《软件工程》

王可泽

中山大学计算机学院 副教授

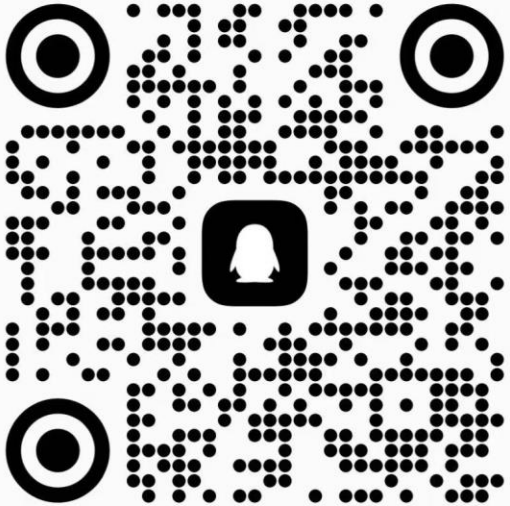
邮箱：kezewang@gmail.com

软件工程 0226签到

微信扫码签到

软件工程20...

群号: 933428733



扫一扫二维码，加入群聊

课程简介

本课程将系统的介绍现代软件工程的基本概念、原理、方法、工具和最佳实践，展示如何开发优秀健壮的软件，其完整的软件生命周期和过程。

1. 掌握软件生命周期模型，了解软件项目管理，掌握敏捷软件开发方法
2. 掌握软件需求工程基本原理和方法，懂得产品定义和原型设计
3. 掌握软件架构设计基本原理和方法，了解设计原则、设计模式，了解云计算与微服务
4. 掌握软件系统的安全与隐私的基本原理
5. 掌握软件测试的基本原理与方法，了解测试驱动开发（TDD）、需求驱动开发（BDD）

考核

1. 平时成绩：考勤 10%；作业 30%
2. 期末大作业成绩：60%

程序及其质量保证方法

内容

1. 程序及质量要求

✓程序及其内部和外部质量

2. 程序质量保证方法

✓编码规范、设计方法、代码重用、结对编程

3. 程序质量的分析方法

✓人工审查、自动化分析、代码测试

4. 编写程序需要解决的问题

1.1 何为程序?

- **程序（Program）是由程序设计语言所描述的、能为计算机所理解和处理的一组语句序列**
  - ✓用程序设计语言（Programming Language）来描述的
  - ✓如Java、C、C++、Python
- **程序严格遵循程序设计语言的各项语法和语义规定**
  - ✓确保程序代码能为程序设计语言的编译器所理解，进而编译生成相应的可运行代码
- **程序代码可表现为二种形式**
  - ✓源代码（Source Code）：用程序设计语言所描述的代码
  - ✓可执行代码（Executable Code）：可执行的二进制或中间码

程序代码示例

```
import java.io.*;

public class Main {
    public static void main(String[] args)  {
        try {
            BufferedReader in = new BufferedReader(new FileReader("test.log"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            System.out.println(str);
        } catch (IOException e) {
        }
    }
}
```

用Java语言编写的代码

1.2 程序组成-语句

❑程序中的语句

- ✓**声明、定义、控制、计算**等
- ✓实现特定的功能
- ✓用程序设计语言来描述

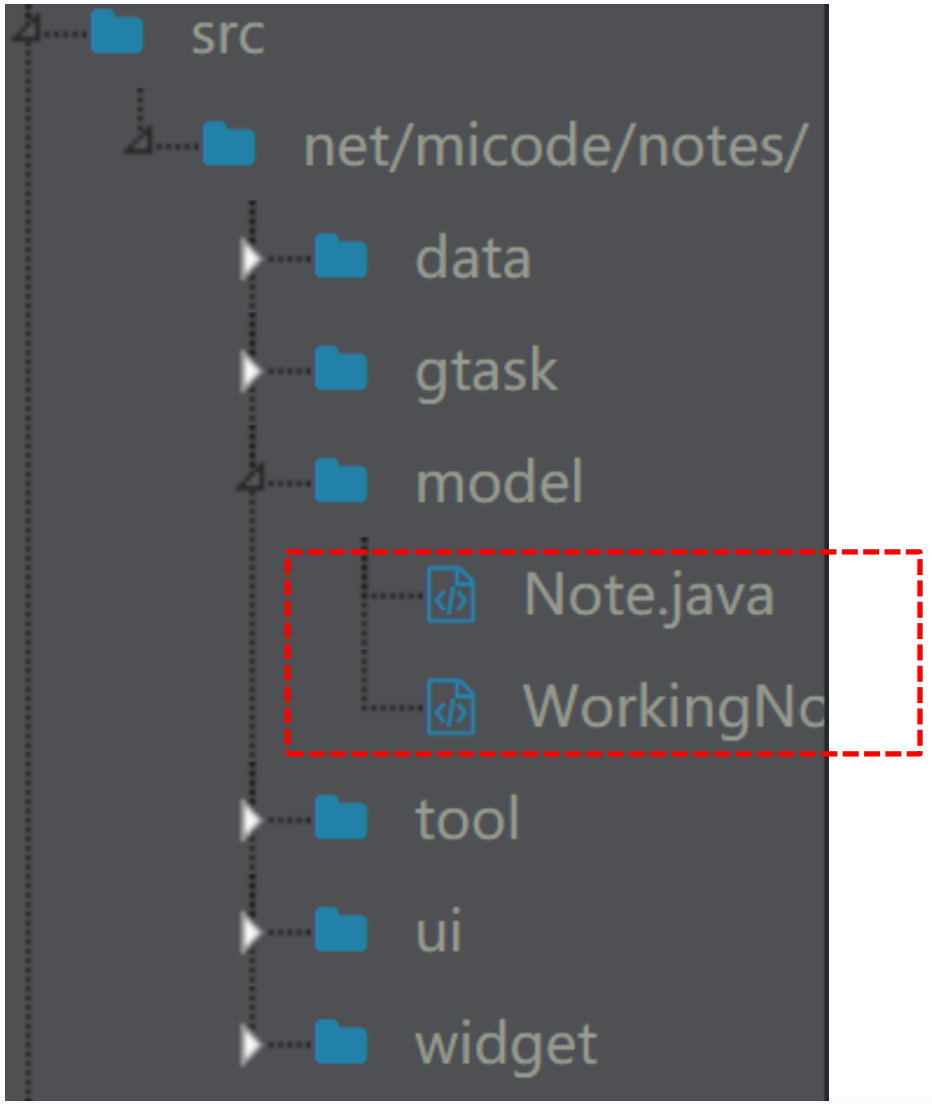
❑计算机可以理解

- ✓可编译成二进制代码
- ✓最终可执行

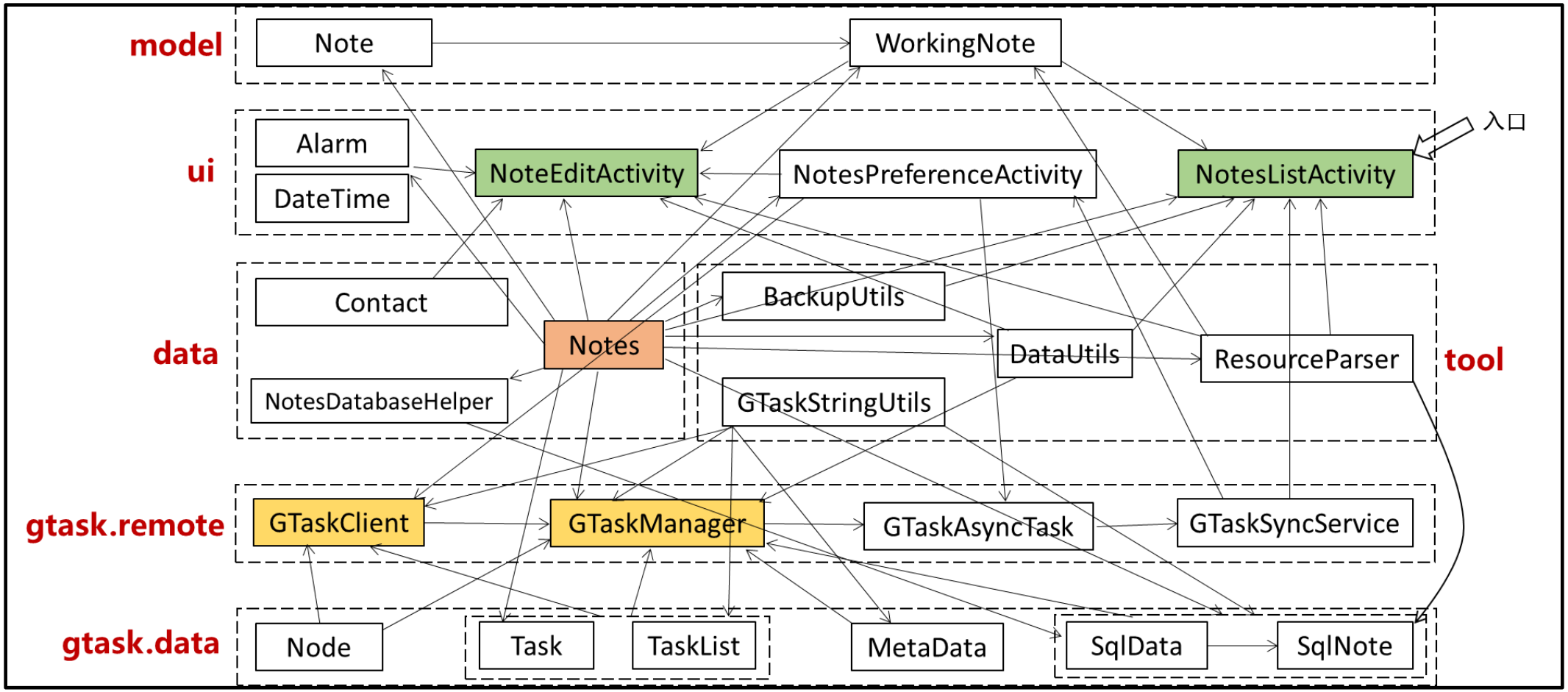
```
/*
 * Copyright (c) 2010-2011, The MiCode Open Source Community (www.micode.net)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *        http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package net.micode.notes.data;

import android.content.Context;

public class Contact {
    private static HashMap<String, String> sContactCache;
    private static final String TAG = "Contact";

    private static final String CALLER_ID_SELECTION = "PHONE_NUMBERS_EQUAL(" + Phone.NUMBER
    + ",?) AND " + Data.MIMETYPE + "='" + Phone.CONTENT_ITEM_TYPE + "'"
    + " AND " + Data.RAW_CONTACT_ID + " IN "
            + "(SELECT raw_contact_id "
            + " FROM phone_lookup"
            + " WHERE min_match = '+')";

    public static String getContact(Context context, String phoneNumber) {
        if(sContactCache == null) {
            sContactCache = new HashMap<String, String>();
        }

        if(sContactCache.containsKey(phoneNumber)) {
            return sContactCache.get(phoneNumber);
        }

