

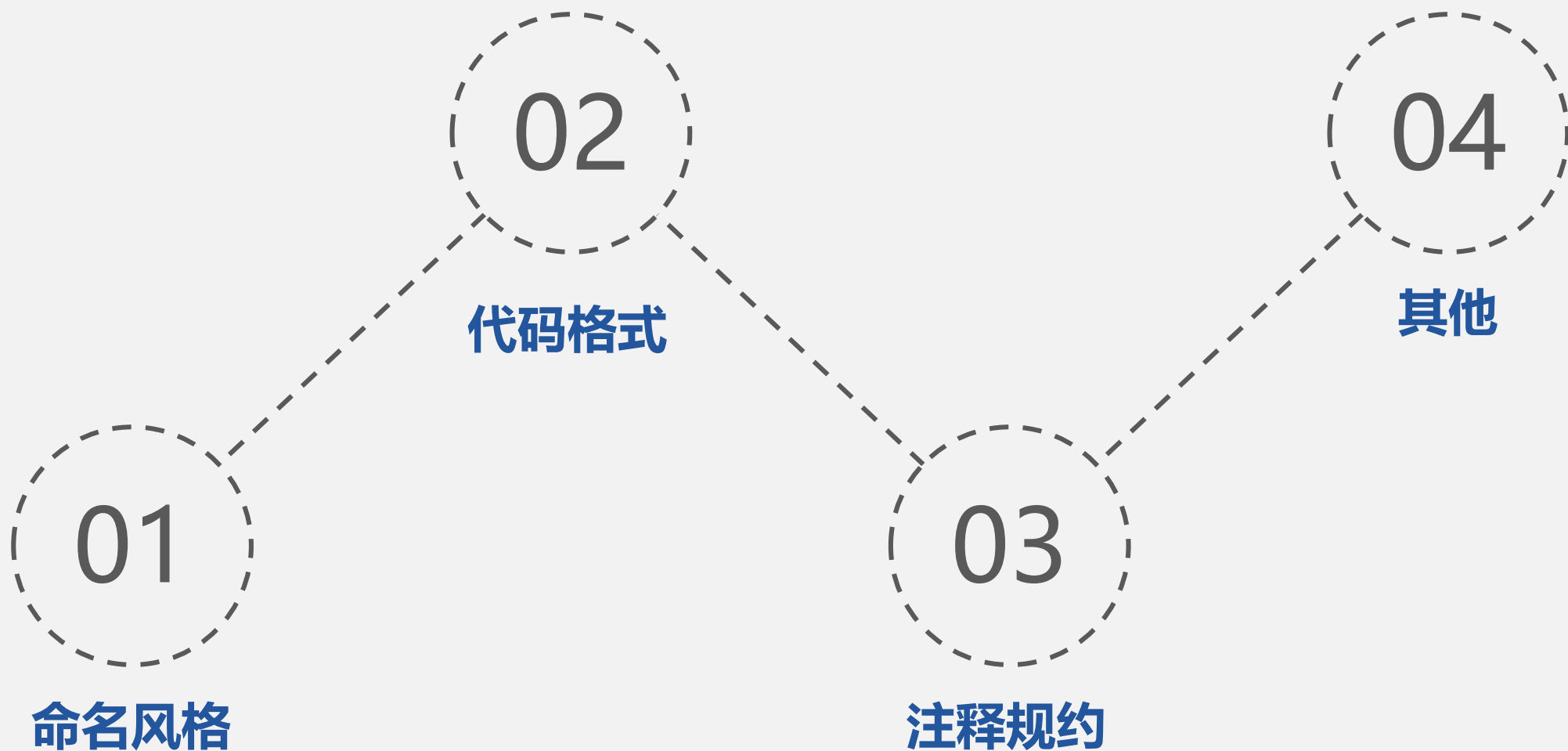


# 编程规范

2019/5/5 于晓静

# 目录

---



# 命名风格

- 驼峰命名法 (camelCase) / 下划线命名法 (under\_score\_case)
- 常量命名全部大写 (MAX\_COUNT)
- 根据情况给名字加前后缀 (g\_value)
- 使用尽可能完整的单词组合来表达自定义变量意思 (AtomicReferenceFieldUpdater) (*int a*)
- 使用可搜索的命名，命名长短应与其作用域相对应。

