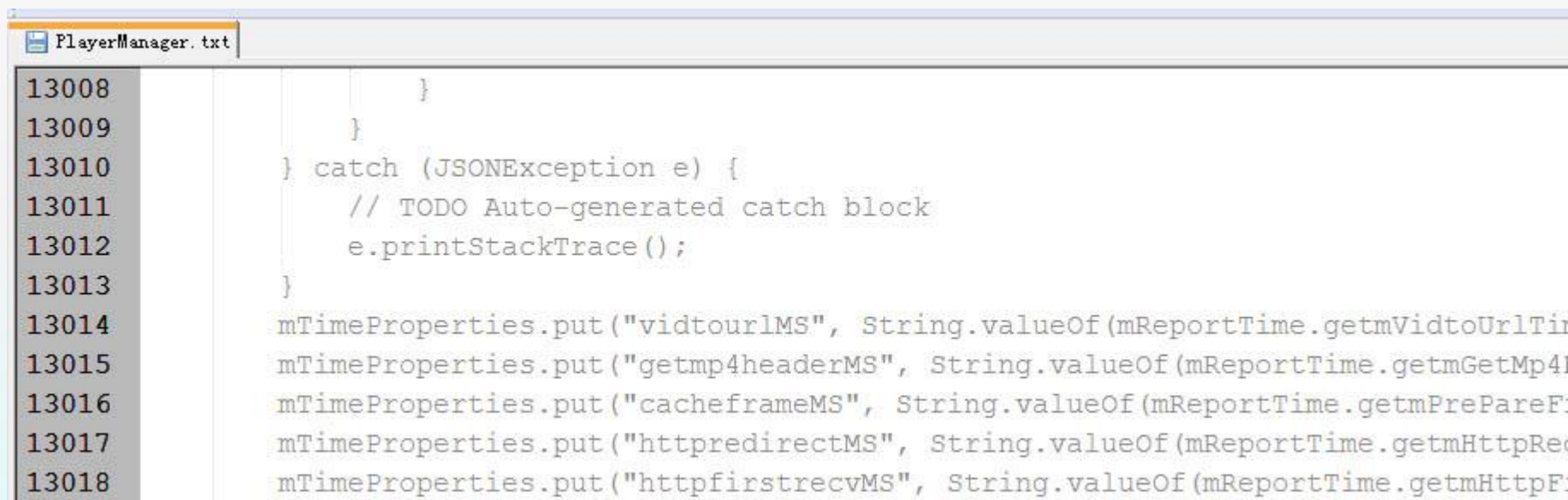


代码整洁之道（Android版）

腾讯视频Android组
oscarjiang

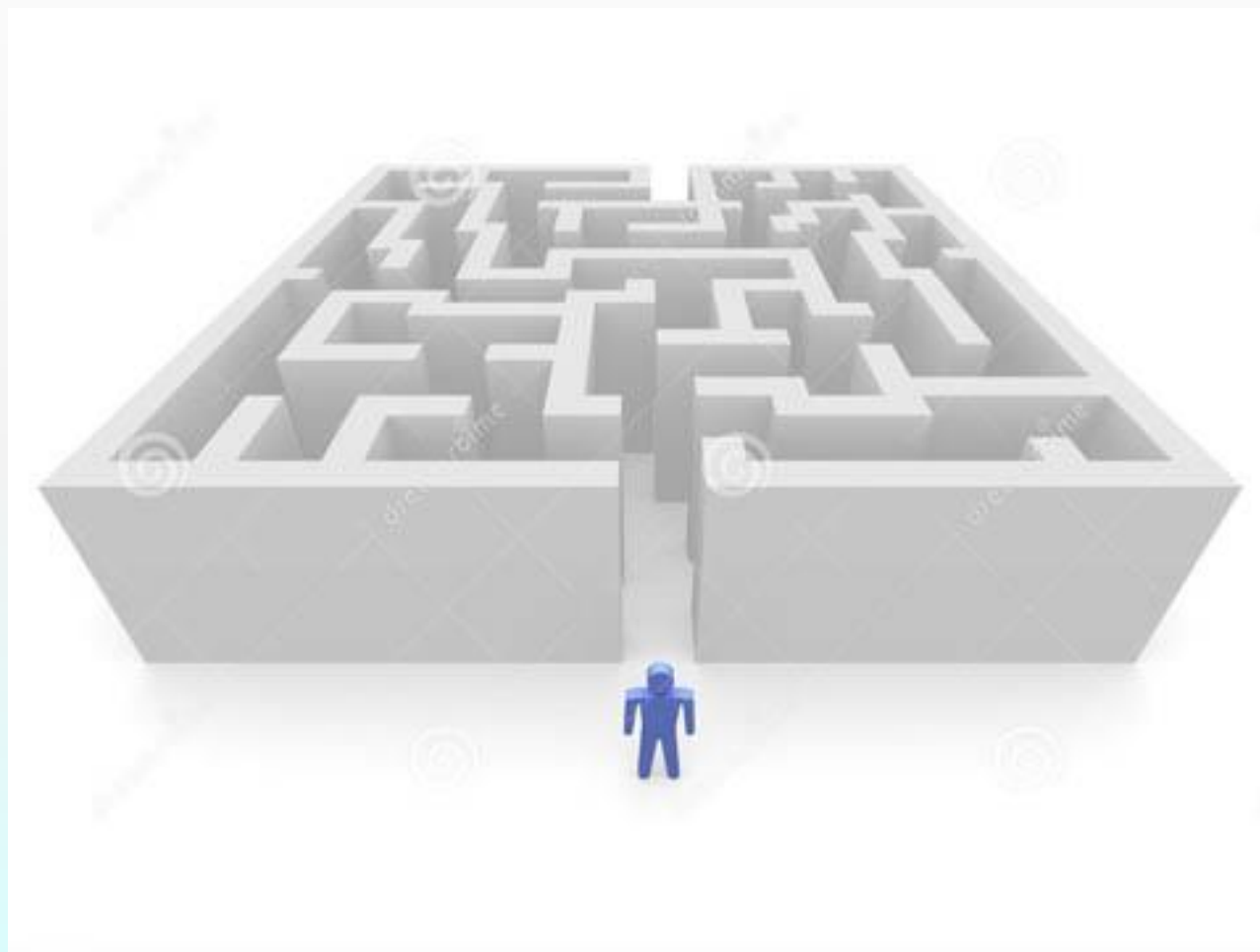


当老大让你去改这样一份代码时...



```
13008         }
13009     }
13010     } catch (JSONException e) {
13011         // TODO Auto-generated catch block
13012         e.printStackTrace();
13013     }
13014     mTimeProperties.put("vidtourlMS", String.valueOf(mReportTime.getmVidtoUrlTim
13015     mTimeProperties.put("getmp4headerMS", String.valueOf(mReportTime.getmGetMp4I
13016     mTimeProperties.put("cacheframeMS", String.valueOf(mReportTime.getmPrePareF
13017     mTimeProperties.put("httpredirectMS", String.valueOf(mReportTime.getmHttpRe
13018     mTimeProperties.put("httpfirstrecvMS", String.valueOf(mReportTime.getmHttpF
```

