Java

极夜酱

目录

Chapter 1 Java 简介

1.1 编程简介

1.1.1 编程简介

程序（program）是为了让计算机执行某些操作或者解决问题而编写的一系列有序指令的集合。由于计算机只能够识别二进制数字 0 和 1，因此需要使用特殊的编程语言来描述如何解决问题过程和方法。

算法（algorithm）是可完成特定任务的一系列步骤，算法的计算过程定义明确，通过一些值作为输入并产生一些值作为输出。

流程图（flow chart）是算法的一种图形化表示方式，使用一组预定义的符号来说明如何执行特定任务。

- 圆角矩形：开始和结束
- 矩形：数据处理
- 平行四边形：输入/输出
- 菱形：分支判断条件
- 流程线：步骤

1.1.2 编程语言（Programming Language）

编程语言主要分为面向机器、面向过程和面向对象三类。C 语言是面向过程的语言，常用于操作系统、嵌入式系统、驱动程序、图形引擎、图像处理、声音效果等。

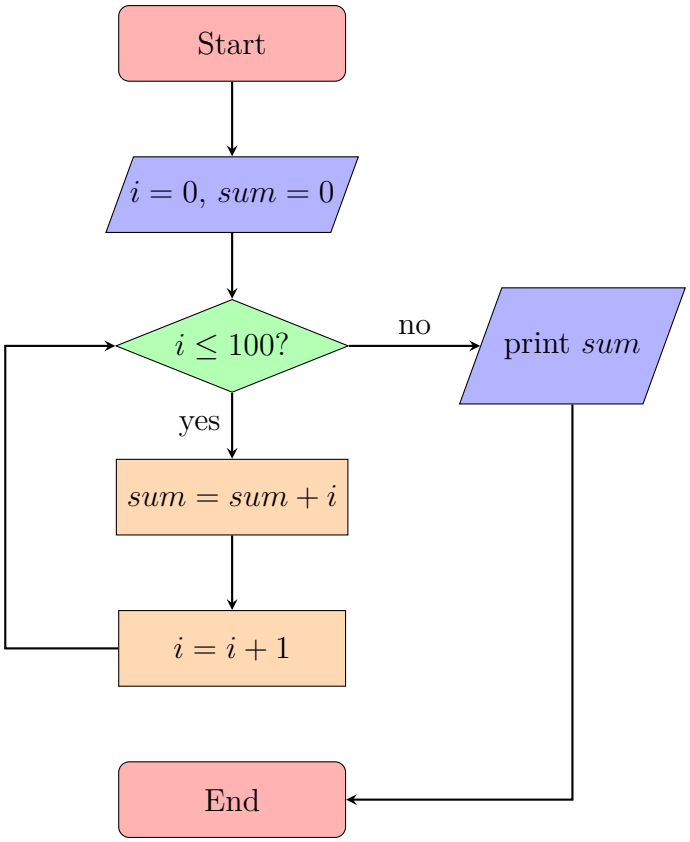

图 1.1: 计算 $\sum_{i=1}^{100} i$ 的流程图

Java 是面向对象语言，吸收了 C/C++ 的优点，并摒弃了难以理解的多继承、指针等概念。Java 可以编写桌面应用程序、Web 应用程序、分布式系统和嵌入式系统应用程序等。