```

程序组成-模块

- 由诸多相互交互的**模块**组成
  - ✓包(Package)
  - ✓类(Class)
  - ✓方法(Method)
- 每个模块实现特定**功能**
- 示例：小米便签中的模块
  - ✓170个文件、471个方法



示例：小米便签的模块结构



1.3 程序如何编写?

要实现的功能

采用编程语言

逻辑思考推理

程序设计经验

逻辑思考、认知和推理的过程

编写程序

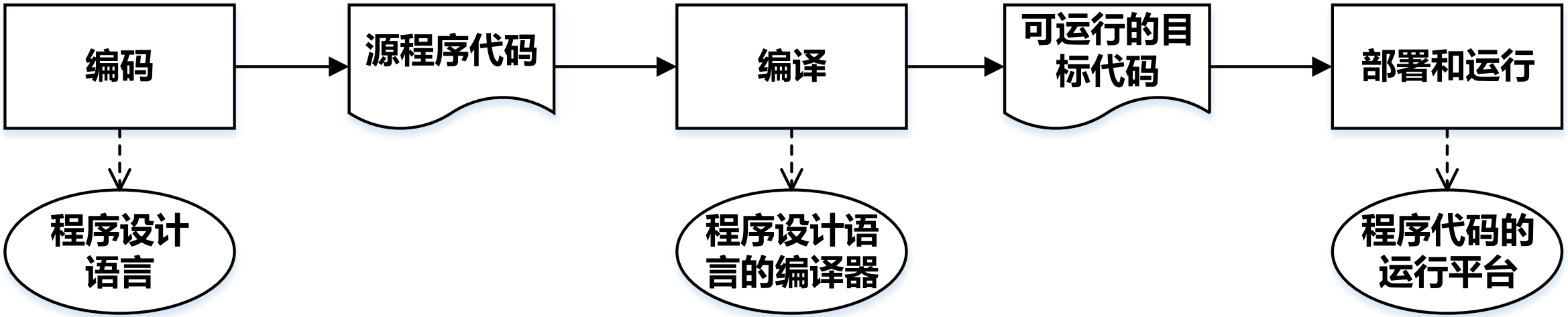
程序代码

你是如何编写程序的?

程序的编辑、编译、部署和运行

如何保证质量问题？！

基于用户需求 ----------> **产生运行结果**

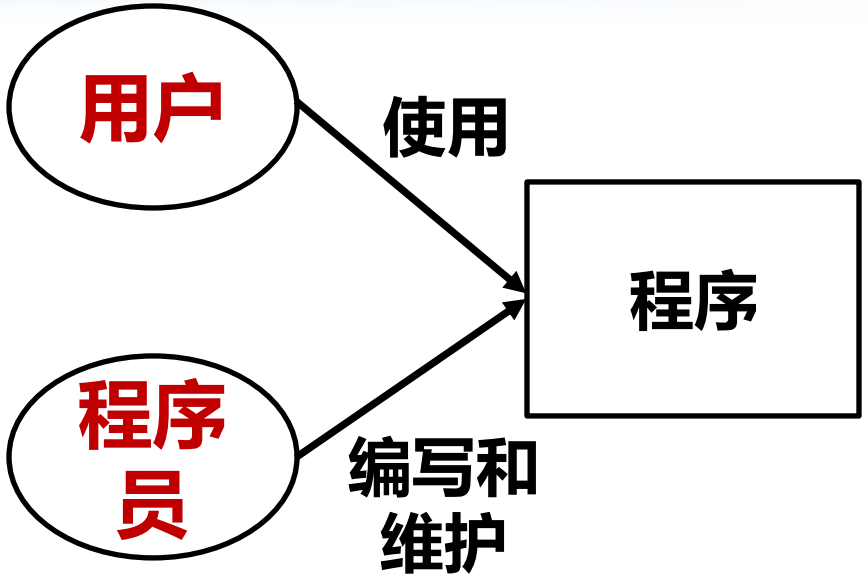


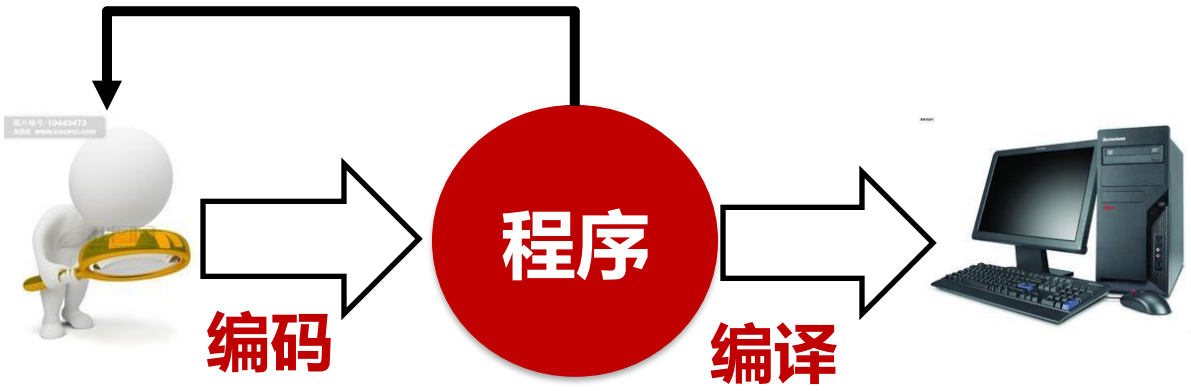
产品的质量问题

- **外在：用户直接可以感觉到的**
  - ✓**使用用户**：圆滑、手感好、美观、不死机、操作便捷
- **内在：用户无法直接感触到的**
  - ✓**维修人员**：易于维护、易于更换

1.4 程序的二类利益相关者

- **用户**
  - ✓程序的运行展示**功能和性能**
  - ✓满足和实现用户**的需求**
- **程序员**
  - ✓编写、阅读和维护程序
  - ✓发现和修改程序中的**缺陷**



