第 5 章

CHAPTER 5

Java 编码规范

俗话说：“没有规矩不成方圆”。编程工作往往都是一个团队协同进行，因而一致的编码规范非常有必要，这样写成的代码便于团队中的其他人员阅读，也便于编写者自己以后阅读。

微课视频

5.1 命名规范

程序代码中到处都是标识符，因此取一个一致并且符合规范的名字非常重要。

命名方法很多，但是比较有名的且被广泛接受的命名方法包括如下两种：

- 匈牙利命名，一般只是命名变量，原则是：变量名＝类型前缀＋描述，如 bFoo 表示布尔类型变量，pFoo 表示指针类型变量。匈牙利命名还是有一定争议的，在 Java 编码规范中基本不被采用。
- 驼峰命名(Camel-Case)，又称“骆驼命名法”，是指混合使用大小写字母来命名。驼峰命名又分为小驼峰法和大驼峰法。小驼峰法就是第一个单词是全部小写，后面的单词首字母大写，如 myRoomCount；大驼峰法是第一个单词的首字母也大写，如 ClassRoom。

除了包和常量外，Java 编码规范命名方法采用驼峰法，分类说明如下：

- 包名：包名是全小写字母，中间可以由点分隔开。作为命名空间，包名应该具有唯一性，推荐采用公司或组织域名的倒置，如 com.apple.quicktime.v2。但 Java 核心库包名不采用域名的倒置命名，如 java.awt.event。
- 类和接口名：采用大驼峰法，如 SplitViewController。
- 文件名：采用大驼峰法，如 BlockOperation.java。
- 变量：采用小驼峰法，如 studentNumber。
- 常量名：全大写，如果是由多个单词构成，可以用下画线隔开，如 YEAR 和 WEEK_ OF_MONTH。
- 方法名：采用小驼峰法，如 balanceAccount、isButtonPressed 等。

命名规范示例代码如下：

```
package com.zhijieketang;

public class Date extends java.util.Date {

    private static final int DEFAULT_CAPACITY = 10;

    private int size;
```

```
    public static Date valueOf(String s) {

        final int YEAR_LENGTH = 4;
        final int MONTH_LENGTH = 2;

        int firstDash;
        int secondDash;
        …
    }

    public String toString () {
        int year = super.getYear() + 1900;
        int month = super.getMonth() + 1;
        int day = super.getDate();
        …
    }
}
```

微课视频

5.2 注释规范

Java 中注释的语法有三种：单行注释(//)、多行注释(/ * … * /)和文档注释(/ ** … * /)。本节通过文件注释、文档注释、代码注释来介绍如何规范使用这些注释语法。

5.2.1 文件注释

文件注释就是在每个文件开头添加注释。文件注释通常包括如下信息：版权信息、文件名、所在模块、作者信息、历史版本信息、文件内容和作用等。

示例代码如下：

```
/*
 * 版权所有 2015 北京智捷东方科技有限公司
 * 许可信息查看 LICENSE.txt 文件
 * 描述:
 *   实现日期基本功能
 * 历史版本:
 *   2015-7-22: 创建 关东升
 *   2015-8-20: 添加 socket 库
 *   2015-8-22: 添加 math 库
 */
```

上述注释只是提供了版权信息、文件内容和历史版本信息等，文件注释要根据本身的实际情况来包括相应的内容。

5.2.2 文档注释

文档注释就是指这种注释内容能够生成 API 帮助文档，JDK 中 javadoc 命令能够提取这些注释信息并生成 HTML 文件。文档注释主要对类(或接口)、实例变量、静态变量、实例方法和静态方法等进行注释。

提示 文档是要给别人看的帮助文档，一般注释的实例变量、静态变量、实例方法和静态方法都应该是非私有的，那些只给自己看的内容可以不用文档注释。

示例代码如下：

```
package com.zhijieketang;

/**
 * 自定义的日期类，具有日期基本功能，继承 java.util.Date
 * <p>实现日期对象和字符串之间的转换</p>
 * @author 关东升
 */
public class Date extends java.util.Date {

    private static final int DEFAULT_CAPACITY = 10;

    /**
     * 容量
     */
    public int size;

    /**
     * 将字符串转换为 Date 日期对象
     * @param s 要转换的字符串
     * @return Date 日期对象
     */
    public static Date valueOf(String s) {

        final int YEAR_LENGTH = 4;
        final int MONTH_LENGTH = 2;

        int firstDash;
        int secondDash;

      ...
    }

    /**
     * 将日期转换为 yyyy-mm-dd 格式的字符串
     * @return yyyy-mm-dd 格式的字符串
     */
    public String toString () {
        int year = super.getYear() + 1900;
        int month = super.getMonth() + 1;
        int day = super.getDate();
      ...
    }
}
```