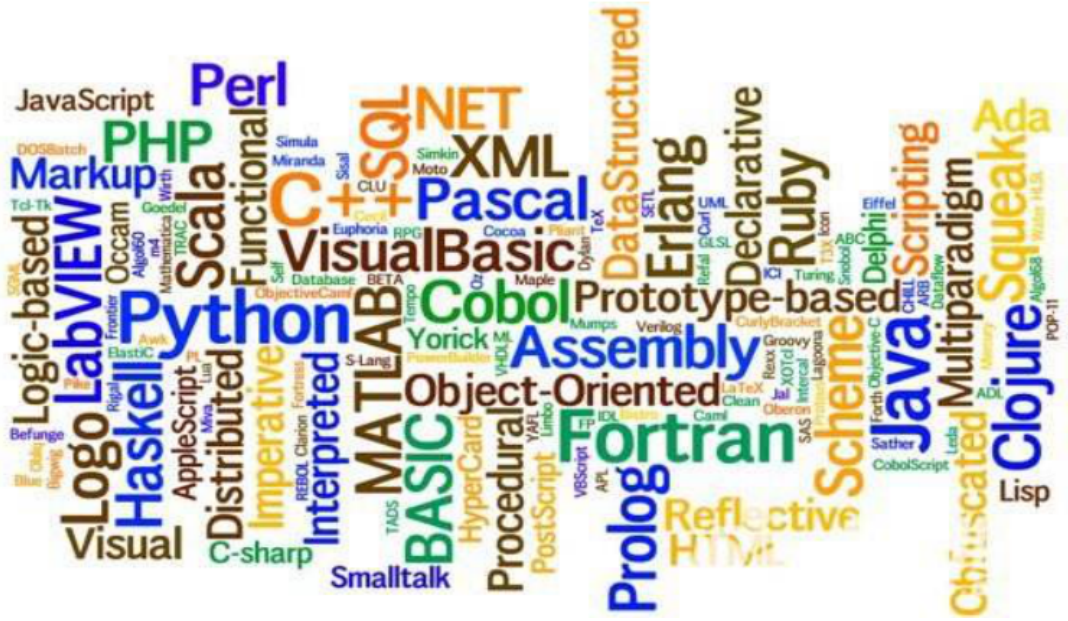

图 1.2: 常见编程语言

1.2 Hello World!

1.2.1 Hello World!

Hello World!

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello World!");
4     }
5 }
```

运行结果

Hello World!

第一行语句中的 public 为访问修饰符，一共有三种：public、private、protected。第一行中 class 表示一个类，类名需要与文件名相同。

第二行中的 `main()`是程序的入口。

第三行的语句 `System.out.println()`的作用是在屏幕上输出 “Hello World” 这个字符串。【;】表示语句结束，注意不要使用中文的分号。

1.2.2 字节码文件

Java 编译器（compiler）的作用是将 Java 源程序编译成中间代码字节码文件。字节码文件是一种和任何具体机器环境及操作系统环境无关的中间代码。Java 程序不能直接运行在现有的操作系统，必须运行在 Java 虚拟机上。Java 的特点是一次编写，到处运行。

1.3 Error or Warning?

1.3.1 Error / Warning

在编写程序的过程中，错误是不可避免的，错误主要能够分为以下三种类别：

1. 语法错误（syntax error）：程序的语法不合符编程语言的要求，编译器会反馈报错信息。
2. 逻辑错误（logical error）：人类在编程过程中的逻辑错误，无法被编译器所检测。
3. 运行时错误（runtime error）例如除以 0、数组越界、指针越界、使用已经释放的空间、栈溢出等情况，可以被编译器发现。

1.4 注释

1.4.1 注释（Comment）

在编程中加入注释可以增加程序的可读性和可维护性，编译器不会对注释的部分进行编译。

Java 中注释分为两类：

1. 单行注释：将一行内【//】之后的内容视为注释。
2. 多行注释：以【/*】开始，【*/】结束，中间的内容视为注释。

注释

```
/*
    这个程序在屏幕上输出Hello World
*/
public class Comment {
    // 主函数
    public static void main(String[] args) {
        System.out.println("Hello World!");     // 输出
    }
}
```

运行结果

Hello World!

1.5 不同语言的 Hello World

1.5.1 编程语言对比

C

```c
#include <stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
```

C++

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World" << endl;
    return 0;
}
```

