构化的基本流程结构（顺序结构、分支结构、循环结构）来组织语句，完成相应的逻辑功能。

- 三种基本流程控制结构示意图分别如下图所示：



结构化程序设计的三种基本流程控制结构

一、 分支语句

a) 条件分支语句

i. if 语句

```
if(返回boolean值的条件表达式){
    语句组;
}
```

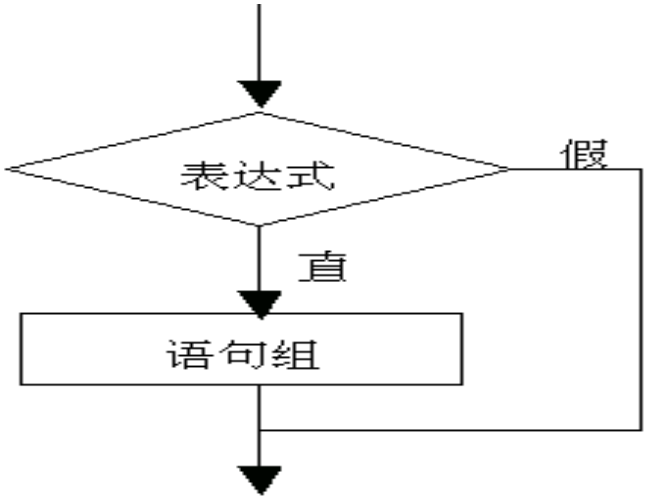


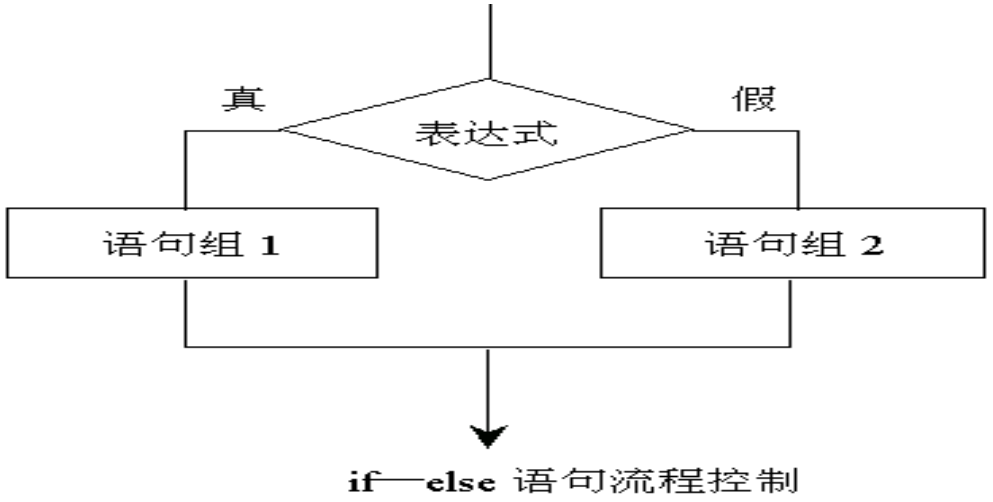
图 3-3 if 语句流程控制

案例：

```
Scanner sc = new Scanner(System.in);
int age = sc.nextInt();
if(age<10){
    System.out.println("发你一个帮帮糖");
}
    System.out.println("请进入游乐场");
```

ii. if-else 语句

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



if—else 语句流程控制

案例：

```
Scanner sc = new Scanner(System.in);
float height = sc.nextFloat();

if(height<1.3f){
      System.out.println("请购买儿童票");
}else {
      System.out.println("请购买成人票");
}
      System.out.println("请进场");
```

iii. if-else if ... -else

案例：百分制的评分：90-100 分以上优秀；80-90 分以上良好；60-80 分以上及格；其它的淘汰。

```
Scanner sc = new Scanner(System.in);
int score = sc.nextInt();

if(score>90){
     System.out.println("优秀");
}else if (score>80) {
     System.out.println("良好");
}else if (score>60) {
     System.out.println("及格");
}else {
     System.out.println("淘汰");
}
```

iv. Switch 语句

switch 语句（又称开关语句）是和 case 语句一起使用的，其功能是根据某个表达式的值在多个 case 引导的多个分支语句中选择一个来执行。它的一般格式如下：