独闯龙潭



```
/**
 * 此处代码极为复杂，如无必要，请绕行
 */
public class PlayerManager
{
    /**
     private static final String TAG = "MediaPlayerMgr";
    */
     private static final String FILE_NAME = "MediaPlayerManager.java";

     private final int INTERNAL_Main_MSG_Load_PostrollAd  = 900001;
    // private final int INTERNAL_Main_MSG_Load_MidAd      = 900002;
```

现象

- ◆ 代码会腐烂
- ◆ 代码泥潭
- ◆ 压力山大

刻舟求剑式编程



祈祷式编程



```
01. //  ┌─┐      ┌─┐
02. //  ┌─┐ ┌───┐ ┌─┐
03. //  │              │
04. //  │          ─    │
05. //  │  └─┐      └─┐
06. //  │              │
07. //  │          ┌    │
08. //  │              │
09. //  └─┐      ┌─┐
10. //  │          │    神兽保佑
11. //  │          │    代码无BUG!
12. //  │          └───┐
13. //  │              └─┐
14. //  │              ┌─┐
15. //  └─┐ ┌─┐ ┌─┐ └─┐
16. //  │ ┌─┐ │ ┌─┐ │
17. //  │ └─┐ │ └─┐ │
```

代码考古工程师



代码考古工程师

```
/**  
 * 这个不知道谁写的  
 * 看起来没用  
 * 但我不敢删  
 */  
public static String makeGetPlayerConfigUrl() {  
    TencentVideo.setStaGuid(staGuid, false);  
    HashMap<String, String> params = new HashMap<String, String>();
```

程序员A

- ◆ 需求第一版：只有两种情况，0和1

```
public String getErrorMessage(int errorCode) {  
    if (errorCode == 0) {  
        return "";  
    } else {  
        return "下载错误";  
    }  
}
```

程序员B

◆ 需求第二版：增加4种情况

```
public String getErrorMessage(int errorCode) {  
    if (errorCode == 1) {  
        return "存储不足";  
    } else if (errorCode == 2) {  
        return "网络异常";  
    } else if (errorCode == 3) {  
        return "视频下架";  
    } else if (errorCode == 4) {  
        return "IP限制";  
    } else if (errorCode == 5) {  
        return "版权受限";  
    } else {  
        return "";  
    }  
}
```

程序员C

◆ 需求第三版：再增加4种情况

```
public String getErrorMessage(int errorCode) {  
    if (errorCode == 1){  
        return "存储不足";  
    } else if (errorCode == 2) {  
        return "网络异常";  
    } else if (errorCode == 3) {  
        return "视频下架";  
    } else if (errorCode == 4) {  
        return "IP限制";  
    } else if (errorCode == 5) {  
        return "版权受限";  
    } else if (errorCode == 6) {  
        return "存储异常";  
    } else if (errorCode == 7) {  
        return "服务器异常";  
    } else if (errorCode == 8) {  
        return "VIP专享";  
    } else if (errorCode == 9) {  
        return "账号异常";  
    } else {  
        return "";  
    }  
}
```


代码为什么会腐烂

- ◆ 程序员的惯性思维





破窗效应

- ◆ 一个很干净的地方，人会不好意思丢垃圾
- ◆ 但是一旦地上有垃圾出现之后，人就会毫不犹豫的丢垃圾上去，**丝毫不觉羞愧。**
- ◆ 糟糕的代码会引来滚雪球效应。
- ◆ **勿以恶小而为之，勿以善小而不为**

```
float getPayAmount(int storeId,int count) {  
    float price = getPrice(storeId);  
    price = price * count;  
    price = price * getDiscount(storeId, count);  
    return price;  
}
```

变量不混用原则


```
String appendParam(String url, String param) {  
    if (url.endsWith("?")) {  
        url += param;  
    } else {  
        url += "?" + param;  
    }  
    return url;  
}
```

参数不修改原则

```
boolean checkSecurity(String[] people) {  
    boolean found = false;  
    for(int i = 0; i < people.length; i++) {  
        if (!found) {  
            if(people[i].equals("Don")) {  
                showAlert();  
                found = true;  
            }  
            if(people[i].equals("John")) {  
                showAlert();  
                found = true;  
            }  
        }  
    }  
    return found;  
}
```