Python

```python
print("Hello World")
```

Chapter 2 数据类型

2.1 变量

2.1.1 变量（Variable）

Java 是一种强类型的语言，任何数据都有一个确定的类型。

变量是计算机中一块特定的内存空间，由一个或多个连续的字节组成，不同数据存入具有不同内存地址的空间，相互独立，通过变量名可以简单快速地找到在内存中存储的数据。

变量名需要符合以下的要求：

1. 由字母、数字和下划线组成，第一个字符必须为字母或下划线。
2. 不能包含除【_】以外的任何特殊字符，如【%】、【#】等。
3. 不可以使用保留字或关键字。
4. 准确、顾名思义，不要使用汉语拼音。

关键字是编程语言内置的一些名称，具有特殊的用处和意义。

abstract	do	implements	protected	throws
boolean	double	import	public	transient
break	else	instanceof	return	true
byte	extends	int	short	try
case	false	interface	static	void
catch	final	long	strict	volatile
char	finally	native	super	while
class				

表 2.1: 关键字

2.1.2 数据类型

Java 中变量主要有三大类型：

1. 整型
    - 字节型 byte
    - 短整型 short
    - 整型 int
    - 长整型 long
2. 浮点型
    - 单精度浮点型 float
    - 双精度浮点型 double
3. 字符型 char
4. 布尔型 boolean

数据类型	**位数**	**取值范围**
int	32	$-2^{31} \sim 2^{31}-1$
float	32	$-3.4E38 \sim 3.4E38$
double	64	$-1.7E308 \sim 1.7E308$
char	8	-128 127
boolean	1	true / false

表 2.2: 不同数据类型的取值范围

2.2 初始化

2.2.1 初始化（Initialization）

变量可以在定义时初始化，也可以在定义后初始化。

在编程中，【=】不是数学中的“等于”符号，而是表示“赋值”，即将【=】右边的值赋给左边的变量。

```
1 int n = 10;
2 double wage = 8232.56;
```

2.2.2 常量（Constant）

常量是一个固定值，在程序执行期间不会改变，即在定义后不可修改。常量可以是任何的基本数据类型，比如整数常量、浮点常量、字符常量。

常量

```
1 public class Contant {
2     public static void main(String[] args) {
3         final double PI = 3.14159;
4         PI = 4;
5     }
6 }
```

运行结果

The final local variable PI cannot be assigned.

2.3 算术运算符

2.3.1 四则运算

数学符号	Java 符号	含义
+	+	加法
−	-	减法
×	*	乘法
÷	/	除法
.....	%	取模

表 2.3: 四则运算

Java 中除法【/】的意义与数学中不同：

1. 当相除的两个运算数都为整型，则运算结果为两个数进行除法运算后的整数部分，例如 21 / 5 的结果为 4。

2. 如果两个运算数其中至少一个为浮点型，则运算结果为浮点型，如 21 / 5.0 的结果为 4.2。

取模（modulo）【%】表示求两个数相除之后的余数，如 22 % 3 的结果为 1；4 % 7 的结果为 4。

2.3.2 复合赋值运算符

运算符	描述
+=	a += b 等价于 a = a + b
-=	a -= b 等价于 a = a - b
*=	a *= b 等价于 a = a * b
/=	a /= b 等价于 a = a / b
%=	a %= b 等价于 a = a % b

表 2.4: 复合赋值运算符

2.4 输入输出函数

2.4.1 System.out.println()

System.out.println() 的功能是向屏幕输出指定格式的字符串内容。通过【+】运算符可以连接两个字符串。

字符串连接

```
1 public class StringConcatenation {
2     public static void main(String[] args) {
3         System.out.println("Hello" + "World");
4         System.out.println("Hello" + 2);
5         System.out.println("Hello" + 2 + 3);
6         System.out.println(2 + 3 + "Hello");
7     }
8 }
```

