# Java 编码规范与实践

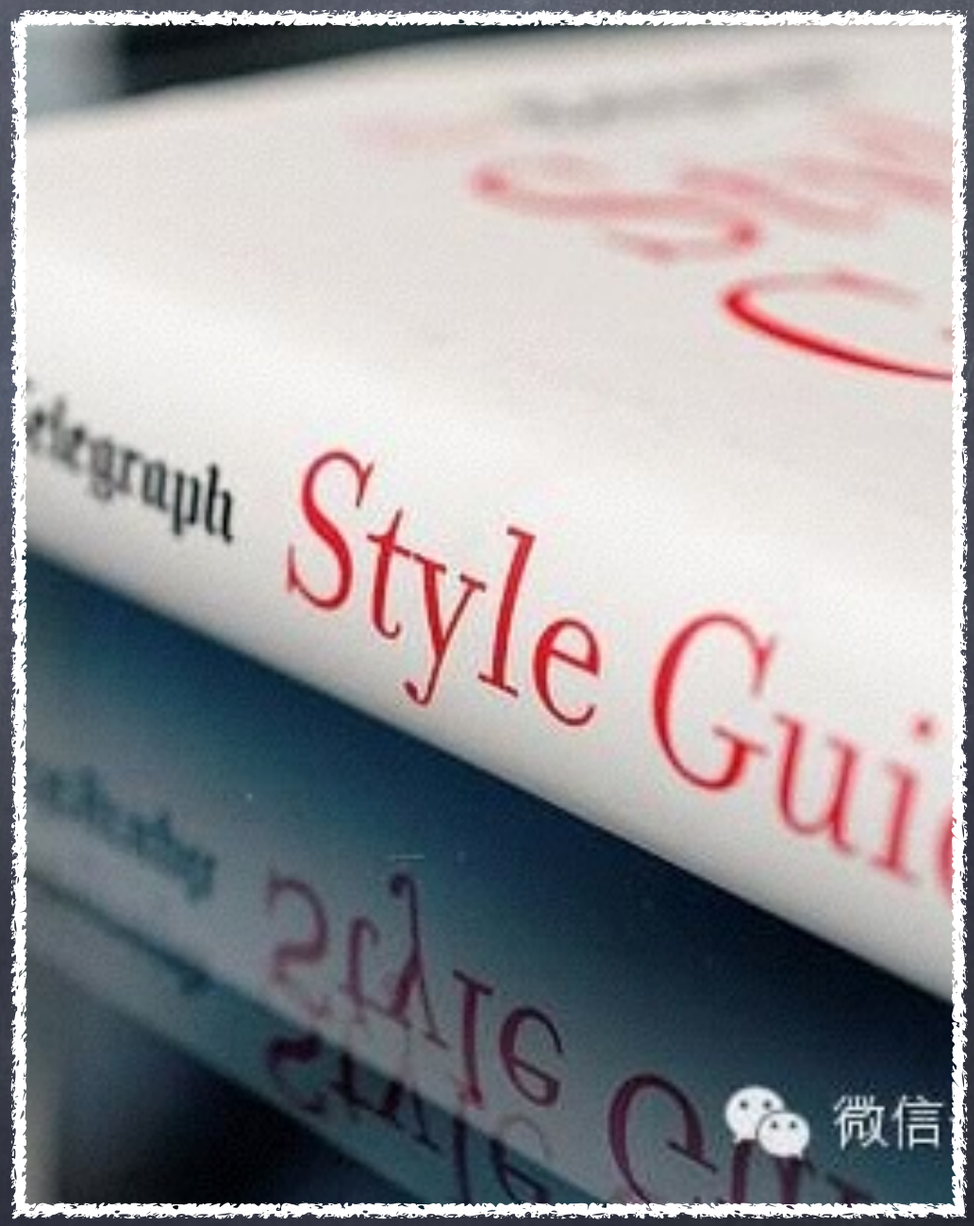
胡 峰 @云下山巅

产品创新 - 通讯交互平台部

2014-04-16

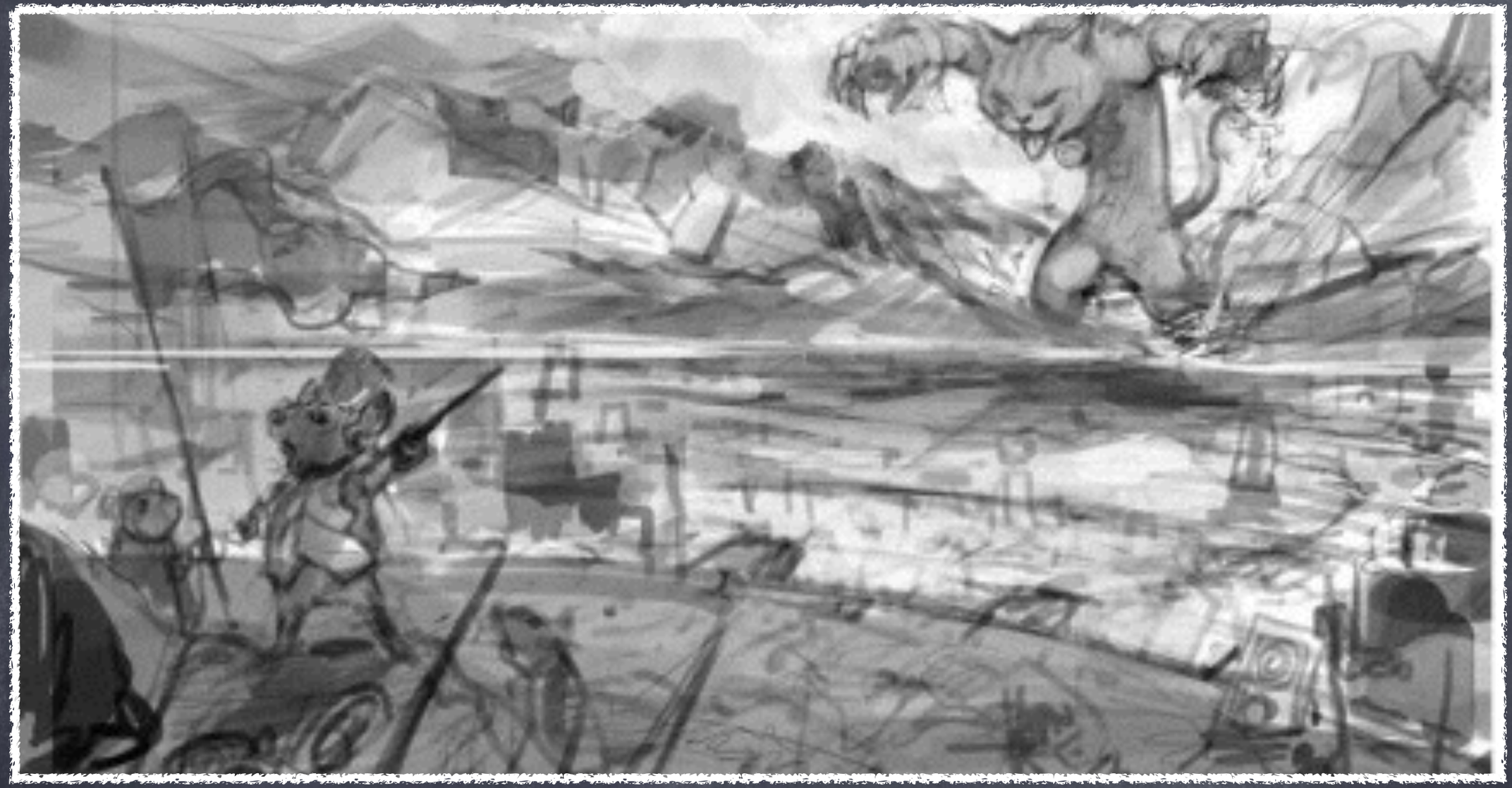
# 意义

- 改善可读性
- 降低维护成本
- 提升团队开发合作效率



# 境界

- 能读
- 易读
- 赏心悦目



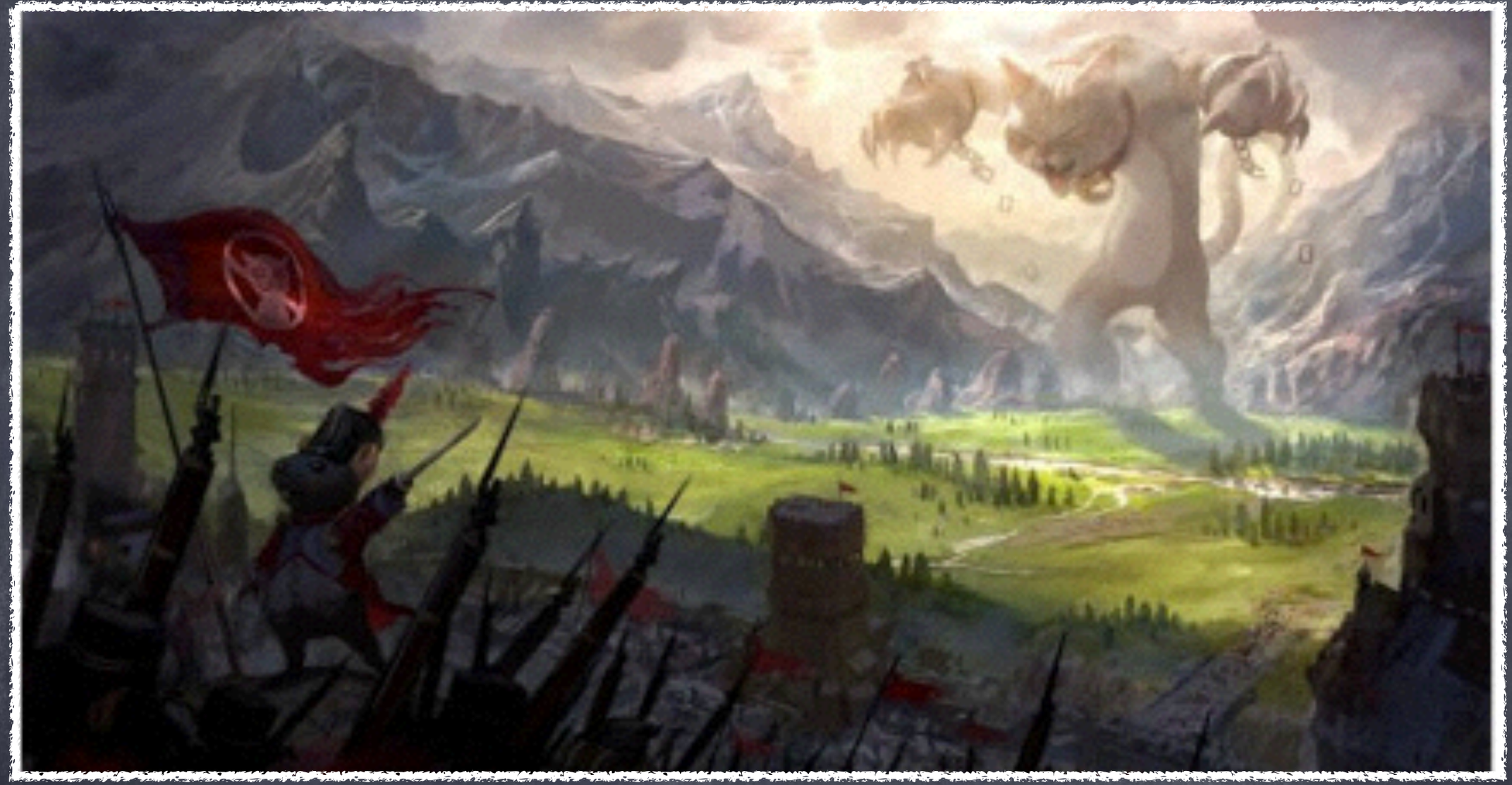
# 境界一：能读

Java Code Style (oracle, google)约定像命名、编码、缩进、断行等等

IDE 协助检查，自动格式化

# 境界二：易读

一致性（术语、风格）| 名词、动词、形容词选择 | 符合英文语法，可发音 | 留白、聚焦、去重



# 境界三：赏心悦目

简洁 清晰 平衡 优雅

```java
@Autowired
private Config config;
@Autowired
private CacheServiceForServer cacheServiceForServer;
@Autowired
private CacheServiceForStatus cacheServiceForStatus;
private ScheduledExecutorService ses = Executors
        .newSingleThreadScheduledExecutor(new NamedThreadFactory(
                "chat-core-monitor-report"));
private Map<String, String> hash = new TreeMap<String, String>();
private String key;
private String ip;
```

# 能读 - IDE自动格式化

```