烂代码对公司的伤害

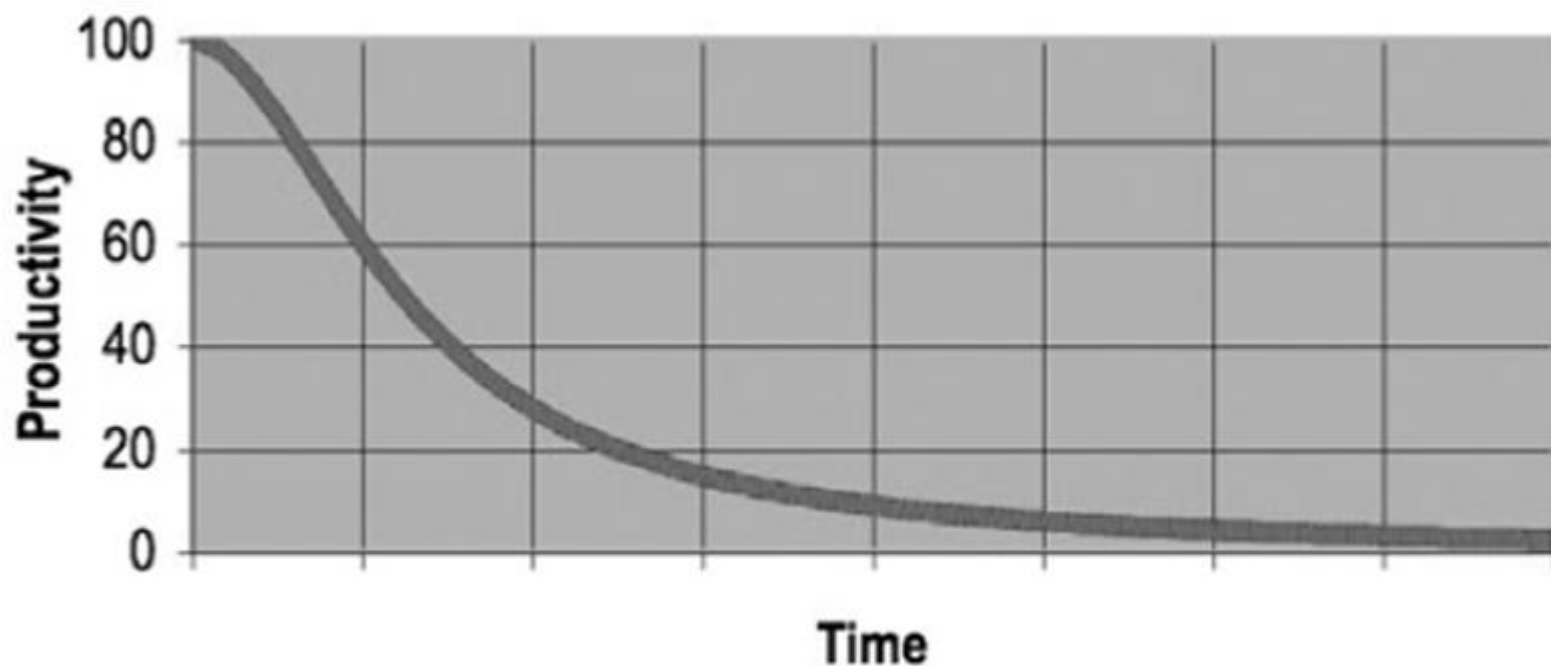


Figure 1-1
Productivity vs. time

烂代码对程序员的伤害

- ◆ 如果你不去努力编写优秀、整洁和稳定的代码，那你每天都将和糟糕的代码相伴了。
- ◆ 从此养成的一种恶习，到时你想改正的时都很困难，这是已经成为一种习惯。

台湾人民喜欢地震吗？

◆ 生于忧患死于安乐



总结一下

- ◆ 代码腐烂是随处可见的
- ◆ 惯性思维是代码腐烂的主要原因
- ◆ 生于忧患死于安乐

何谓重构

- ◆ 重构不是项目结束时，发布版本时，迭代结束时进行的
- ◆ 重构是我们每隔一个小时或半个小时就要去做的事情
- ◆ 重构是小步骤的，每小步都是微不足道的，几乎不值得去做的。

代码的坏味道

- ◆ 识别代码坏味道，是创造好代码的重要一步



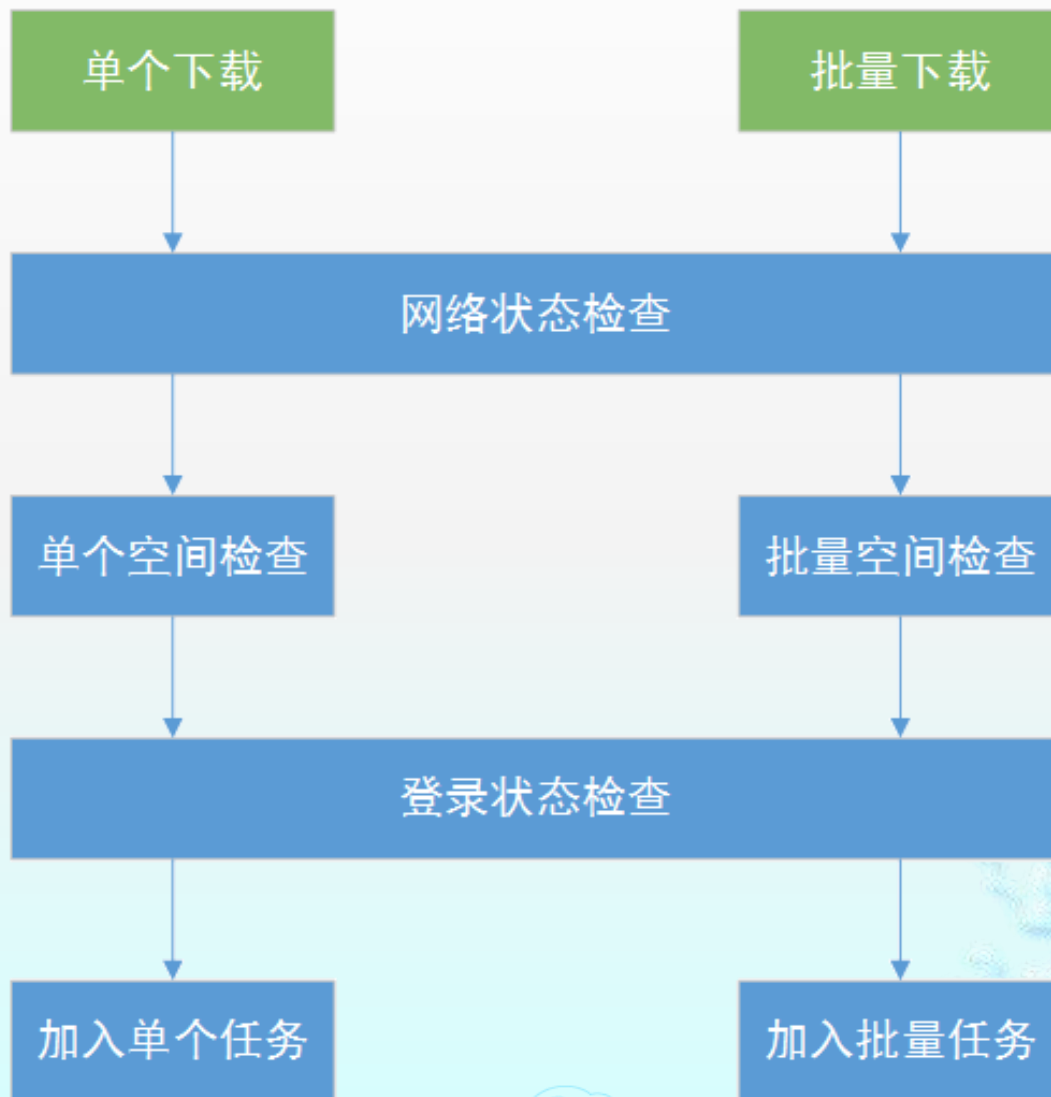
重复

- ◆ 重复可能是软件中一切邪恶的根源
- ◆ 重复意味着额外的工作量，额外的风险，额外且不必要的复杂度

重复的形式

- ◆ 重复的形式可能是多种多样的，举例说明：
 - ◆ 相似的流程
 - ◆ 相似的UI设计
 - ◆ 反复出现的代码片段

添加下载任务



Dialog UI设计



反复出现的代码片段

```
/**
 * @since 1.5
 */
public boolean contains(CharSequence cs) {
    if (cs == null) {
        throw new NullPointerException("cs == null");
    }
    return indexOf(cs.toString()) >= 0;
}
```



```
void method() {  
    if (!(score > 700)  
        || ((income >= 40000)  
            && (income <= 100000)  
            && authorized  
            && (score > 500)  
        || (income > 100000))) {  
        reject();  
    } else {  
        accepte();  
    }  
}
```