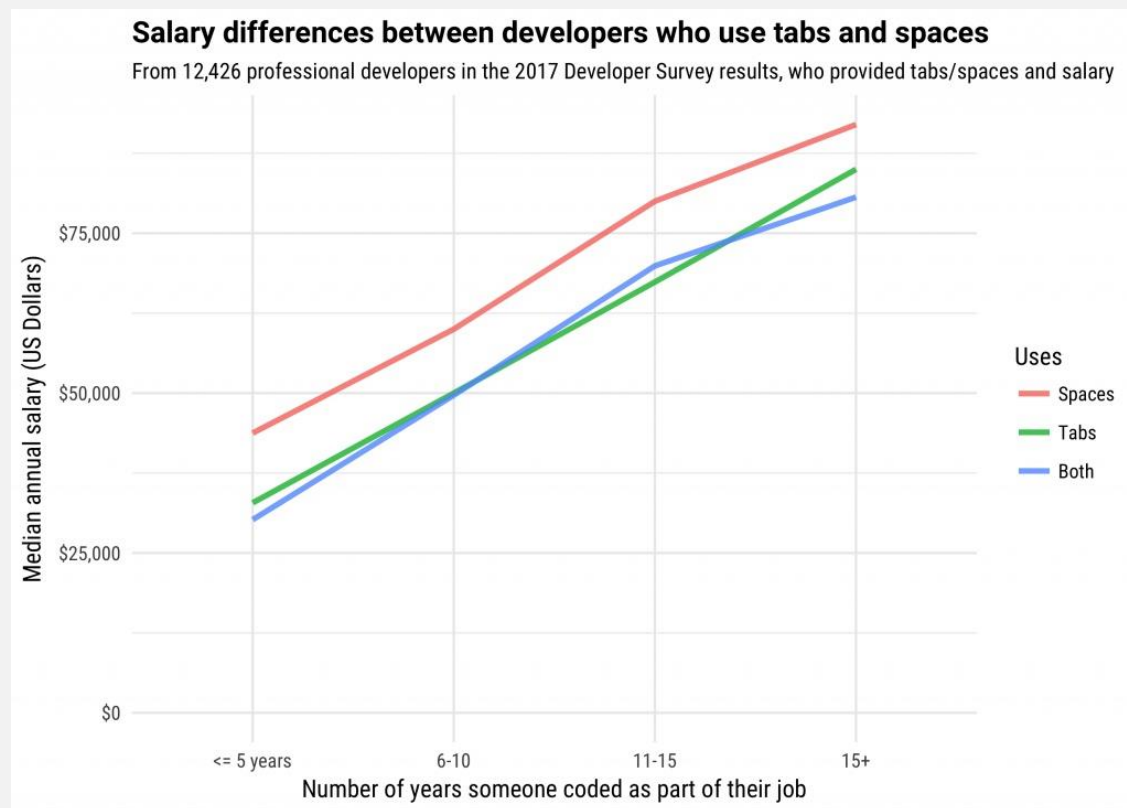
**需要注释来补充的命名就不算是好命名!**

# 代码格式

- 4 字符缩进，不使用tab字符。 (如果使用tab缩进，设置1个tab为4个空格)

# 代码格式

- 4 字符缩进，不使用tab字符。（如果使用tab缩进，设置1个tab为4个空格）



# 代码格式

- 4 字符缩进，不使用tab字符。 (如果使用tab缩进，设置1个tab为4个空格)
- 一行最长不超过80个字符，超过的使用换行展示，尽量保持格式优雅。
- 每行只写一条语句，每行只写一个声明。
- 大括号规则以及换行规则与团队保持一致即可，**不要混用！**

# 注释规约

- 好的命名、代码结构是自解释的，注释力求精简准确、表达到位。
- 将注释放在单独的行上，而非代码行的末尾。
- 谨慎注释代码。在上方详细说明，而不是简单地注释掉。如果无用，则删除。
- 代码修改的同时，注释也要进行相应的修改。
- 以句点结束注释文本。

# 其他代码规范

- 不在代码中使用魔法值（即未经预先定义的常量）。  
*(key = 1000 + tradeNumber)*
- 不要在条件判断中执行复杂的语句，避免采用取反逻辑运算符。  
*(if (x < 628))      (if (!(x >= 628)))*
- 数据结构的构造或初始化指定大小，避免数据结构无限增长吃光内存。
- 不同逻辑和功能的代码之间可插入一个空行分隔开来以提升可读性，但没有必要插入多个空行或重复符号注释进行隔开。



## 举例


```
int max_length =
.....total_length; //初始化以该节点为首个单词的最长单词链长度为1
```

```
while (!path.empty()){
    Vertex &v = l_Vertex[path.back()];
    //如果是搜索到这个节点的话
    if (indexpath.size() < path.size()){
```

```
w = false; c = false; h = false; t = false; n = false;
return longest chain size;
```

```
for (int i = 0; i < words.size(); i++) {
    if (s.count(words[i]) > 0) continue;
    else { s.insert(words[i]); totalWord++; }
}
```

[illegible]



## 参考文档和网站



《码出高效：Java开发手册》

[Microsoft C# Coding Conventions](#)

<https://github.com/golang/go/wiki/CodeReviewComments>

<http://google.github.io/styleguide/>



The background features a light gray field with several thin, dark gray lines that intersect to form a series of overlapping triangles and polygons. The word "THANKS" is centered in a bold, blue, sans-serif font.

**THANKS**