由于文档注释最终会生成 HTML 文档，所以可以在文档注释中使用 HTML 标签，上述注释中的 是 HTML 段落标签。

另外，上述的文档注释中还用到了@author、@return 和@param 等文档注释标签，这些标签能够方便生成 API 帮助文档。表 5-1 所示是常用的文档注释标签。

表 5-1 文档注释标签

标　签	描　述	标　签	描　述
@author	说明类或接口的作者	@see	参考另一个主题的链接
@deprecated	说明类、接口或成员已经废弃	@exception	说明方法所抛出的异常类
@param	说明方法参数	@throws	同@exception 标签
@return	说明返回值	@version	类或接口的版本

如果想生成 API 帮助文档，则可以使用 javadoc 指令，如图 5-1 所示，在命令行中输入 javadoc-encoding UTF-8-d apidoc Date.java 指令，其中参数-encoding UTF-8 是指定文件编码为 UTF-8 中文乱码；-d 参数指明要生成文档的目录；apidoc 是当前目录下面的 apidoc 目录，如果不存在 apidoc，则会创建一个 apidoc 目录；Date.java 是当前目录下的 Java 源文件。

图 5-1 生成 API 帮助文档

如果生成成功，则在当前 apidoc 目录下生成很多 HTML 文件，其中的 index.html 文件是文档的入口，双击此文件，打开如图 5-2 所示的页面，从页面中可见在 Date.java 中注释的内容会出现在 HTML 页面中。

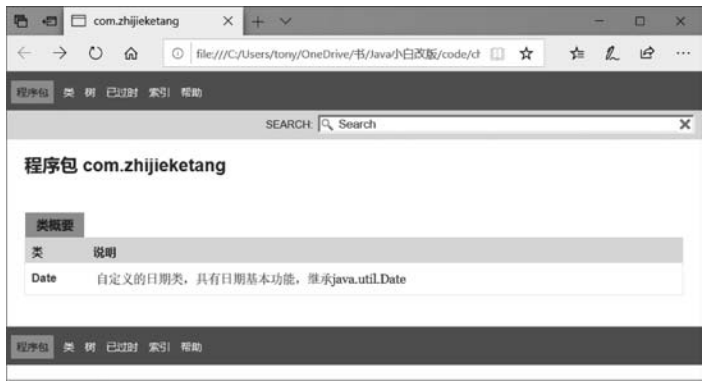

图 5-2 API 帮助文档

5.2.3 代码注释

程序代码中处理文档注释还需要在一些关键的地方添加代码注释，文档注释一般是给一些看不到

源代码的人看的帮助文档，而代码注释则是给阅读源代码的人参考的。代码注释一般采用单行注释(//)和多行注释(/*…*/)。

示例代码如下：

```
public class Date extends java.util.Date {

    // 默认的容量,是一个常量                                          ①
    private static final int DEFAULT_CAPACITY = 10;

    /**
     * 容量
     */
    public int size;

    /**
     * 将字符串转换为 Date 日期对象
     * @param s 要转换的字符串
     * @return Date 日期对象
     */
    public static Date valueOf(String s) {

        final int YEAR_LENGTH = 4;
        final int MONTH_LENGTH = 2;

        int firstDash;
        int secondDash;

        Date d = null;
        …

        /*                                                            ②
         * 判断 d 是否为空,
         * 如果为空则抛出异常 IllegalArgumentException,否则返回 d
         */
        if (d == null) {
            throw new java.lang.IllegalArgumentException();
        }

        return d;
    }

    /**
     * 将日期转换为 yyyy-mm-dd 格式的字符串
     * @return yyyy-mm-dd 格式的字符串
     */
    public String toString () {
        int year = super.getYear() + 1900;     //计算年份              ③
        int month = super.getMonth() + 1;     /*计算月份*/             ④
        int day = super.getDate();
        …
    }
}
```

上述代码第①行采用了单行注释，要求与其后的代码具有一样的缩进层级。如果注释的文字很多，则可以采用多行注释，见代码第②行。多行注释也要求与其后的代码具有一样的缩进层级。有时也会