圈复杂度

最佳	1~4
可接受	5~9
不可接受	> 9

- ◆ 有能力的程序员会充分认识到自己的大脑容量是多么地有限；所以，他会非常谦卑地处理编程任务

● Edsger Dijkstra

如何降低圈复杂度

- ◆ 降低函数圈复杂度的手段通常是**抽取子函数**
- ◆ 该方案并不会降低整个系统的复杂度
- ◆ 但是会提高函数的易读性和可维护性

```
public class Animal {  
    private static final int TYPE_CHICKEN = 1;  
    private static final int TYPE_DUCK = 2;  
    private int mType;  
  
    public void setType(int type) {  
        mType = type;  
    }  
  
    public boolean canSwim() {  
        if (mType == TYPE_CHICKEN) {  
            return false;  
        } else if (mType == TYPE_DUCK) {  
            return true;  
        }  
        return false;  
    }  
  
    public void shout() {  
        if (mType == TYPE_CHICKEN) {  
            System.out.println("ge ge da");  
        } else if (mType == TYPE_DUCK) {  
            System.out.println("wang wang");  
        }  
    }  
}
```



```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    initConfig();  
    Intent intent = getIntent();  
    selectedPhotoList = intent.getStringArrayListExtra(PhotoConst.PHOTO_PATHS);  
    if (selectedPhotoList == null) {  
        selectedPhotoList = new ArrayList<String>();  
    }  
    initSelectedPhotoList = new ArrayList<String>(selectedPhotoList);  
    isContainGif = false;  
    for (String photo : selectedPhotoList) {  
        if (photo.contains("GIF")) {  
            isContainGif = true;  
            break;  
        }  
    }  
    initUI();  
}
```

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    initConfig();  
    initData();  
    initUI();  
}
```

抽象层次一致原则

- ◆ 整洁的代码如同优美的散文。
- ◆ 整洁的代码从不隐藏设计者的意图
- ◆ 充满了干净利落的抽象和直截了当的控制语句

雕版印刷术



BitmapManager.java

```
void updateImageView(ImageView imageView, String url) {  
    Bitmap bitmap = extractBitmap(url);  
    imageView.setImageBitmap(bitmap);  
}  
  
void updateBackground(View view, String url) {  
    Bitmap bitmap = extractBitmap(url);  
    view.setBackground(new BitmapDrawable(bitmap));  
}
```


BitmapManager.java

```
void extractBitmap(String url, IGetBitmapCallback callback) {  
    ...  
    Bitmap bitmap = extractBitmap(url);  
    if (callback != null) {  
        callback.onGetBitmap(bitmap);  
    }  
}
```

单一职责原则

- ◆ 高内聚，低耦合是灵活的前提
- ◆ 不能有求必应



```
public List<int[]> getThem() {  
    List<int[]> list1 = new ArrayList<int[]>();  
    for (int[] x : mTheList)  
        if (x[0] == 4)  
            list1.add(x);  
    return list1;  
}
```

```
public List<int[]> getFlaggedCells() {  
    List<int[]> flaggedCells = new ArrayList<int[]>();  
    for (int[] cell : mGameBoard)  
        if (cell[STATUS_VALUE] == FLAGGED)  
            flaggedCells.add(cell);  
    return flaggedCells;  
}
```

```
public List<Cell> getFlaggedCells() {  
    List<Cell> flaggedCells = new ArrayList<Cell>();  
    for (Cell cell : mGameBoard)  
        if (cell.isFlagged)  
            flaggedCells.add(cell);  
    return flaggedCells;  
}
```


模糊度

- ◆ 命名模糊
- ◆ 魔法数字



《韦氏学院词典（第11版）》。我去哪里都带着它。这倒不是你实际上要读的东西，但是我说过，写程序的时候，必须能命名好变量。你的文笔必须好。没有好的字典，我就会觉得少了点儿什么。

■ Joshua Bloch

Google首席
Java架构师



Sun 公司核心技术丛书

Effective Java 中文版 第2版

Effective Java **Second Edition**

蔚蓝网



(美) Joshua Bloch 著
杨春花 俞黎敏 译

“我很希望10年前就拥有这本书。可能有人认为我不需要任何Java方面的书籍，但是我需要这本书。”

Java之父James Gosling



机械工业出版社
China Machine Press

不好的命名会带来什么样的问题？

```
File newFile = new File(f.getAbsolutePath() + SUFFIX);  
writeDataToNewFile(f, data);
```



```
public void showLoadingView(boolean flag) {  
    if (flag) {  
        mLoadingProgress.startLoading();  
        mLoadingProgress.setVisibility(View.VISIBLE);  
    } else {  
        mLoadingProgress.setVisibility(View.GONE);  
        mLoadingProgress.stopLoading();  
    }  
}
```

布尔变量勿用中性词

```
public interface DialogInterface {  
  
    public static final int BUTTON_POSITIVE = -1;  
  
    public static final int BUTTON_NEGATIVE = -2;  
  
    public static final int BUTTON_NEUTRAL = -3;  
  
    @Deprecated  
    public static final int BUTTON1 = BUTTON_POSITIVE;  
  
    @Deprecated  
    public static final int BUTTON2 = BUTTON_NEGATIVE;  
  
    @Deprecated  
    public static final int BUTTON3 = BUTTON_NEUTRAL;  
}
```


冗余

- ◆ 问：做哪些事情可以提升生活品质
- ◆ 答：定期扔东西，相信我！

■ 知乎经典问答



```
public ImageInfo createFromParcel(Parcel source) {  
    ImageInfo imageInfo = new ImageInfo();  
    imageInfo.thumbExist = source.readInt() == 0 ? false : true;  
    imageInfo.bigImgExists = source.readInt() != 0 ? true : false;  
    imageInfo.thumbnail = source.readString();  
    imageInfo.filePath = source.readString();  
    imageInfo.toUin = source.readString();  
    imageInfo.progress = source.readInt();  
    imageInfo.showProgress = source.readInt() == 0 ? false : true;  
    imageInfo.decode = source.readInt() == 0 ? false : true;  
    imageInfo.isLargePhoto = source.readInt() == 0 ? false : true;  
    imageInfo.compressed = source.readInt() == 1 ? true : false;  
    imageInfo.secretfileShot = source.readInt() == 1 ? true : false;  
    imageInfo.isGif = source.readInt() != 1 ? false : true;  
    return imageInfo;  
}
```