private final    Queue<NioByteChannel>           newChannels       = new ConcurrentLinkedQueue<NioByteChannel>()    ;
private final    Queue<NioByteChannel>           flushingChannels  = new ConcurrentLinkedQueue<NioByteChannel>()    ;
private final    Queue<NioByteChannel>           closingChannels   = new ConcurrentLinkedQueue<NioByteChannel>()    ;
private final    Map<String, NioByteChannel>     udpChannels       = new ConcurrentHashMap<String, NioByteChannel>();
private final    AtomicReference<ProcessThread> processThreadRef  = new AtomicReference<ProcessThread>()           ;
private final    NioByteBufferAllocator          allocator         = new NioByteBufferAllocator()                   ;
private final    AtomicBoolean                   wakeupCalled      = new AtomicBoolean(false)                       ;
private final    NioChannelIdleTimer             idleTimer                                                          ;
private final    NioConfig                       config                                                             ;
private final    Executor                        executor                                                           ;
private          IoProtocol                      protocol                                                           ;
private volatile Selector                        selector                                                           ;
private volatile boolean                         shutdown          = false                                          ;
```

# 易读 — 对齐聚焦

```
private volatile boolean                    shutdown        = false                                                    ;


// ~ ------------------------------------------------------------------------------------------------------------------


NioProcessor(NioConfig config, IoHandler handler, NioChannelEventDispatcher dispatcher, NioChannelIdleTimer idleTimer) {
    this.config     = config;
    this.handler    = handler;
    this.dispatcher = dispatcher;
    this.idleTimer  = idleTimer;
    this.executor   = Executors.newCachedThreadPool(new NamedThreadFactory("craft-atom-nio-processor"));

    try {
        selector = Selector.open();
    } catch (IOException e) {
        throw new RuntimeException("Fail to startup a processor", e);
    }
}


// ~ ------------------------------------------------------------------------------------------------------------------


/**
 * Adds a nio channel to processor's new channel queue, so that processor can process I/O operations associated this channe
 *
 * @param channel
 */
public void add(NioByteChannel channel) {
```