在代码的尾端进行注释,这要求注释内容极短,应该再有足够的空白来分开代码和注释,见代码第③行和第④行。


微课视频

5.3 代码排版

代码排版包括空行、空格、断行和缩进等内容。代码排版内容比较多,工作量很大,也非常重要。

5.3.1 空行

空行用以将逻辑相关的代码段分隔开,以提高可读性。空行使用规范如下:

(1) 类声明和接口声明之间保留两个空行。见示例 Date.java 代码第⑧行和第⑨行。

(2) 两个方法之间保留一个空行。见示例 Date.java 代码第⑦行。

(3) 方法的第一条语句之前保留一个空行。见示例 Date.java 代码第⑤行。

(4) 代码注释(除尾端注释外)之前保留一个空行。见示例 Date.java 代码第①、②、③、④行。

(5) 一个方法内的两个逻辑段之间保留一个空行。见示例 Date.java 代码第⑥行。

示例 Date.java 代码如下:

```
/*
 * 版权所有 2015 北京智捷东方科技有限公司
 * 许可信息查看 LICENSE.txt 文件
 * 描述:
 *   实现日期基本功能
 * 历史版本:
 *   2015-7-22: 创建 关东升
 *   2015-8-20: 添加 socket 库
 *   2015-8-22: 添加 math 库
 */
package com.zhijieketang;
            ①
/**
* 自定义的日期类,具有日期基本功能,继承 java.util.Date
* <p>实现日期对象和字符串之间的转换</p>
* @author 关东升
*/
public class Date extends java.util.Date {
            ②
    // 默认的容量,是一个常量
    private static final int DEFAULT_CAPACITY = 10;
            ③
    /**
     * 容量
     */
    public int size;
            ④
    /**
     * 将字符串转换为 Date 日期对象
     * @param s 要转换的字符串
     * @return Date 日期对象
     */
    public static Date valueOf(String s) {
            ⑤
```

```
        final int YEAR_LENGTH = 4;
        final int MONTH_LENGTH = 2;

        int firstDash;
        int secondDash;
            ⑥
        Date d = null;
        …

        /*
         * 判断 d 是否为空,
         * 如果为空则抛出异常 IllegalArgumentException,否则返回 d.
         */
        if (d == null) {
            throw new java.lang.IllegalArgumentException();
        }

        return d;
    }
            ⑦
    /**
     * 将日期转换为 yyyy-mm-dd 格式的字符串
     * @return yyyy-mm-dd 格式的字符串
     */
    public String toString () {

        int year = super.getYear() + 1900; //计算年份
        int month = super.getMonth() + 1; /* 计算月份 */
        int day = super.getDate();
        …
    }
}

            ⑧
class A {

}

            ⑨
class B {

}
```

5.3.2 空格

代码中的有些位置是需要有空格的,这个工作量也很大。下面是使用空格的规范。

(1) 赋值符号“=”前后各有一个空格。示例代码如下:

```
int YEAR_LENGTH = 4;
int day = super.getDate();
```

(2) 所有的二元运算符都应该使用空格与操作数分开。示例代码如下:

```
a += c + d;
prints("size is " + foo + "\n");
```

(3) 一元操作符,即负号“-”、自增“++”和自减“--”等,它们与操作数之间没有空格。示例代码如下:

```
int a = -b;
a++;
--b;
```

(4) 小左括号“(”之后,小右括号“)”之前不应有空格。示例代码如下:

```
a = (a + b) / (c * d)
```

(5) 大左括号“{”之前有一个空格。示例代码如下:

```
while (a == d) {
    n += 1
}
```

(6) 方法参数列表小左括号“(”之前没有空格,小右括号“)”之后有一个空格,参数列表中参数逗号“,”之后也有一个空格。示例代码如下:

```
String format(Object obj, StringBuffer toAppendTo, FieldPosition fieldPosition) {
        …
}
```

(7) 关键字之后紧跟着小左括号“(”,关键字之后应该有一个空格。如下示例中 while 之后有一个空格。

```
while (a == d) {
    …
}
```

5.3.3 缩进

4 个空格常被作为缩进排版的一个单位。虽然在开发时程序员使用制表符进行缩进,而默认情况下一个制表符等于 8 个空格,但是不同的 IDE 工具中一个制表符与空格对应个数会有不同。IntelliJ IDEA 中默认是一个制表符对应 4 个空格。