```
public ImageInfo createFromParcel(Parcel source) {  
    ImageInfo imageInfo = new ImageInfo();  
    imageInfo.thumbExist = source.readInt() == 1;  
    imageInfo.bigImgExists = source.readInt() == 1;  
    imageInfo.thumbnail = source.readString();  
    imageInfo.filePath = source.readString();  
    imageInfo.toUin = source.readString();  
    imageInfo.progress = source.readInt();  
    imageInfo.showProgress = source.readInt() == 1;  
    imageInfo.decode = source.readInt() == 1;  
    imageInfo.isLargePhoto = source.readInt() == 1;  
    imageInfo.compressed = source.readInt() == 1;  
    imageInfo.secretfileShot = source.readInt() == 1;  
    imageInfo.isGif = source.readInt() == 1;  
    return imageInfo;  
}
```

冗余的代码

- ◆ 每段代码都得有人去理解它、维护它，这些工作都是要花钱的。
- ◆ 如果一个类的所得不值其身价，请让这个类**庄严赴义**。

```

//  /**
//   * 登录会员
//   * */
//   public String getTipAfterLoginVip(String hasTicket,int videoPaystatus,int isup,String displayStyle){
//       return  getTipAfterLogin(TAG_YES, hasTicket, videoPaystatus, isup, displayStyle);
//   }
//
//  /**
//   * 登录非会员
//   * */
//   public String getTipAfterLoginNoVip(String hasTicket,int videoPaystatus,int isup,String displayStyle){
//       return  getTipAfterLogin(TAG_NO,hasTicket, videoPaystatus, isup, displayStyle);
//   }

public String getRightTips(String islogin ,String  isVip,String hasTicket,int movie_type,String versionname,int isup,String displayStyle){
    if (haolaiwuList != null) {
        for (Iterator<PayVipEntity> iterator = haolaiwuList.iterator(); iterator.hasNext(); ) {
            PayVipEntity entity = iterator.next();
            if (entity.getIs_login().equals(islogin) && entity.getDisplay_style().equals(displayStyle) && entity.getIs_vip().equals(isVip)
                //精确匹配到了
                return getEntityTips(entity, isup);
            }
        }
    }
    return Constants.EMPTY;
}

//   public static final int PayStatus_ChargeFor_SingleVideo = 4; //单片点播（会员用券，非会员支持单点，用券）3
//   public static final int PayStatus_ChargeFor_SingleExclusiveVIP = 5; //单片点播，会员免费（会员免费，非会员可以单点，用券）2
//   public static final int PayStatus_OnlyFreeFor_VIP = 6; //会员免费（必须开通会员）1

```

冗余的注释

- ◆ 仅在非常关键的代码添加注释