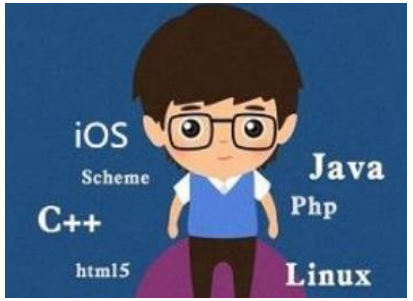


对程序的不同质量需求

用户角度

- 正确性
- 高效性
- 可靠性
- 友好性
- ……

程序员角度



- 可理解
- 易修改
- 可维护
- 可重用
- ……

用户和程序员会对程序质量分别提出什么样的要求？为什么？

1.5 程序质量的内在和外在体现

❑外在(External)质量

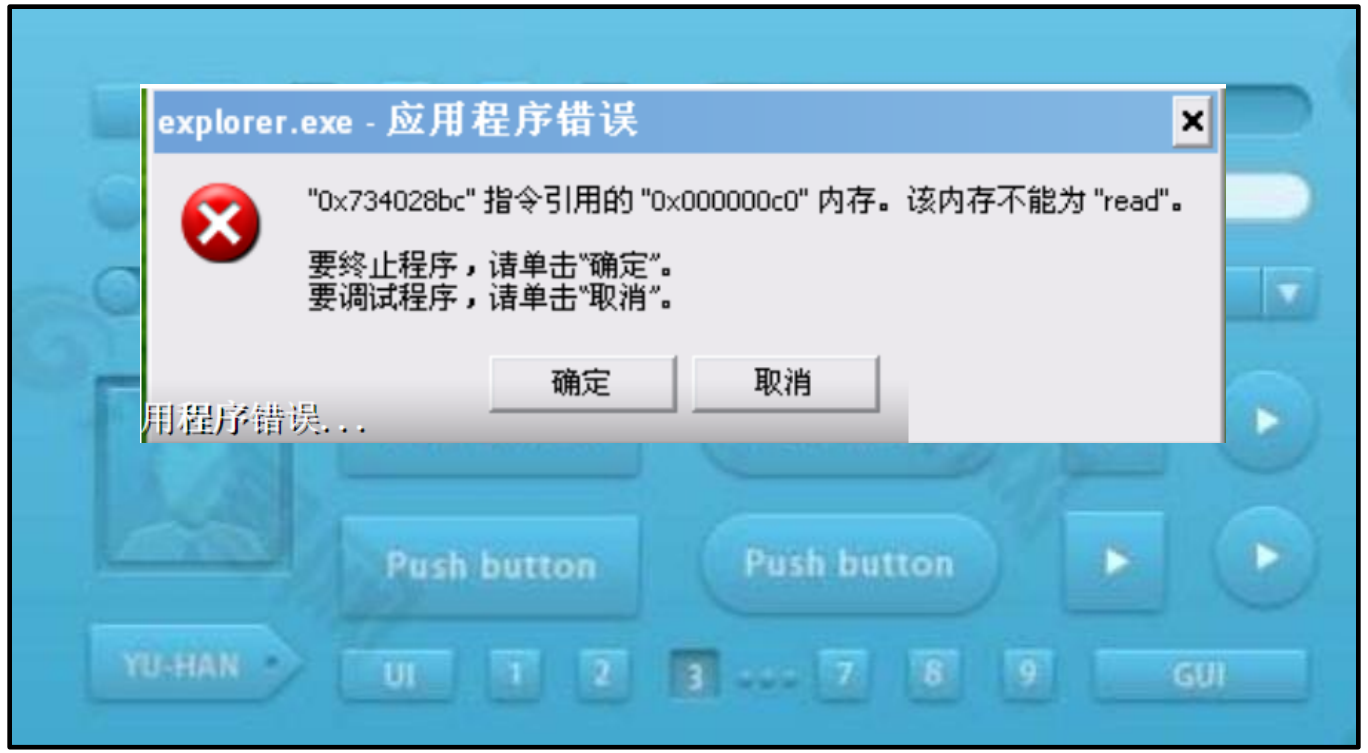
- ✓**用户**视点
- ✓对外可展示，用户可直接感触到、所关心的
- ✓**使用流畅性、响应速度、界面美观、操作简易性、运行可靠性**等

❑内在(Internal)质量

- ✓**程序员**视点
- ✓体现在程序的内在方面，程序员可以感触到的、所关心的
- ✓**易于理解、结构清晰、易于修改、可重用好**等

你是否对你或他人编写的程序有质量方面的要求和抱怨?

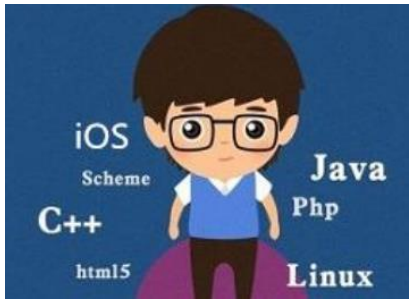
示例：从用户和程序员视角看程序质量



运行效果如何?

- ✓ 可靠吗?
- ✓ 易于操作和使用吗?
- ✓ ……

```
#include "stdafx.h"
#include "stdio.h"
void test();
int _tmain
(int argc,
_TCHAR* argv[])
{ test(); return
0; } char C[25]
[40];void d(int x,
int y) {C[x][y]=
C[x][y+1]=32;}
int f(int x){return
(int)x*x*.08;}
void test(){int i,j;
char s[5]="TEST";
for(i=0;i<25;i++)
for(j=0;j<40;j++)
C[i][j]=s[(i+j)%4];
for(i=1;i<=7;i++)
{d(18-i,12);
C[20-f(i)][i+19]=
C[20-f(i)][20-i]=32;
}d(10,13);d(9,13);
d(8,14);d(7,15);
d(6,16);d(5,18);d(5,20);
d(5,22);d(5,26);
d(6,23);d(6,25);d(7,25);for(i=0;i<25;i++,printf("\n"))
for(j=0;j<40;printf("%c",C[i][j++]));
scanf("%c", &s[0]);
}
```



这段代码如何?

- ✓ 易于理解吗?
- ✓ 易于修改吗?
- ✓ 结构清晰吗?
- ✓ ……

示例：程序代码质量

```
1  package MarDetecter;
2
3
4  import jade.core.Agent;
10
11 public class DigMineRobot extends BasicRobot {
12     private Coordinate startposi = null;
13     private MarUI ui = null;
14     private Mine mine = null;
15
16
17     public DigMineRobot() {
18         ui = MarUI.getUI();
19         mine = Mine.getMine();
20         startposi = new Coordinate(MarUI.GRID_SIZE - 1, MarUI.GRID_SIZE - 1);
21     }
22
23
24
25     public void setup() {
26         System.out.println("开始采矿！");
27         ui.runInfo.setText(ui.runInfo.getText() + "开始采矿！" + "\n");
28
29         addBehaviour(new CooperationBev(this));
30         ACLMessage msg = this.receive();
31         addBehaviour(new DigMineBev(this));
32         addBehaviour(new DumpMineBev(this));
33
34
35     }
36 }
37
```

这段程序写的如何？

- ✓ 易于理解吗？
- ✓ 结构清晰吗
- ✓ 易于修改吗？

……

为什么？

……

思考和讨论

- 你觉得什么样的程序是高质量?
- 你编写程序时是否考虑了质量问题?
- 你所写的程序质量如何?
- 如何才能编写出高质量的程序?

内容

1. 程序及质量要求
   - ✓程序及其内部和外部质量
2. **程序质量保证方法**
   - ✓编码规范、设计方法、代码重用、结对编程
3. **程序质量的分析方法**
   - ✓人工审查、自动化分析、代码测试
4. **编写程序需要解决的问题**

2.1 程序质量的语法和语义体现

❑代码风格规范-语法

- ✓语法和结构层次
- ✓明确如何来规范程序的书写
- ✓表现为是否易于阅读和理解

❑代码设计规范-语义

- ✓语义和内涵层次，外在的
- ✓明确如何来组织和封装程序语句
- ✓表现为良好的结构和易于重用

```
1  package MarDetecter;
2
3
4  import jade.core.Agent;
10
11 public class DigMineRobot extends BasicRobot {
12     private Coordinate startposi = null;
13     private MarUI ui = null;
14     private Mine mine = null;
15
16
17     public DigMineRobot() {
18         ui = MarUI.getUI();
19         mine = Mine.getMine();
20         startposi = new Coordinate(MarUI.GRID_SIZE - 1, MarUI.GRID_SIZE - 1);
21     }
22
23
24
25     public void setup() {
26         System.out.println("开始采矿！");
27         ui.runInfo.setText(ui.runInfo.getText() + "开始采矿！" + "\n");
28
29         addBehaviour(new CooperationBev(this));
30         ACLMessage msg = this.receive();
31         addBehaviour(new DigMineBev(this));
32         addBehaviour(new DumpMineBev(this));
33
34
35     }
36 }
37
```

程序质量保证方法

- 遵循编码风格
- 采用程序设计方法
- 开展代码重用
- 进行结对编程

2.2 遵循编码风格

❑良好的编程行为

- ✓在编码过程中，程序员对代码符号进行良好的组织、合理的命名、提供必要的注释，那么将可增强代码的可读性和可理解性，进而提高代码的可维护性和可重用性，提升代码的内部质量

❑何为编码风格

- ✓程序员在编码时要遵循特定的样式及要求，以规范程序员的编程行为以及所产生程序代码的样式

示例：遵循编码风格

```
#include "stdafx.h"
#include "stdio.h"
void test ();
int _tmain
(int argc,
_TCHAR* argv[])
{ test(); return
0; } char C[25]
[40];void d(int x,
int y) {C[x][y]=
C[x][y+1]=32;}
int f(int x){return
(int)x*x*.08;}
void test(){int i,j;
char s[5]="TEST";
for(i=0;i<25;i++)
for(j=0;j<40;j++)
C[i][j]=s[(i+j)%4];
for(i=1;i<=7;i++)
{d(18-i,12);
C[20-f(i)][i+19]=
C[20-f(i)][20-i]=32;
}d(10,13);d(9,13);
d(8,14);d(7,15);
d(6,16);d(5,18);d(5,20); d(5,22);d(5,26);
d(6,23);d(6,25);d(7,25);for(i=0;i<25;i++,printf("\n"))
for(j=0;j<40;printf("%c",C[i][j++]));
scanf("%c", &s[0]);
```