缩进可以依据如下一般规范。

(1) 在方法、Lambda、控制语句等包含大括号“{}”的代码块中,代码块的内容相对于首行缩进一个级别(4 个空格)。

(2) 如果是 if 语句中条件表达式的断行,那么新的一行应该相对于上一行缩进两个级别(8 个空格),再往后的断行要与第一次的断行对齐。

示例代码如下:

```
public class Date extends java.util.Date {

    …

    public String getString() {

        int year = super.getYear() + 1900; // 计算年份
        int month = super.getMonth() + 1; /* 计算月份 */
```

```
        int day = super.getDate();

        if ((longName1 == longName2)
                || (longName3 == longName4) && (longName3 > longName4)          ①
                && (longName2 > longName5)) {                                   ②

        }

        return null;
    }
}
```

上述代码第①行和第②行是 if 语句条件表达式的断行，代码第①行和第②行要对齐。

5.3.4 断行

一行代码的长度应尽量不要超过 80 个字符，如果超过则需断行，可以依据下面的一般规范断开。

(1) 在一个逗号后面断开。

(2) 在一个操作符前面断开，要选择较高级别的运算符(而非较低级别的运算符)断开。

(3) 新的一行应该相对于上一行缩进两个级别(8 个空格)。

下面通过一些示例加以说明。

```
longName1 = longName2 * (longName3 + longName4 - longName5)
        + 4 * longName6                                                         ①
longName1 = longName2 * (longName3 + longName4
        - longName5) + 4 * longName6                                            ②

private static DateFormat get(int timeStyle, int dateStyle,
                              int flags, Locale loc) {                          ③
    ...
}

if ((longName1 == longName2)
    || (longName3 == longName4) && (longName3 > longName4)
    && (longName2 > longName5)) {                                               ④

}

boolName1 = (longName3 == longName4)
      ? (longName3 > longName4)
      : (longName2 > longName5);                                                ⑤
```

上述代码第①行和第②行是带有小括号运算的表示式，其中代码第①行的断开位置比第②行的断开位置要好。因为代码第①行断开处位于括号表达式的外边，这是一个较高级别运算符的断开。代码第③行是方法名断开，是在参数逗号之后。代码第④行是 if 判断语句，由于可能有很多长的条件表达式，断开的位置应在逻辑运算符处。代码第⑤行是三元运算符的断开。

5.4 其他规范


微课视频

除了上述规范外，还有很多零散的规范。下面补充一些重要的规范。

(1) 在声明变量或常量时推荐一行一个声明。示例代码如下：

```
// 推荐使用:
int longName1 = 0 ;
int longName2 = 0 ;
// 不推荐使用:
int longName1 = 0, longName2 = 0 ;
```

(2) 左大括号“{”位于声明语句同行的末尾。右大括号“}”另起一行,与相应的声明语句对齐,除非是一个空语句,右大括号“}”应紧跟在左大括号“{”之后。示例如下:

```
public class Date extends java.util.Date {

    int longName1 = 0;
    int longName2 = 0;

    boolean boolName1 = true;

    public String getString() {

        int year = super.getYear() + 1900;     // 计算年份
        int month = super.getMonth() + 1;      /* 计算月份 */
        int day = super.getDate();

        return null;
    }

    public void setString() {}
}
```

(3) 每行至多包含一条语句。示例如下:

```
// 推荐使用:
argv++;
argc--;
// 不推荐使用:
argv++; argc--;
```

(4) 虽然 Java 语言允许 if、for 等控制语句在只有一行代码情况下省略左右两个大括号,但是编码规范并不推荐这样使用。示例如下:

```
// 推荐使用:
if (1 == 3) {
    x = 2.3;
}
// 不推荐使用:
if (1 == 3)
    x = 2.3;
```

关于规范,事实上还有很多,不能穷尽,这里不再赘述。

5.5 本章小结

通过对本章内容的学习,读者可以了解 Java 的编码规范,包括命名规范、注释规范、声明规范和代码排版等内容。

5.6 同步练习

选择题

1. 下列选项中哪些Java类命名符合Java命名规范？（ ）

A. dummy_threading　　B. SplitViewController

C. WEEK_OF_MONTH　　D. balance_account

2. 下列选项中哪些方法名符合Java命名规范？（ ）

A. dummy_threading　　B. SplitViewController

C. WEEK_OF_MONTH　　D. balanceAccount

3. 下面哪种注释是Java支持的？（ ）

A. /*...*/　　B. /**...*/

C. #　　D. 三重双引号“"""”包裹起来