```
/**  
 * 登录后提示  
 * */  
public String getTipAfterLogin(String isVip,String hasTicket,int videoPaystatus,ir  
    return  getTip(TAG_YES, isVip, hasTicket, videoPaystatus, isup, displayStyle);  
}
```


代码之美

我喜欢最小化的代码，那是姿势优美、井井有条的代码。假设你已开始删减。当删到不能再删的时候，也就是说多删掉一点东西就不能再工作了，这个时候的代码就美了。当你能够想到的任何修改都只会让算法变差，那么这个时候的代码就叫美。

■ Armstrong
英国机器人学会创始人

腾讯视频的MarkLabelView

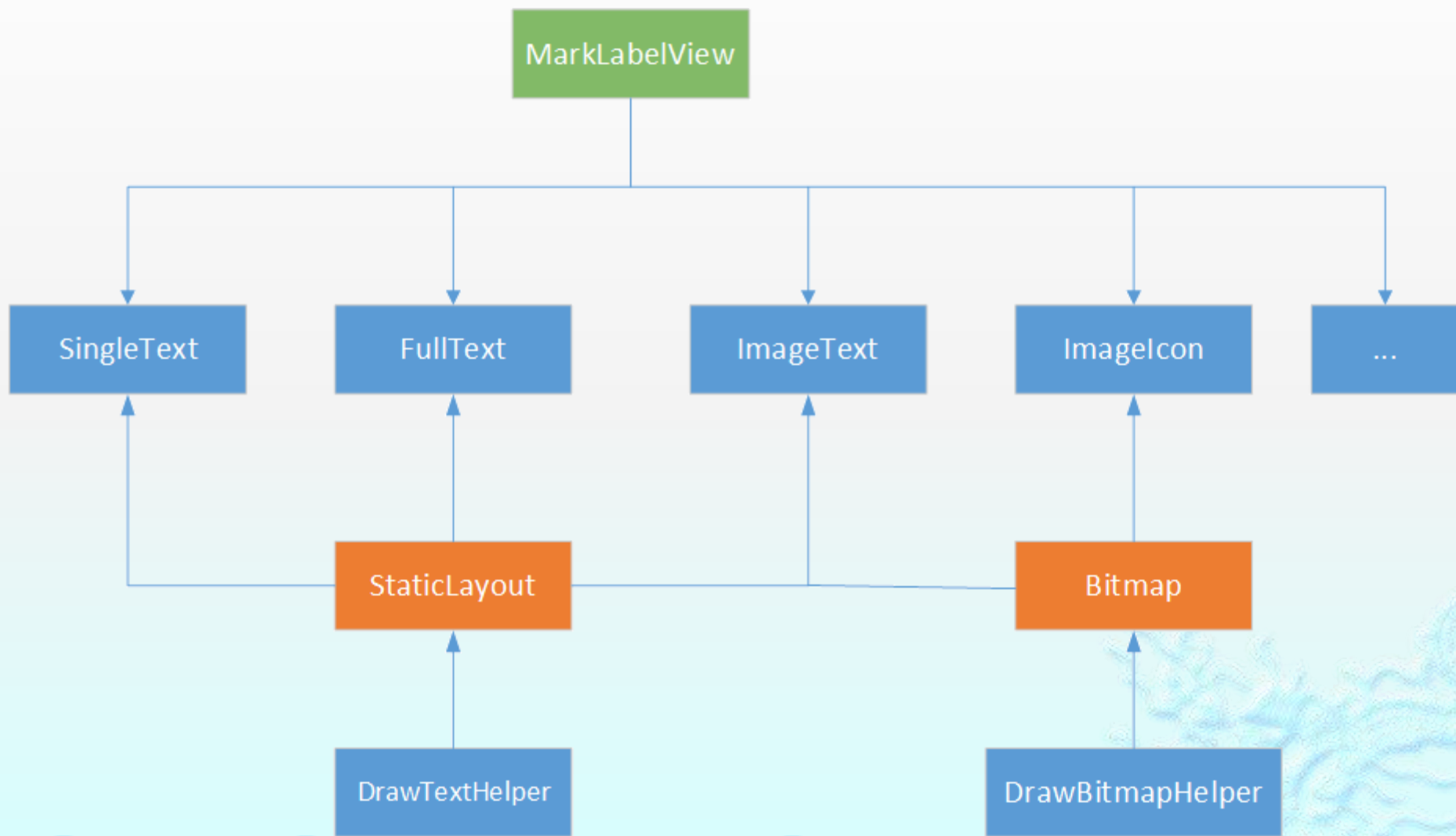