# 易读 - 分割聚焦

```
public abstract class NioBuilder {

    protected final IoHandler                   handler                                                    ;
    protected       NioChannelEventDispatcher   dispatcher        = new NioOrderedThreadPoolChannelEventDispatcher();
    protected       NioBufferSizePredictorFactory predictorFactory = new NioAdaptiveBufferSizePredictorFactory()  ;
    protected       int                         readBufferSize    = 2048                                       ;
    protected       int                         minReadBufferSize = 64                                         ;
    protected       int                         maxReadBufferSize = 65536                                      ;
    protected       int                         ioTimeoutInMillis = 120 * 1000                                 ;
    protected       int                         processorPoolSize = Runtime.getRuntime().availableProcessors()  ;
    protected       int                         executorSize      = processorPoolSize << 3                     ;
    protected       int                         channelEventSize  = Integer.MAX_VALUE                          ;
    protected       int                         totalEventSize    = Integer.MAX_VALUE                          ;
    protected       boolean                     readWriteFair     = true                                       ;


    public NioBuilder(IoHandler handler) {
        this.handler = handler;
    }


    public NioBuilder<T> minReadBufferSize(int size)                                 { this.minReadBufferSize = size      ; return this; }
    public NioBuilder<T> maxReadBufferSize(int size)                                 { this.maxReadBufferSize = size      ; return this; }
    public NioBuilder<T> readBufferSize   (int size)                                 { this.readBufferSize    = size      ; return this; }
    public NioBuilder<T> processorPoolSize(int size)                                 { this.processorPoolSize = size      ; return this; }
    public NioBuilder<T> executorSize     (int size)                                 { this.executorSize      = size      ; return this; }
    public NioBuilder<T> channelEventSize (int size)                                 { this.channelEventSize  = size      ; return this; }
    public NioBuilder<T> totalEventSize   (int size)                                 { this.totalEventSize    = size      ; return this; }
    public NioBuilder<T> ioTimeoutInMillis(int timeout)                              { this.ioTimeoutInMillis = timeout   ; return this; }
    public NioBuilder<T> readWriteFair    (boolean fair)                             { this.readWriteFair     = fair      ; return this; }
    public NioBuilder<T> dispatcher       (NioChannelEventDispatcher dispatcher)     { this.dispatcher        = dispatcher; return this; }
    public NioBuilder<T> predictorFactory (NioBufferSizePredictorFactory factory)   { this.predictorFactory  = factory   ; return this; }


    protected void set(NioConfig config) {
        config.setReadWritefair(readWriteFair)         ;
        config.setTotalEventSize(totalEventSize)       ;
        config.setChannelEventSize(channelEventSize)   ;
        config.setExecutorSize(executorSize)           ;
        config.setProcessorPoolSize(processorPoolSize) ;
        config.setIoTimeoutInMillis(ioTimeoutInMillis) ;
        config.setDefaultReadBufferSize(readBufferSize);
        config.setMinReadBufferSize(minReadBufferSize) ;
        config.setMaxReadBufferSize(maxReadBufferSize) ;
    }

    abstract public T build();

}
```

# 悦目容易，赏心难


# 代码静态检查

```java
@Override
public ChatResult beginFromGuang(String customer, long pid) {
    ChatResult cr = new ChatResult();
    cr.setCanChat(false);
    if (StringUtils.isBlank(customer) || pid == 0) {
        return cr;
    }
    ChatContext ctx = null;
    try {
        ctx = newChatContext(customer, pid);
    } catch (ChatException e) {
        LOG.error("New chat context error!customer=" + customer + ", pid=" + pid, e);
        return cr;
    }
    ctx.setEntry(Entry.guang_item.toString());
    ChatGroup group = chatGuideService.guide(ctx);
    if (group == null) {
        return cr;
    }
    switch (group.getWaiterSeq()) {
    case POP:
        cr = beginFromGuang4Pop(ctx, (PopChatGroup) group);
    case SUPPLIER:
        cr = beginFromGuang4brand(ctx, (SupplierChatGroup) group);
    default:
        break;
    }

    return cr;
}
```

# bug 1

```
public WaiterMessage reconnect4Seller(String waiter, String loginId,
        String connectId) {
    WaiterMessage wmsg = null;
    // 检查是否进行了 ddAuthService 登陆认证
    if (!ddAuthService.isAuth(waiter, loginId)) {
        if(waiter != "client_net_check_pin") {
            LOG.error("RECONNECT: ChatCoreServiceImpl.reconnect is error: not invoke ddAuthService.isAuth!waiter=" +
                    ",loginId=" + loginId + ",connectId=" + connectId);
        }
        throw new ChatException(ChatCode.RECONNECT_NEED_RELOGIN);
    }

    boolean succ = sellerWaiterService.reconnect4Seller0(waiter, connectId);
    if (succ) {
        wmsg = WaiterMessage.newInstance(WaiterMessage.SERVER_FROM, waiter, MessageType.reconnect, loginId, Boolean.t
    } else {
        throw new ChatException(ChatCode.RECONNECT_FAILED);
    }

    return wmsg;
}
```