运行结果

HelloWorld

Hello2

Hello23

5Hello

2.4.2 转义字符

在一个字符串描述的过程中，有可能会有一些特殊字符的信息。

转义字符	描述
\\	表示一个反斜杠 \
\'	表示一个单引号 '
\"	表示一个双引号 "
\n	换行
\t	横向制表符

表 2.5: 转义字符

转义字符

```
1 public class EscapeCharacter {
2     public static void main(String[] args) {
3         System.out.print("全球最大同性交友网站\n");
4         System.out.println("\'https://github.com\'");
5     }
6 }
```

运行结果

全球最大同性交友网站

'https://github.com'

2.4.3 Scanner

通过 Scanner 类可以获取用户的输入，使用 Scanner 类需要导入 java.util.Scanner。根据实例化的 Scanner 类的对象，调用 next() 和 nextLine() 可以获取输入的字符串。

使用完 Scanner 类的对象后，需要关闭输入流。

计算长方形面积

```
1  import java.util.Scanner;
2
3  public class CircleArea {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          double length, width;
7          double area;
8
9          System.out.print("输入长度: ");
10         length = scanner.nextDouble();
11         System.out.print("输入宽度: ");
```

```
        width = scanner.nextDouble();
        area = length * width;

        System.out.println(String.format("面积 = %.2f", area));
        scanner.close();
    }
}
```

运行结果

输入长度：20

输入宽度：30

面积 = 600.00

计算圆面积

```
import java.util.Scanner;

public class CircleArea {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        final double PI = 3.14159;
        double r;
        double area;

        System.out.print("输入半径：");
        r = scanner.nextDouble();
        area = PI * r * r;

        System.out.println(String.format("面积 = %.2f", area));
        scanner.close();
    }
}
```

运行结果

输入半径：5

面积 = 78.539750

逆序三位数

```java
import java.util.Scanner;

public class Reverse {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num;
        int a, b, c;

        System.out.print("输入一个三位数：");
        num = scanner.nextInt();

        a = num / 100;
        b = num / 10 % 10;
        c = num % 10;

        System.out.println("逆序：" + (c*100 + b*10 + a));
        scanner.close();
    }
}
```

运行结果

输入一个正三位数：520

逆序：25

2.5 类型转换

2.5.1 类型转换

类型转换是把变量从一种类型转换为另一种数据类型。类型转换可以是隐式的，由编译器自动执行，也可以是显式的，通过使用强制类型转换运算符来指定。在有需要类型转换时都用上强制类型转换运算符是一种良好的编程习惯。

隐式类型转换

```
1 public class Implitcit {
2     public static void main(String[] args) {
3         int a = 1;
4         double b = a;
5         System.out.println("b = " + b);
6     }
7 }
```

运行结果

b = 2

显式类型转换

```
1 public class Explicit {
2     public static void main(String[] args) {
3         int sum = 821;
4         int num = 10;
5         double average = (double)sum / num;
6         System.out.println(String.format("average = %.2f", average));
7     }
8 }
```

运行结果

82.10

Chapter 3 判断

3.1 逻辑运算符

3.1.1 关系运算符

数学符号	关系运算符
$<$	<
$>$	>
$\leq$	<=
$\geq$	>=
$\neq$	!=
$=$	==

表 3.1: 关系运算符

3.1.2 逻辑运算符

Java 中逻辑运算符有三种：

1. 逻辑与 && (logical AND)：当多个条件同时为真，结果为真。

条件 1	条件 2	条件 1 && 条件 2
T	T	T
T	F	F
F	T	F
F	F	F

表 3.2: 逻辑与

2. 逻辑或 || (logical OR)：多个条件有一个为真时，结果为真。

条件 1	**条件 2**	**条件 1 || 条件 2**
T	T	T
T	F	T
F	T	T
F	F	F

表 3.3: 逻辑或

3. 逻辑非!（logical NOT）：条件为真时，结果为假；条件为假时，结果为真。

条件	**！条件**
T	F
F	T

表 3.4: 逻辑非

3.2 if

3.2.1 if

当 if 语句的条件为真时，进入花括号执行内部的代码；若条件为假，则跳过花括号执行后面的代码。

if 语句主要有以下几种形式：