- ◆ 角标可以组合出现
- ◆ 每个角标有两个属性：位置和类型
- ◆ 项目目前一共有6种位置，7种类型

腾讯视频的MarkLabelView



腾讯视频的MarkLabelView



总结一下

- ◆ 重复
- ◆ 圈复杂度
- ◆ 抽象层次一致
- ◆ 单一职责
- ◆ 模糊度
- ◆ 冗余

代码自检工具

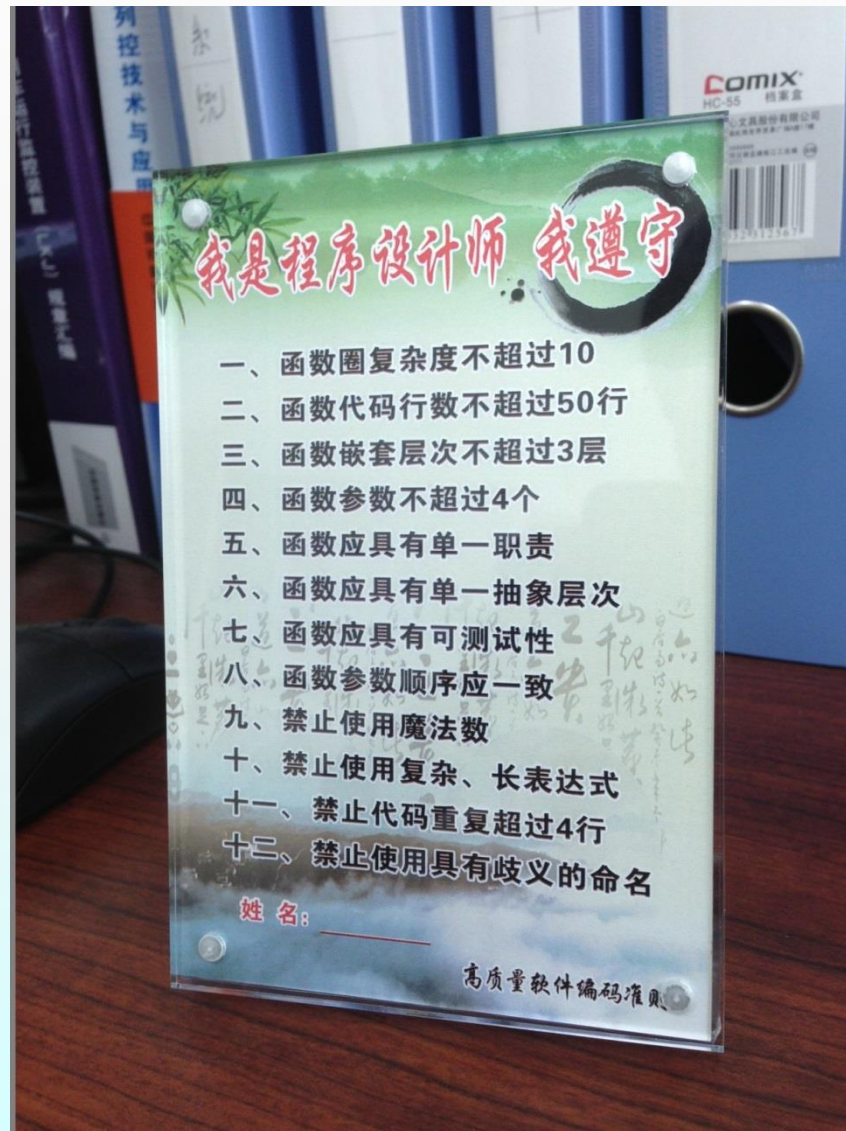
- ◆ Android Studio的自带工具
- ◆ 阿里巴巴Java代码检查插件

重构菜单

- ◆ 重命名 Alt + Shift + R
- ◆ 提取函数 Alt + Shift + M
- ◆ 提取常量 Ctrl + Alt + C
- ◆ 提取局部变量 Alt + Shift + L