# bug 2

```
        if (StringUtils.isNotBlank(sid)) {
            // TODO 此处PamsUid为空
            String pamsUid = cacheServiceForServer.getChatSessionPamsUid(sid);
            if (StringUtils.isNotBlank(pamsUid)) {
                ctx = cacheServiceForServer.getChatContext(pamsUid);
                chatContext = ctx.clone();
            }
        }

        if (ChatEntrance.SUP_TO_SUP_TRANS.name().equals(tmsg.getChatEn())) {
            chatContext.setTransGroup(group); // 设置转接技能组
            if (chatContext != null) {
                chatContext.switchTransPrams();
            }
        }
```

bug 3

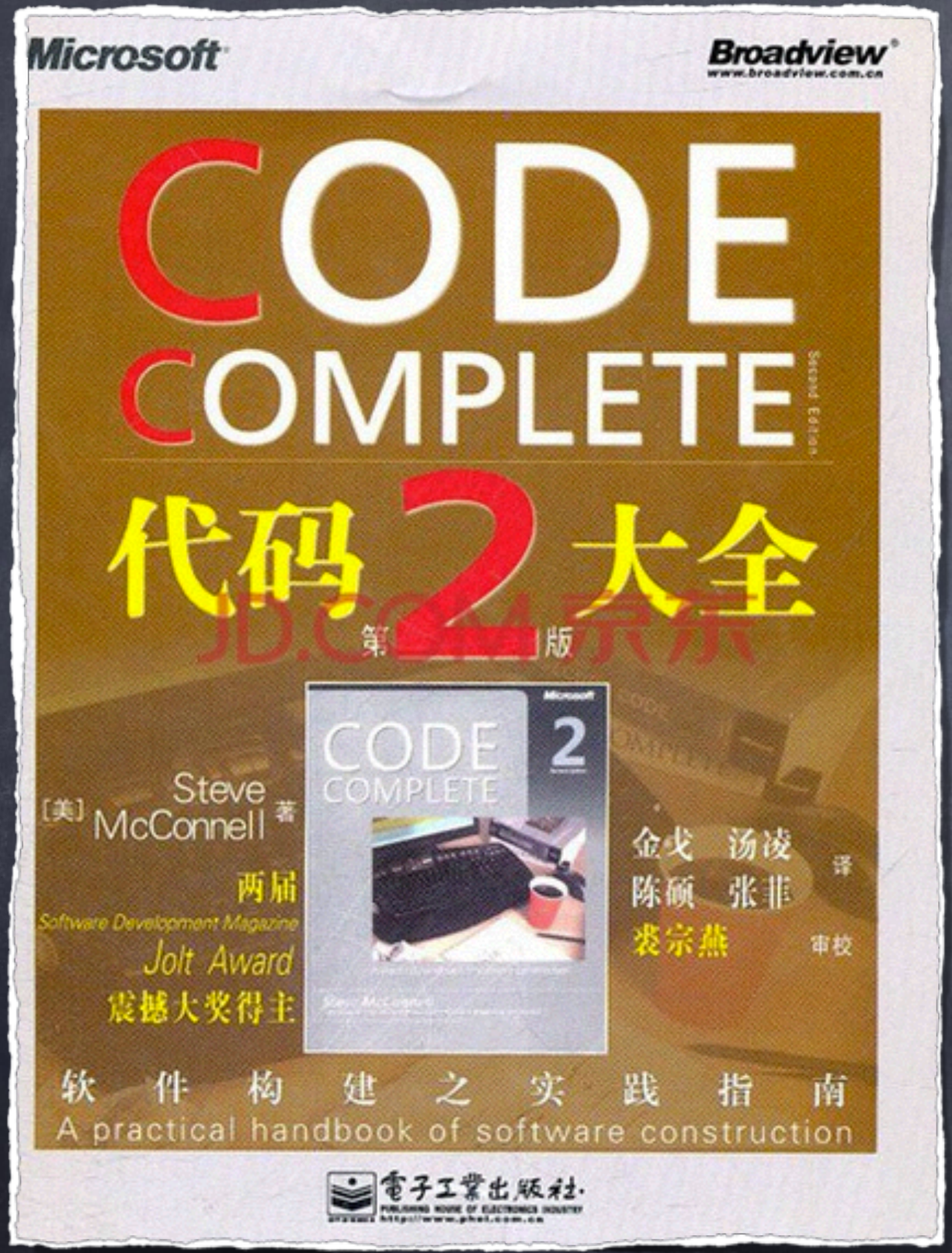
```java
public boolean isServesHit(String servesPin) {
    boolean isHit = true;

    // 全部跳转
    if (this.isAllRedirect != null && true == this.isAllRedirect) {
        return true;
    }

    // 按客服账号
    if (this.servesSet != null ) {
        isHit &= this.servesSet.contains(servesPin);
    }

    // 按百分比
    if (percentage != null) {
        isHit &= (random.nextInt(MOD) < percentage);
    }

    // 按客服类型
    if (serverTypeSet != null) {
        Map<String, String> typeMap = WaiterTypeFactory.getWaiterTypeEntry();
        if(typeMap != null){    //可能为空
            String type = typeMap.get(servesPin);
            if (null != type ) {
                isHit &= serverTypeSet.contains(type);
            }
        }
    } else {
        LOG.equals("The accountTypeMap is null! Please check the status of worker!");
    }
    return isHit;
}
```

