chapter 2 语言基础

1. 标识符和关键字
   - 标识符

     > 用来起名的字符

     - 包括字母/数字/下划线/美元符号 `$`
     - 首字符不能是数字
     - 区分大小写
     - 不能是关键字以及 `true` / `false`
   - Java 命名规范
     - 类名
       首字母大写，第二个单词起首字母大写
     - 方法名，变量名，参数名
       首字母小写，第二个单词起首字母大写
     - `switch` / `case` 中 `default` 后要写 `break`
     - `if` 后语句体只有一个，也要写 `{}`

     > Google Java Style Guide

   - 关键字
     - 被 Java 语言占用的一些单词，不能作为标志符

     > Java 关键字表

类别	关键字	说明
访问控制	private	私有的
	protected	受保护的
	public	公共的
类、方法和变量修饰符	abstract	声明抽象
	class	类
	extends	扩展，继承
	final	终极，不可改变的
	implements	实现

	interface	接口
	native	本地
	new	新建，创建
	static	静态
	strictfp	严格，精准
	synchronized	线程，同步
	transient	短暂
	volatile	易失
程序控制语句	break	跳出循环
	continue	继续
	return	返回
	do	运行
	while	循环
	if	如果
	else	否则
	for	循环
	instanceof	实例
	switch	根据值选择执行
	case	定义一个值以供 switch 选择
	default	默认
错误处理	assert	断言表达式是否为真
	catch	捕捉异常
	finally	有没有异常都执行
	throw	抛出一个异常对象
	throws	声明一个异常可能被抛出
	try	捕获异常
包相关	import	引入
	package	包

基本类型	boolean	布尔型
	byte	字节型
	char	字符型
	double	双精度浮点
	float	单精度浮点
	int	整型
	long	长整型
	short	短整型
	null	空
变量引用	super	父类，超类
	this	本类
	void	无返回值
保留关键字	goto	不是关键字，但不能使用

2. 数据类型、直接量和变量
    - **数据类型**
        - 基本数据类型 `primitive data type`
            - **布尔类型** `boolean`
            - 基本数值类型
                - 定点类型
                    - 字节 `byte`
                    - 字符 `char`
                    - 短整型 `short`
                    - **整形** `int`
                    - 长整型 `long`
                - 浮点类型
                    - 单精度浮点数 `float`
                    - **双精度浮点数** `double`
        - 引用数据类型 `reference data type`
            - 类 `class`
            - 接口 `interface`
            - 枚举 `enum`
            - 数组 `array`
    - 基本类型 `primitive` 和装箱基本类型 `boxed primitive`
        - `Boolean` - `boolean`
        - `Byte` - `byte`
        - `Character` - `char`

- `Short` - `short`
- `Integer` - `int`
- `Long` - `long`
- `Float` - `float`
- `Double` - `double`
- 基本类型只有值；装箱基本类型有值和不同的引用
- 基本类型只有一个功能值；装箱基本类型有功能值和非功能值 `null`
- 基本类型比装箱基本类型更节省时间和空间
- 自动装箱 `autoboxing` JDK1.5
- 降低了使用装箱基本类型的繁琐性，但存在风险：装箱基本类型的 `==` 和 `!=` 几乎总是错误的
- 自动拆箱 `auto-unboxing` JDK1.5
- 混合使用装箱基本类型和基本类型时，自动拆箱

```
public static void main(String[] args) {
    Long sum = 0L; // Hideously slow! |'hɪdɪəsli|
    for(int i = 0; i < Integer.MAX_VAL; i++) {
        sum += i;
    }
    System.out.println(sum);
}
```

- **基本类型优于装箱基本类型**

◦ 浮点数的精度损失

> They perform `binary floating-point arithmetic`, which was carefully designed to furnish accurate `approximations` quickly over a broad range of magnitudes.
>
> The float and double types are particularly ill-suited for monetary calculations.

- `java.math.BigDecimal`
- `int` or `long`

◦ 变量
- 类型
- 名字
- 值
- 存储单元

◦ 直接量
- 基本数据类型直接量
  - 整形默认为 `int`

```
long l = 2147483648L;
```

整形的二进制 `0b` 八进制 `0` 十六进制 `0x`
  - 浮点型默认为 `double`

```
float f = 1.23f;
```

- `char` 的直接量
  - 1 整数 `[0,65535]`
  - ‘a’ 单个字符
  - ‘\123’ 3位八进制字符 `\000 ~ \377`
  - ‘\u4E00’ 4位十六进制字符，unicode 字符
  - ‘\t’ 转义字符串 `escape sequence`

```
\t    tab
\b  backspace
\n  newline
\r  carriage return
\f  form feed
\'  single quote character
\"  a double quote character
\\  a backslash character
```

  - 字符串直接量
  - `null` 一切引用数据类型的直接量

3. 运算符
   - 算术运算符 `Arithmetic operators`
     - +

> + 在数值与字符串中的运算规则

```
+ 的运算顺序是从左至右，其他类型与字符串的 + 为字符串拼接
```

     - -
     - *
     - / 整数除法，截去余数
     - ++ 自增运算，注意先后的区别
     - -- 自减运算，注意先后的区别
     - % 模运算，求余，符号只与被除数有关
   - 关系运算符 `Relational operators`
     - >
     - <
     - >=
     - <=
     - == 可用于任意类型比较
     - != 可用于任意类型比较
   - 布尔逻辑运算符 `Logical operators`
     - & 逻辑与
     - | 逻辑或

- ^ 异或
- ! 非
- && 条件与
- || 条件或

> 短路规则

> &&
> ||
> 第一个表达式即可确定运算结果时，不再判断第二个表达式
> 使用短路规则与否，可能副作用不同

- 位运算符 `Bitwise operators`
  - &
  - |
  - ^
  - ~ `非 取反` `~x equals (-x) - 1`
  - <<
  - >>> `无符号右移`
    - 补零 `0 fill`
  - >> `有符号右移`
    - 正数补零 positive `0 fill`
    - 复数补一 negative `1 fill`

> 二进制正负转换：取反加一

- 赋值类运算符 `Assignment operators`
  - =
  - +=
  - -=
  - *=
  - /=
  - %=
  - &= 针对布尔值或定点类型值
  - |= 针对布尔值或定点类型值
  - ^= 针对布尔值或定点类型值
  - <<=
  - >>=
  - >>>=
- 条件运算符 `Conditional operator`
  - (condition)?(true_value):(false_value) 三目运算符
- 其他运算符 `Misc operators`
  - `(类型)` 强制类型转换 `cast`
    - 强制类型转换有风险
    - `Java` 合法类型转换

> `-->` 无信息丢失
>
> `-->` 可能有精度损失
>
> `-->` 或 `-->` 的逆向需要强制转换

```
char --> int
byte --> short --> int --> long
int --> double
int --> float
long --> float
long --> double
```

- `.` 引用
- `[]` 数组
- `()` 改变运算优先级，推荐使用

> Java 的运算优先级

> 非 > 算术 > 关系 > 与 > 或 > 赋值
>
> - `instanceof`
> - `new`

4. 控制结构
    - 程序运行的控制结构
        - 顺序结构
        - 选择结构
            - `if`
            - `if` / `else` (`if`)

> 注意 `else` 有无的区别

            - `switch` / `case`

> 可用于 `switch` 的变量类型

```
byte char int short enum String(JDK 1.7+)
```

        - 循环结构
            - `for`
            - `for-each` - `enhanced for loop`
            - `while`
            - `do` / `while`

- `break` 跳出当前这一个循环
- `continue` 跳出当前这一次循环