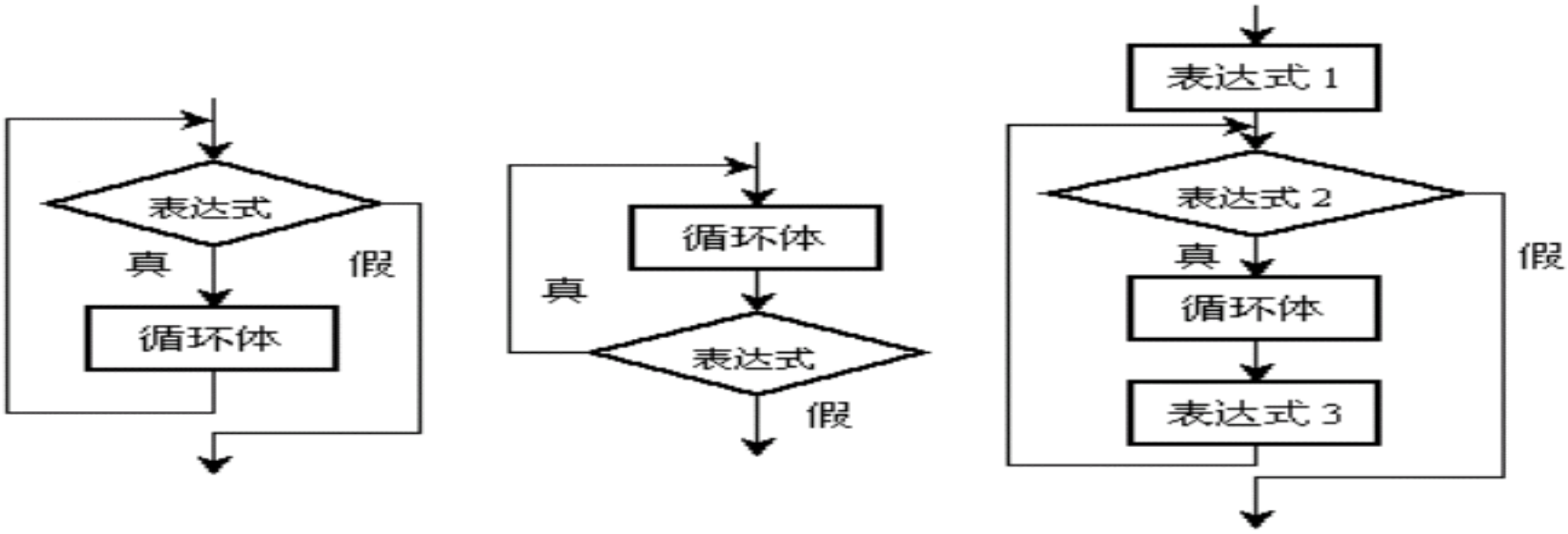
```
switch (key) {
case value:
     语句块1;
     break;
     ……
default:
     break;
}
```

案例：

```
int i = 3;
switch (i) {
case 1:
     System.out.println(1);
     break;
case 2:
     System.out.println(2);
     break;
default:
     System.out.println("default");
     break;
}
```

- default 语句是可选的，它接受除上面接受值的其他值，通俗的讲，就是谁也不要的都归它。
- case 后面可以跟多个语句，这些语句可以不用大括号括起来。
- **switch 后面括号中表达式的值必须是符合 byte，char，short，int 类型的常量表达式，jdk1.7 以后可以使用 String，而不能用浮点类型或 long 类型。**
- 一个 switch 语句可以代替多个 if–else 语句组成的分支结构，而 switch 语句从思路上显得更清晰

二、 循环结构



（a） while 语句　　（b） do-while 语句　　（c） for 语句

图 3-4　三种循环语句结构

1. while 循环

语法：

```
while （条件表达式） {
    循环体语句；
}
```

在循环刚开始时，会计算一次“条件表达式”的值。当条件为假时，将不执行循环体，直接跳转到循环体外，执行循环体外的后续语句；当条件为真时，便执行循环体。每执行完一次循环体，都会重新计算一次条件表达式，当条件为真时，便继续执行循环体，直到条件为假才结束循环。

案例：循环输出 1-100；

```
int i =1;
while(i<=100){
    System.out.println(i);
    i++;
}
```

2. do while 循环

语法：

```
do {
    循环体语句；
} while （条件表达式）;
```

do-while 循环与 while 循环的不同在于：它先执行循环中的语句，然后再判断条件是否为真，如果为真则继续循环；如果为假，则终止循环。因此，do-while 循环至少要执行一次循环语句。

注意：while()后一定要加“；”

案例：循环输出 1=100；

```
int i =1;
do {
    System.out.println(i);
    i++;
} while (i<=100);
```

PS：while 循环和 do-while 循环的区别

while 循环只有当条件表达式为 true 的时候才会进入循环体，do-while 循环首先执行一次循环体中的内容，无论条件表达式是否为 true；

3. for 循环

语法：

```
for (表达式1;表达式2;表达式3) {
    循环体语句；
}
```

表达式 1 一般是一个赋值语句，它用来给循环控制变量赋初值；表达式 2 是一个布尔类型的表达式，它决定什么时候退出循环；表达式 3 一般用来修改循环增量表达式。这三个部分之间用 “；” 分开。

案例：循环输出 1-100；

```
for (int i = 1; i <= 100; i++) {
    System.out.println(i);
}
```

练习：请问 i 输出是多少？

```
int i = 1;
for (; i <= 100; i++) {
}
System.out.println(i);
```

4. 增强 for 循环(JDK5.0 之后的新特性)

可以用来处理数组中的每个元素(其它类型的集合也可以)，而不需要指定下标；

```
for (类型 变量 : 集合) {

}
```

类型：值集合中保存的数据的数据类型；

变量：表示集合中的每一个元素；

集合：需要处理的集合对象；

5. 跳转语句

   break、continue

   在 Java 语言中，可用 break 和 continue 控制循环的流程。其中，break 用于强行退出循环，不执行循环中剩余的语句。而 continue 则停止执行当前的循环，开始新的循环。

6. 递归

   案例：1+2+3….+100;

```
public static int sum(int num){
    return num==1?1:num+sum(num-1);
}
```