没有遵循编码风格

```
/*
 * Copyright (c) 2010-2011, The MiCode Open Source Community (www.micode.net)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *        http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package net.micode.notes.data;

import android.content.Context;

public class Contact {
    private static HashMap<String, String> sContactCache;
    private static final String TAG = "Contact";

    private static final String CALLER_ID_SELECTION = "PHONE_NUMBERS_EQUAL(" + Phone.NUMBER
    + ",?) AND " + Data.MIMETYPE + "='" + Phone.CONTENT_ITEM_TYPE + "'"
    + " AND " + Data.RAW_CONTACT_ID + " IN "
            + "(SELECT raw_contact_id "
            + " FROM phone_lookup"
            + " WHERE min_match = '+')";

    public static String getContact(Context context, String phoneNumber) {
        if(sContactCache == null) {
            sContactCache = new HashMap<String, String>();
        }

        if(sContactCache.containsKey(phoneNumber)) {
            return sContactCache.get(phoneNumber);
        }

```

遵循编码风格

编写代码的基本原则

- **易读，一看就懂**
  - ✓理解代码的内涵和意图
  - ✓望文生义的符号、缩进和括号、代码注释，遵循编码风格
- **简明，减低复杂度**
  - ✓避繁就简、不用goto语句、减少嵌套层数、简单算法
- **易改，便于维护**
  - ✓便于修改程序代码、增加新的代码
  - ✓封装、参数化、模块化、隐藏、常元
- **无二义，不产生歧义**
  - ✓不要让人产生误解

编码风格-代码布局

- 缩进，用好Tab键
- 用括号来表示优先级
- 断行处的{}

```
if (condition) {
    DoSomething();
} else {
    DoSomething();
}
```

- 一行至多只一条语句
  - ✓不要在一行定义多个变量

```
146     public void onDismiss(DialogInterface dialog) {
147         stopAlarmSound();
148         finish();
149     }
150
151     private void stopAlarmSound() {
152         if (mPlayer != null) {
153             mPlayer.stop();
154             mPlayer.release();
155             mPlayer = null;
156         }
157     }
158 }
```

编码风格-代码组织

- 按一定次序来说明数据
- 按字母顺序说明对象
- 尽可能避免使用嵌套结构
- 采用统一的缩进规则
- 单入口单出口

```
91      private void playAlarmSound() {
92          Uri url = RingtoneManager.getActualDefaultRingtoneUri(this,
    RingtoneManager.TYPE_ALARM);
93
94          int silentModeStreams = Settings.System.getInt(getContentResolver(),
95                  Settings.System.MODE_RINGER_STREAMS_AFFECTED, 0);
96
97          if ((silentModeStreams & (1 << AudioManager.STREAM_ALARM)) != 0) {
98              mPlayer.setAudioStreamType(silentModeStreams);
99          } else {
100             mPlayer.setAudioStreamType(AudioManager.STREAM_ALARM);
101         }
102         try {
103             mPlayer.setDataSource(this, url);
104             mPlayer.prepare();
105             mPlayer.setLooping(true);
106             mPlayer.start();
107         } catch (IllegalArgumentException e) {
108             // TODO Auto-generated catch block
```

编码风格-命名规范

- 采用有意义、一目了然的命名方式
  - ✓变/常/函数/方法/类/包
  - ✓一看就懂，望文生义
- 无意义的命名
  - ✓i, j, x, y
- 有意义的命名
  - ✓szPath
  - ✓vPrintName( )
  - ✓bReturn

```
38  public class NotesProvider extends ContentProvider {
39      private static final UriMatcher mMatcher;
40
41      private NotesDatabaseHelper mHelper;
42
43      private static final String TAG = "NotesProvider";
44
45      private static final int URI_NOTE            = 1;
46      private static final int URI_NOTE_ITEM       = 2;
47      private static final int URI_DATA            = 3;
48      private static final int URI_DATA_ITEM       = 4;
49
50      private static final int URI_SEARCH          = 5;
51      private static final int URI_SEARCH_SUGGEST  = 6;
52
```

编码风格-命名

❑命名通则

- ✓使用**英文单词或缩写**，不要使用拼音
- ✓**望文知义**原则，含义清晰、明确
- ✓命名**不要过长**
- ✓尽量使用**全称，少用缩写**

■ 不规范的命名

✓DaYinWenJian 与 PrintFile

```
public class LoginManager{
    private UserLibrary userLib;

    public void LoginManager( ) ; //构造函数
    public int login(account, password) ; //用户登录
    public boolean isUserValid(String account, String password);    //判断用户是否合法
}
```

编码风格-命名

❑命名和大小写

- ✓类/类型/变量：**用名词和名词短语**，所有单词第一个字母大写
- ✓函数/方法：**动词或者动词短语**，第一个单词小写，随后单词第一个字母大写
- ✓示例：Member, ProductInfo, getName(), setName(), renderPage()

❑不规范的命名

- ✓变量：print;
- ✓方法：Print()

```
public class LoginManager{
    private UserLibrary userLib;

    public void LoginManager( ) ; //构造函数
    public int login(account, password) ; //用户登录
    public boolean isUserValid(String account, String password);    //判断用户是否合法
}
```

编码风格-代码注释

- **帮助理解程序**
  - ✓注释要说明程序：(1) **做什么**；(2) **为什么这么做**；(3) **注意事项**
  - ✓无需解释程序如何做
- **注解位置**
  - ✓**类头、函数/函数头、关键语句块头、关键语句尾**
- **有效、必要、简洁的注释**
  - ✓太少和过多均不可取
  - ✓注释要可理解、准确、无二义
- **随代码的修改而修改**

```
1  /*
2   * Copyright (c) 2010-2011, The MiCode Open Source Community (www.micode.net)
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   *        http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  */
16
17 package net.micode.notes.tool;
18
19 import android.content.Context;
37
38
39 public class BackupUtils {
40     private static final String TAG = "BackupUtils";
41     // Singleton stuff
42     private static BackupUtils sInstance;
43
44     public static synchronized BackupUtils getInstance(Context context) {
45         if (sInstance == null) {
46             sInstance = new BackupUtils(context);
47         }
48         return sInstance;
49     }
50
51     /**
52      * Following states are signs to represents backup or restore
53      * status
54      */
55     // Currently, the sdcard is not mounted
56     public static final int STATE_SD_CARD_UNMOUONTED           = 0;
```

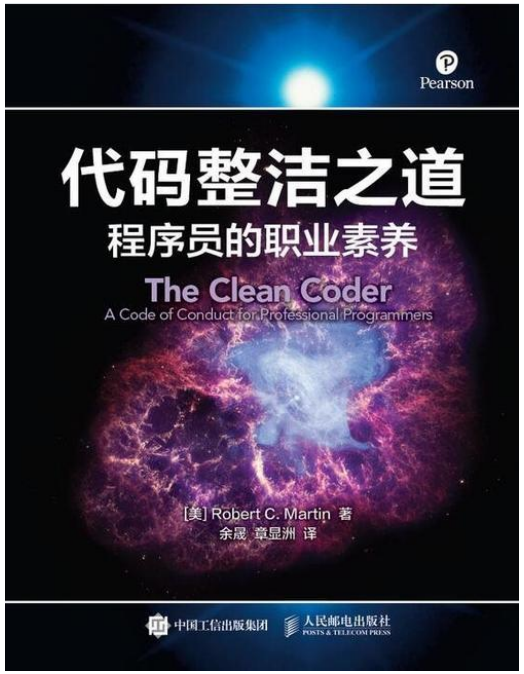
编码风格示例

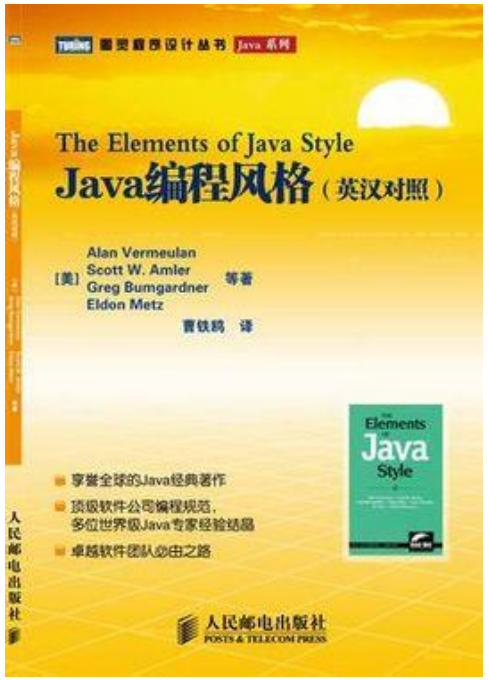
```java
/*
 * Copyright (c) 2010-2011, The MiCode Open Source Community (www.micode.net)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *        http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package net.micode.notes.data;

import android.content.Context;

public class Contact {
    private static HashMap<String, String> sContactCache;
    private static final String TAG = "Contact";

    private static final String CALLER_ID_SELECTION = "PHONE_NUMBERS_EQUAL(" + Phone.NUMBER
    + ",?) AND " + Data.MIMETYPE + "='" + Phone.CONTENT_ITEM_TYPE + "'"
    + " AND " + Data.RAW_CONTACT_ID + " IN "
            + "(SELECT raw_contact_id "
            + " FROM phone_lookup"
            + " WHERE min_match = '+')";

    public static String getContact(Context context, String phoneNumber) {
        if(sContactCache == null) {
            sContactCache = new HashMap<String, String>();
        }

        if(sContactCache.containsKey(phoneNumber)) {
            return sContactCache.get(phoneNumber);
        }