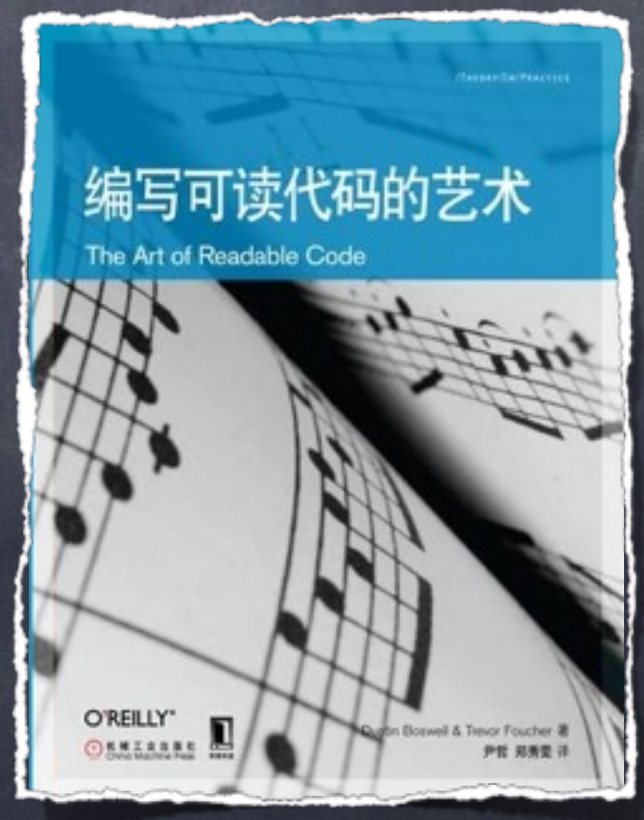
bug 4

# 总结：普适原则

- 将一段程序当作一个表格
- 命名，选择简单的英语（正确适用名、动、形容词），可发音
- 使用空白、分割表现结构
- 短方法，浅嵌套
- 个人风格与一致性：必要时牺牲个人保持一致，尊重原作者
- 像写注释一样写代码（可读性），像写代码一样写注释（精确性）
- 聚焦代码，而非注释
- 清晰胜于机巧
- 审美很重要

# 推荐阅读


Microsoft
Broadview
www.broadview.com.cn
CODE COMPLETE
Second Edition
代码大全
第二版
CODE COMPLETE 2
[美] Steve McConnell 著
两届
Software Development Magazine
Jolt Award
震撼大奖得主
金戈 汤凌
陈硕 张菲 译
裘宗燕 审校
软件构建之实践指南
A practical handbook of software construction
電子工業出版社


编写可读代码的艺术
The Art of Readable Code
O'REILLY

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

-Martin Fowler