- ◆ 修改函数签名 Alt + Shift + C
- ◆ 移动（常量） Alt + Shift + V
- ◆ 内联（inline） Alt + Shift + I

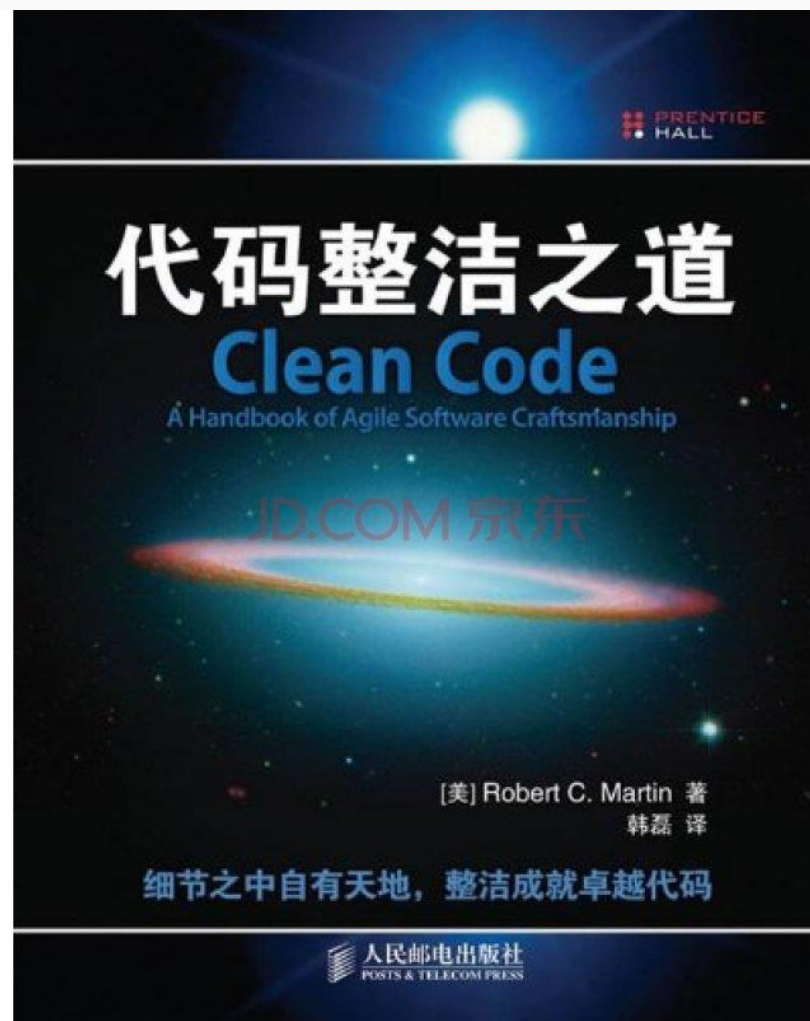
Code Review

◆ Check list

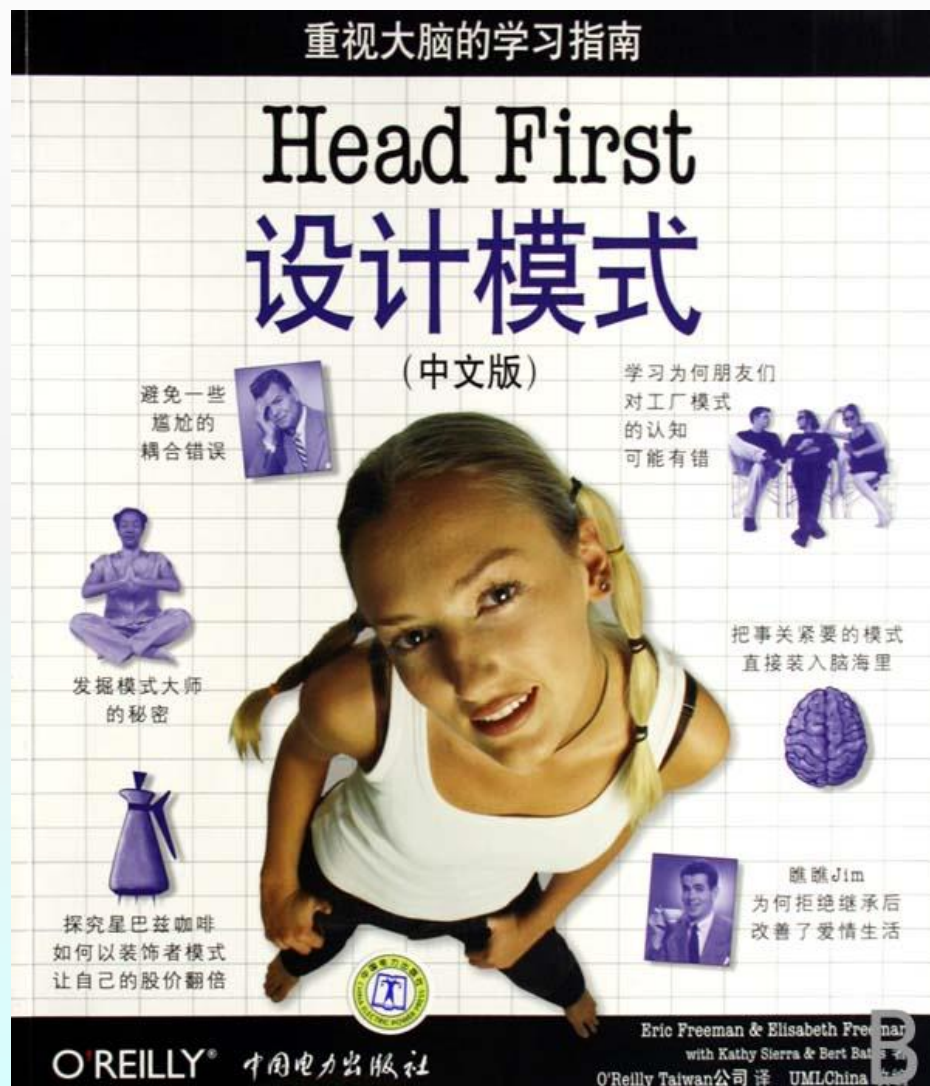


好书推荐

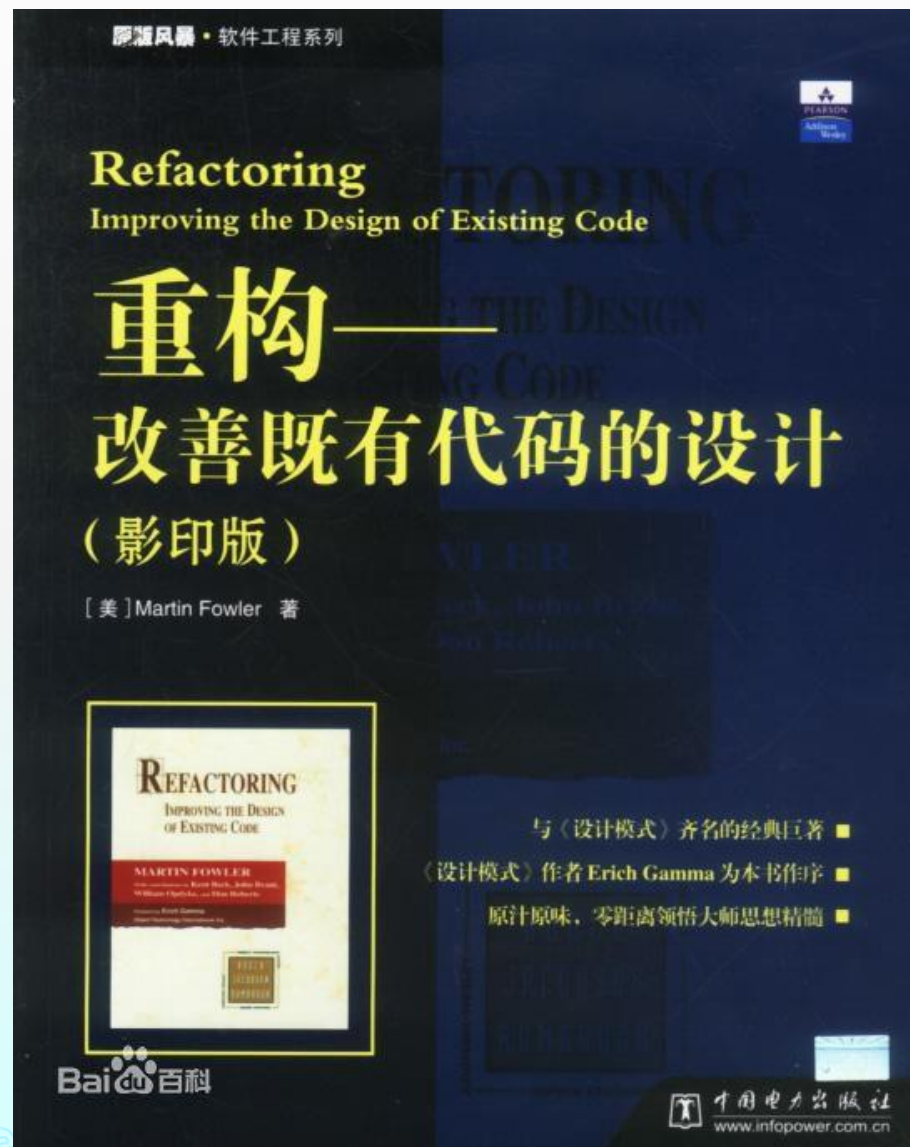
◆ 代码质量必读书



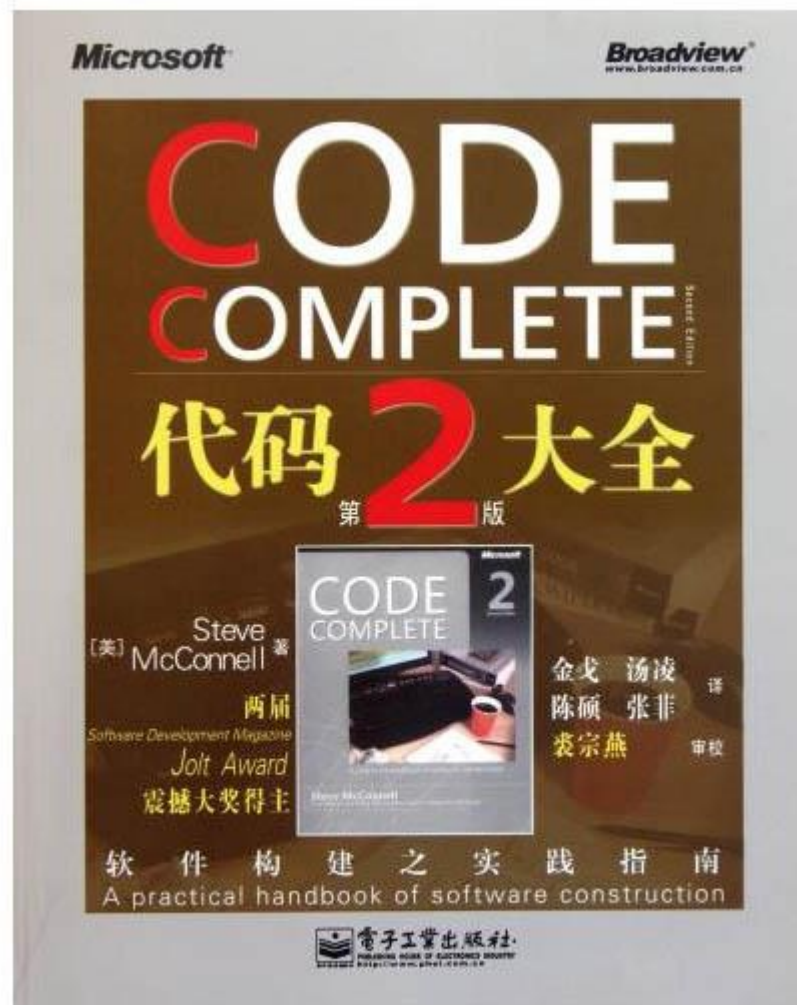
◆ 轻松版设计模式



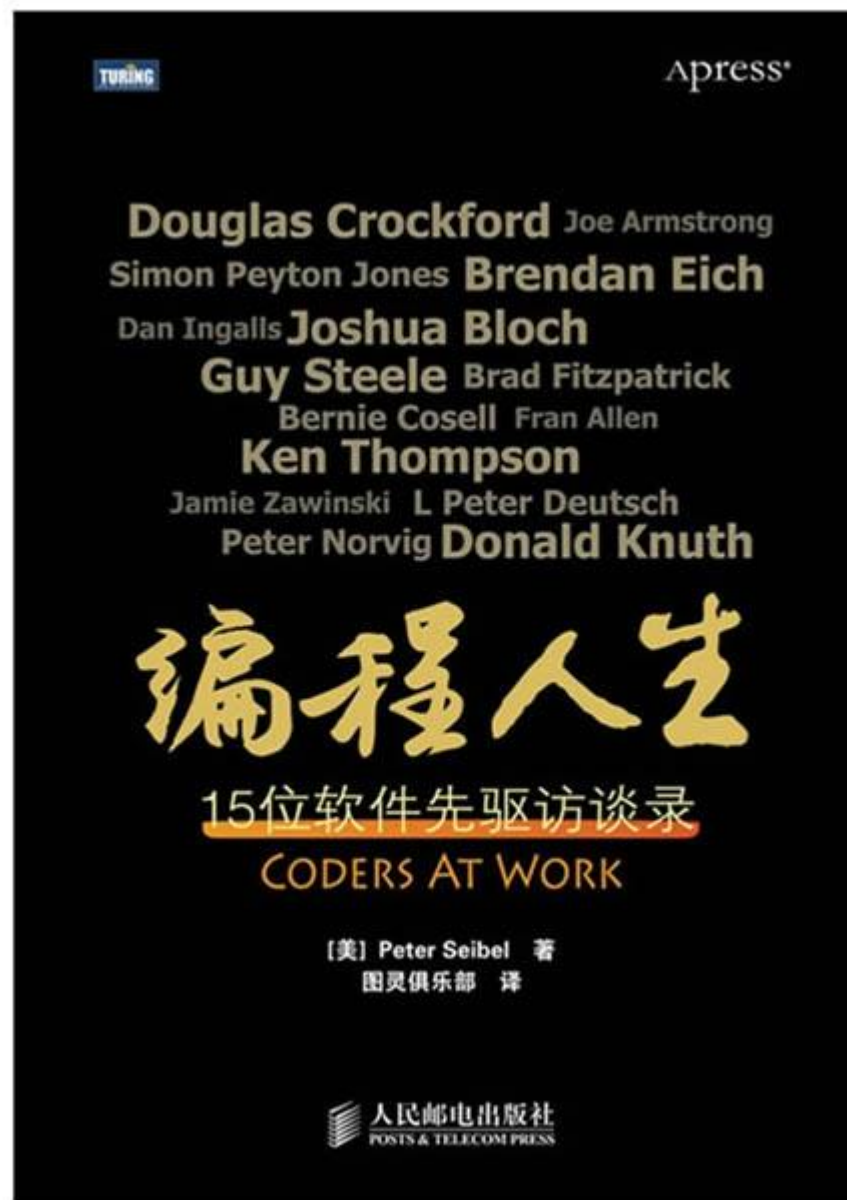
◆ 代码的坏味道出处



◆ 代码质量入门书



◆ 向大牛看齐





Thanks