```

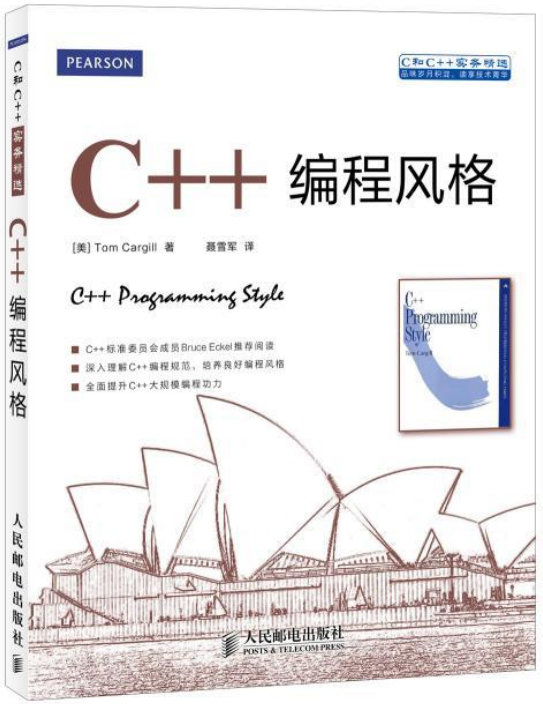
- ✓注释
- ✓命名
- ✓布局
- ✓结构

编码风格的相关书籍

- C++编程风格
- Java编程风格
- 代码整洁之道







不同程序设计语言有不同的编码风格要求

讨论代码风格

- ✓ 代码风格如何？好的一面和不好的一面？
- ✓ 你编写代码时注意风格了吗？

```
private void showLinearMenus()
{

    int[] size = DeviceUtility.getScreenSize(this);

    if (!areMenusShowing) {
        SpringAnimation.startAnimations(
                this.menusWrapper, ZoomAnimation.Direction.SHOW, size);
        //this.imageViewPlus.startAnimation(this.animRotateClockwise);
    } else {
        SpringAnimation.startAnimations(
                this.menusWrapper, ZoomAnimation.Direction.HIDE, size);
        //this.imageViewPlus.startAnimation(this.animRotateAntiClockwise);
    }
```

```
1  package com.spring.menu.animation;
2
3  import android.view.View;
4  import android.view.animation.AlphaAnimation;
5  import android.view.animation.ScaleAnimation;
6
7  public class EnlargeAnimationIn extends ZoomAnimation {
8      public EnlargeAnimationIn(int i) {
9          super(ZoomAnimation.Direction.HIDE, i, new View[0]);
10     }
11
12     @Override
13     protected void addShrinkAnimation(View[] views) {
14         // TODO Auto-generated method stub
15         addAnimation(new ScaleAnimation(0F, 1F, 0F, 1F, 1, 0.5F, 1, 0.5F));
16         addAnimation(new AlphaAnimation(0F, 1F));
17     }
```

2.3 采用程序设计方法学

- 语句设计
- 模块化设计
- 高内聚度、低耦合度原则

代码设计规范-语句设计

- 程序设计方面问题，程序内在质量的体现
- **单入口单出口**，少用goto语句
- 加强对**异常**处理
  - ✓**分析和验证输入参数正确性**，如名字不为空，年龄大于0少于150
  - ✓执行结果处理，设置必要**断言**来分析

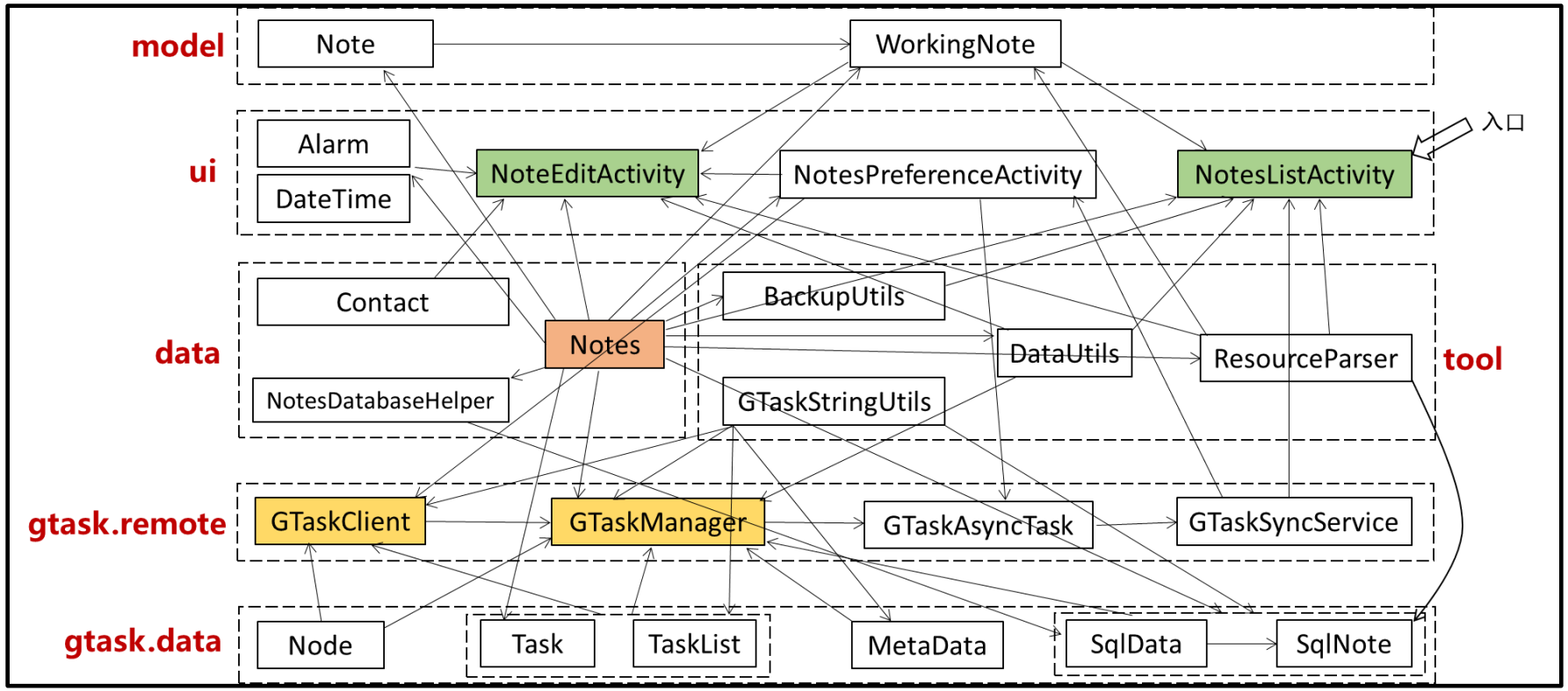
```
如 p = AllocateNewSpace( );
    Assert(p != NULL);
    if (p != NULL) {
            ……
    }
```

  - ✓处理异常语句 Try { … } catch (Exception e) { … }

将更多的时间和精力放在处理异常代码上

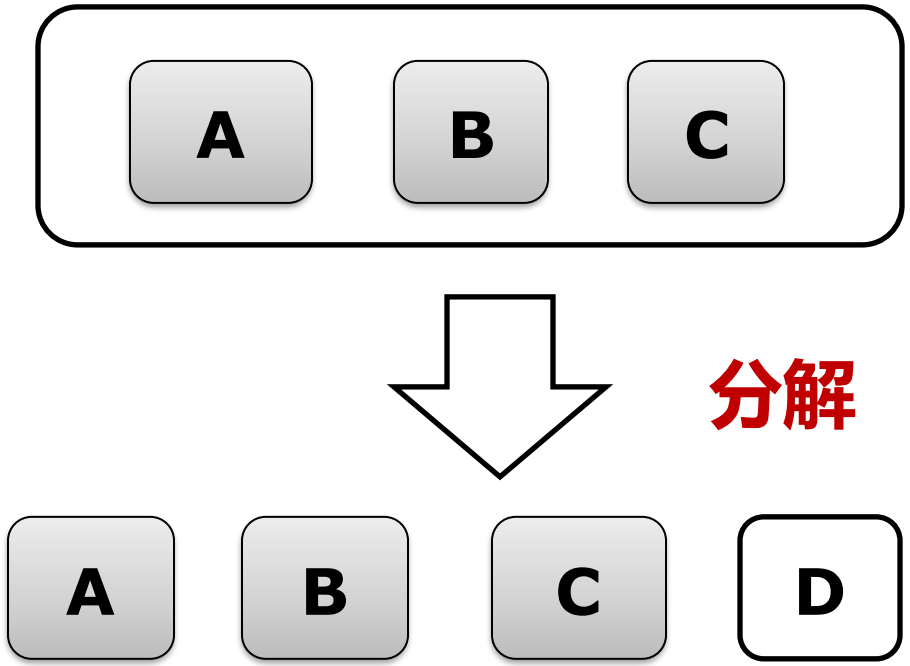
代码设计规范-模块化设计

- 模块是逻辑上相对独立、具有良定义接口的编程单位
- 模块可表现为函数、过程、方法、类、程序包等



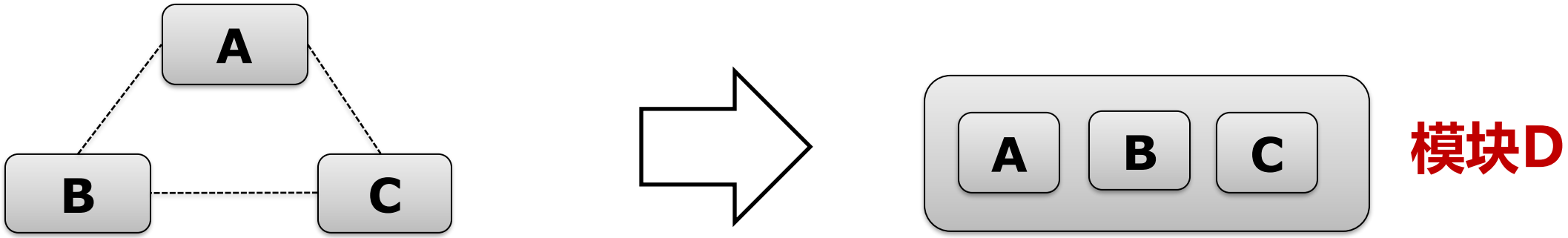
代码设计规范-高内聚度

- 模块内各要素紧密相关，仅实现单一功能
- 如果模块内多个要素关系不密切，需分解产生多个模块



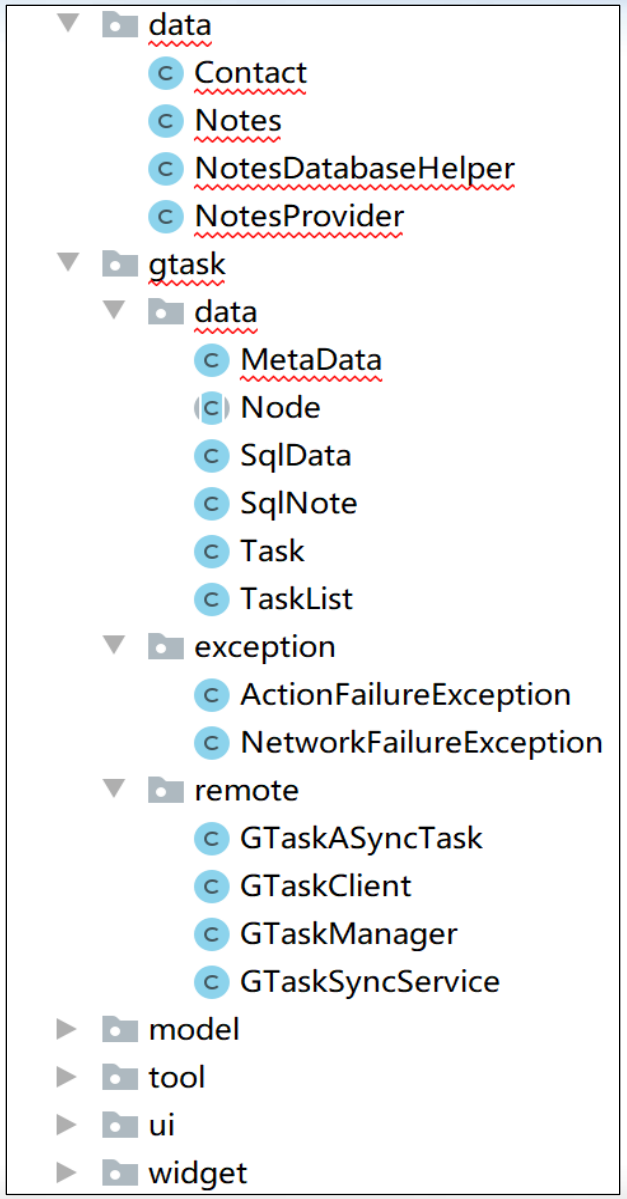
代码设计规范-低耦合度

- 模块间的关系应设计的非常松散
- 如果多个模块间的关系非常密切，可将这些模块合并为一个模块



示例：采用程序设计方法

- 语句设计
- 模块化
- 高内聚度
- 低耦合度




```java
package net.micode.notes.data;


import ...


public class NotesProvider extends ContentProvider {
    private static final UriMatcher mMatcher;

    private NotesDatabaseHelper mHelper;

    private static final String TAG = "NotesProvider";

    private static final int URI_NOTE            = 1;
    private static final int URI_NOTE_ITEM       = 2;
    private static final int URI_DATA            = 3;
    private static final int URI_DATA_ITEM       = 4;

    private static final int URI_SEARCH          = 5;
    private static final int URI_SEARCH_SUGGEST  = 6;

    static {
        mMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        mMatcher.addURI(Notes.AUTHORITY, "note", URI_NOTE);
        mMatcher.addURI(Notes.AUTHORITY, "note/#", URI_NOTE_ITEM);
```

2.4 代码重用

❑何为代码重用

- ✓在编写代码过程中，**充分利用已有和现成的代码**，并将其集成到程序之中，从而来实现程序功能

❑代码重用有何好处

- ✓由于被重用的代码经过多次反复的使用，代码质量得到充分检验，因而代码重用不仅可极大提高编程效率，而且还可有效提高程序质量

为什么代码重用可以提高代码的质量?

重用代码片段

- 寻找他人实现某些子功能的代码片段
- 可在Stack Overflow上寻找到有价值的代码片段

```
private Database(LSE item) {
    ric = item.get_ric();
    volume = item.get_volume();
}

public static final Database getInstance(LSE item) {
   if (INSTANCE == null) {
         INSTANCE = new Database(LSE item);
    }

   return INSTANCE;
 }

public void writeToDb() throws SQLException{
      //setString
}
}
```

If your application will be using Threads (Concurrency), I suggest you also to prepare your singleton for those situations , see this question

重用函数、类和软构件

- C函数库
- MFC类库(Microsoft Foundation Classes)
- Java软件开发包
- 机器人操作系统（ROS）的节点软构件

重用开源代码

- **到Github中找到粗粒度的代码开源**
  - ✓几万、甚至几十万的程序代码
- **重用开源代码来实现粗粒度的功能**
  - ✓完成诸如数据库管理、图像识别、语音分析等功能

思考和讨论

- 你在编程时是否有软件重用？重用了哪些内容？

2.5 结对编程(Pair-Programming)

- **两位程序员坐在同一工作台前一起开发软件**
  - ✓一人扮演“领航员”角色，负责具体编写工作，如写程序
  - ✓一人扮演“观察者”角色，负责观察行为及结果，如看程序，发现问题
  - ✓二者相互讨论，共同完成编程任务

个体开发的局限性

❑个体知识和技能的局限性

- ✓没有人无所不能
- ✓有专长，但很难做到面面俱到
- ✓总会有不懂、不会的

❑个体开发行为的局限性

- ✓人总是会犯错误的，错误很难避免
- ✓老虎也有打盹的时候，注意力不集中就会犯错误
- ✓很难看到或者看清自身的错误

软件开发是集体性/群体性行为

- **团队开发**
  - ✓多人参与、具有共同目的、明确任务分工
- **合作开发**
  - ✓相互支持、互相配合、共同解决问题
- **群体开发**
  - ✓利用大众的力量、借助大众的智慧和成果
- **抛弃个人主义/英雄主义**
  - ✓单枪匹马难成大事、单干/蛮干容易出事

现实世界的结对示例

- ✓ 驾驶员
- ✓ 领航员

- ✓ 驾驶员
- ✓ 作战员

如何实现结对编程

- **职责明确**
  - ✓一人写设计文档、编写程序和单元测试等等
  - ✓一人审阅文档、复审程序代码、考虑单元测试的覆盖率、是否需要重构、解决具体的技术问题等等
- **互换角色**
  - ✓不要连续超过工作1小时，提高效率
- **主动参与**
  - ✓开展讨论、解决问题、做出贡献

编程行为及其特点

- **快速完成**
  - ✓尽快得到可运行软件系统、尽早交付给用户、快速应对需求变化
- **多种技能**
  - ✓工具和环境
  - ✓程序设计语言
  - ✓业务领域知识
  - ✓编码规范和设计规范
- **质量要求**
  - ✓正确性、可读性、可维护性等

编程包袱：效率和质量问题

结对写程序

- **任务**
  - ✓编写程序代码
- **方式**
  - ✓一人写一人复审
  - ✓规范性、正确性、可读性等
- **讨论**
  - ✓改进与提高
  - ✓如增加注释、更改名称等

结对编程中的代码复审

- 结对编程是一个不断“复审”代码的过程
- 每一行代码都被二双眼睛看过，被二个脑子思考过
- 代码随着改动不断地被复审
- 每个人的一举一动（编码行为）不断地被另一个人审查，确保过程和活动置于监督之下，迫使认真工作，防止随意行为
- 促进“频繁”交流，提高个人能力和素养

结对写文档

- **任务**
  - ✓撰写计文档
- **方式**
  - ✓一人写一人复审
  - ✓规范、正确、合理性等
- **讨论**
  - ✓改进和完善
  - ✓如文档格式、语言表达、图表、错别字

结对做测试

- **任务**
  - ✓软件测试
- **方式**
  - ✓运行测试用例，收集测试结果
  - ✓一人写一人复审和帮助
  - ✓完整性、代表性、适当性等
- **讨论**
  - ✓完善和提升

结对编程带来的好处

- **提高程序质量**
  - ✓提供更好的设计质量和代码质量
  - ✓合作解决问题能力强，1+1 > 2
- **提升开发效率**
  - ✓开发人员更加信心
  - ✓有效地避免了闭门造车
  - ✓更易于发现问题和纠正问题
- **促进学习交流**
  - ✓有效的学习，做中学效果更好
  - ✓相互学习和分享经验
  - ✓更好应对人员流动，一个走了另一个人可以替换上

结对编程可以获得更高的投入/产出比

结对编程的不同阶段和技巧

- **萌芽阶段**
  - ✓刚认识，交流少，有礼貌，小心翼翼，避免冲突，有不同期望和要求
- **磨合阶段**
  - ✓开始相互接触，摸清对方，尝试交流，开展初步合作
- **规范阶段**
  - ✓相互了解对方，了解习性和特点，配合较为默契
- **创造阶段**
  - ✓能够高效开展工作，取得较好的结对编程效果
- **解体阶段**
  - ✓结对解体，各走各的路

思考和讨论

- 如何在阅读、分析和维护开源软件实践中应用结对编程的方法？有何实践经验可供分享？

内容

1. 程序及质量要求
   - ✓程序及其内部和外部质量
2. 程序质量保证方法
   - ✓编码规范、设计方法、代码重用、结对编程
3. **程序质量的分析方法**
   - ✓人工审查、自动化分析、代码测试
4. **编写程序需要解决的问题**

3.1 程序代码中潜在的质量问题

❑质量问题

- ✓**编写不合理** – 没有遵循编码规范
- ✓**设计不合理** – 没有遵循设计规范
- ✓**代码有错误** – 代码编写的不正确

❑原因

- ✓受软件开发人员**经验和水平**的限制
- ✓人可能会犯错误，**人为引入错误**

你编写的代码是否存在问题？存在哪些方面的问题？

代码质量分析的常用方法

- 人工审查方法
- 自动化分析方法
- 程序测试方法

3.2 人工审查代码

❑方法描述

- ✓阅读和理解代码
- ✓发现缺陷和问题
- ✓提出改进的建议

❑方法特点

- ✓人工审查效率低
- ✓难以发现一些深层次问题
- ✓难以全面地进行系统分析

```
1  package MarDetecter;
2
3  import java.awt.Image;
20
21 /*构建类：基础机器人；属性：当前位置，方法：在网格上由当前位置单步移动到另外一个位置*/
22 public class BasicRobot extends Agent {
23     /* 基础机器人的位置*/
24     public Coordinate posi = null;
25     private BasicRobot basicrobot = null;
26
27     public BasicRobot(){
28         basicrobot = new BasicRobot();
29     }
30 /*设置机器人位置*/
31     public void setRobotPosi(Coordinate p){
32         posi = p;
33     }
34
35 /*获取机器人位置*/
36     public Coordinate getRobotPosi(BasicRobot thisrobot){
37         return(thisrobot.posi);
38     }
39     /* 基础机器人由当前位置单步移动到另外一个位置方法实现*/
40     public void moveStep(Coordinate start, Coordinate next) {
41         ShortestPath path = new ShortestPath();
42         path.setStart(start.x, start.y);
43         path.setTarget(next.x, next.y);
44
45     }
46
47     protected void setup() {
48         System.out.println("BasciRobot'name is " + getLocalName());
49         addBehaviour(new RandomWalkBev(this));
50
51     }
52
53 }
54
```

人工审查些什么？

- 代码是否符合编程规范
- 代码中是否存在缺陷
  - ✓逻辑错误，“+”写成“-”
  - ✓算法错误，不够优化、边界条件没有处理好
  - ✓潜在错误，当前修改导致以前修复的错误重现
- 从质量的角度哪些代码需要改进

- 读别人编写的高质量代码能让你受益匪浅
- 读低质量的代码能让你非常痛苦

人工审查-谁负责审查

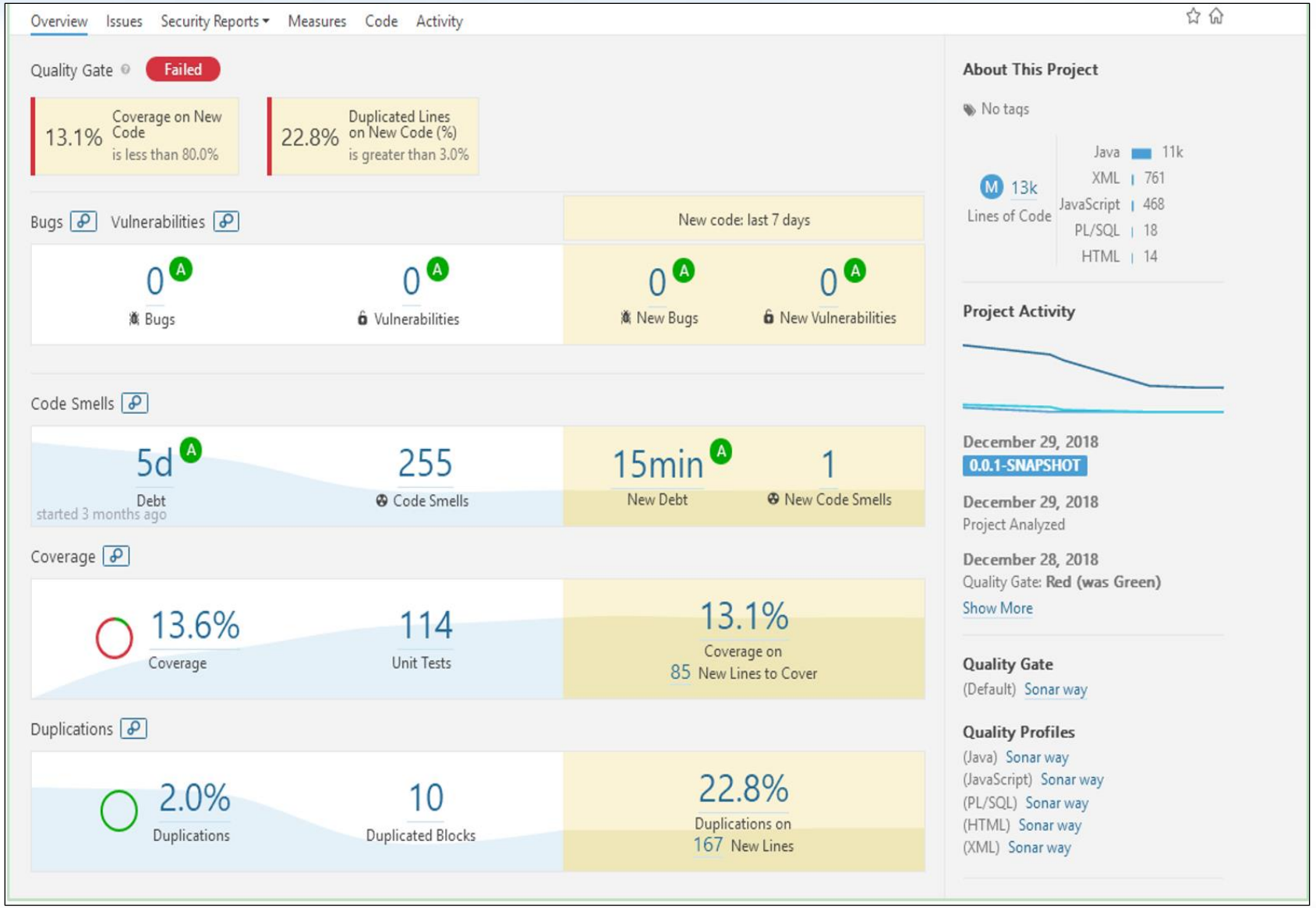
- 自我复审，效果不一定好
- 同伴复审，常用方法 → 结对编程
- 团队复审，团队成员参加

3.3 自动化工具审查-代码静态分析

- **由计算机软件来自动完成代码审查**
  - ✓**无需运行被测代码**，仅通过分析或检查程序的语法、结构、过程、接口等来检查程序
  - ✓30% - 70% 的代码缺陷可通过静态分析发现
- **分析什么**
  - ✓**找出代码隐藏的错误和缺陷**，如参数不匹配，有歧义的嵌套语句，错误的递归，非法计算，可能出现的空指针引用等等
  - ✓**程序遵循编码规范的程度**
- **特点**
  - ✓快速定位、有效发现隐藏的错误和缺陷

自动化代码分析工具

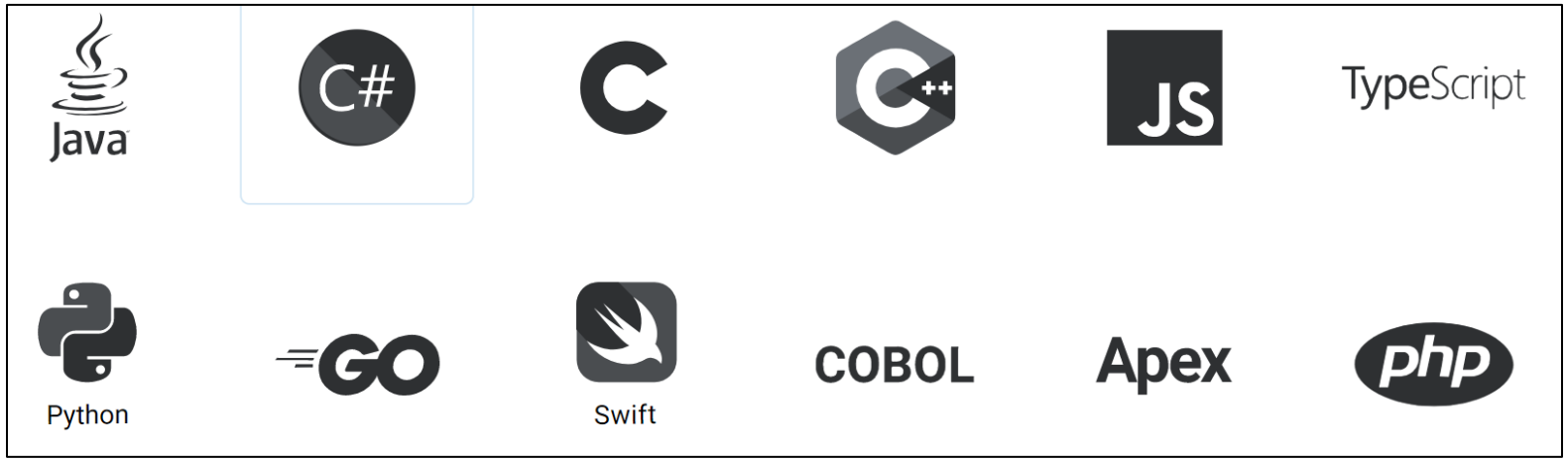
- SonarQube
- CheckStyle
- FindBugs，
- PMD
- Jtest
- ……



SonarQube概述

- 基于Web、用于管理程序代码质量的**代码分析工具**
  - ✓www.sonarqube.org
  - ✓能以插件的形式集成到众多的软件开发环境（如Eclipse）

- 支持二十多种程序设计语言代码的质量分析



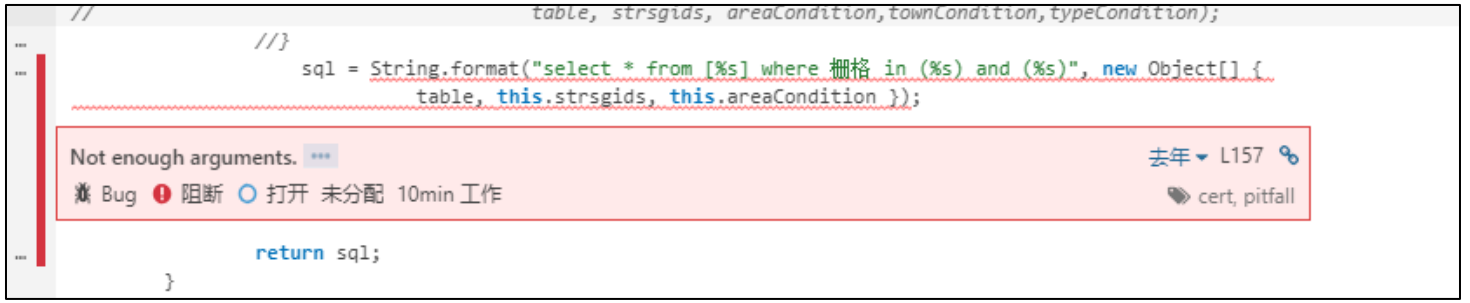
sonarqube

SonarQube可分析的质量问题

- 是否违反**编码规则**
- 是否存在静态**常规缺陷**
- 模块、方法、类的**复杂度**是否过高
- 是否存在**重复的代码**
- 代码的**注释**是否恰当和充分
- 统计和分析代码的**单元测试覆盖率**
- 判断软件体系结构设计是否**合理**

SonarQube分析发现的问题类别

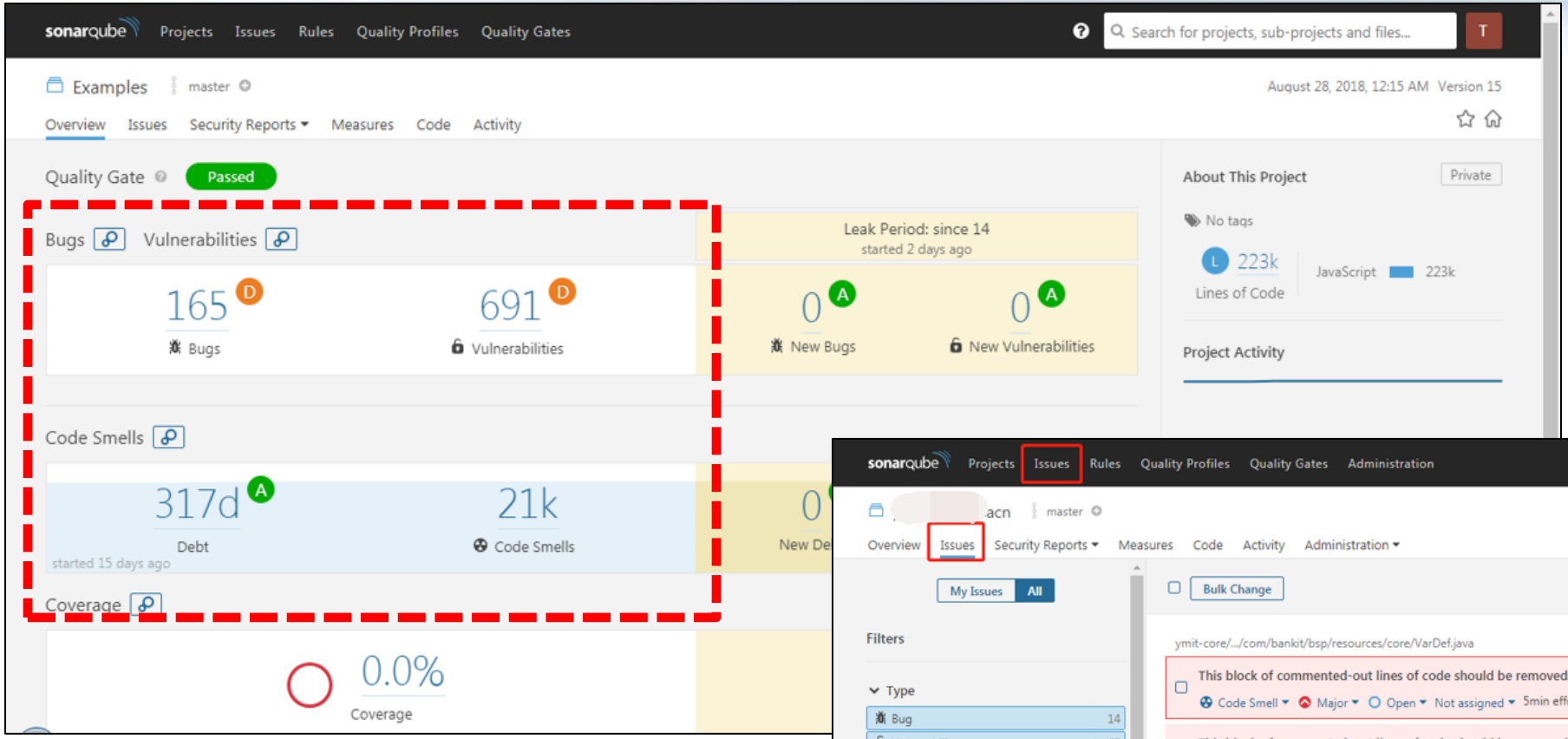
- **Bug（错误）**
  - ✓中等影响
  - ✓如参数不够

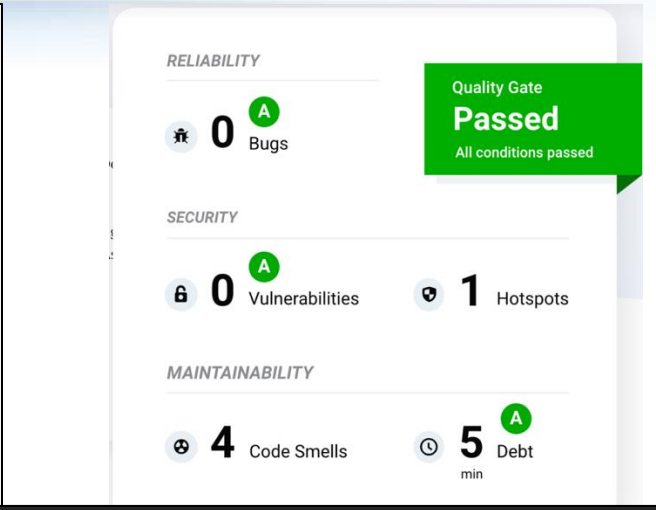


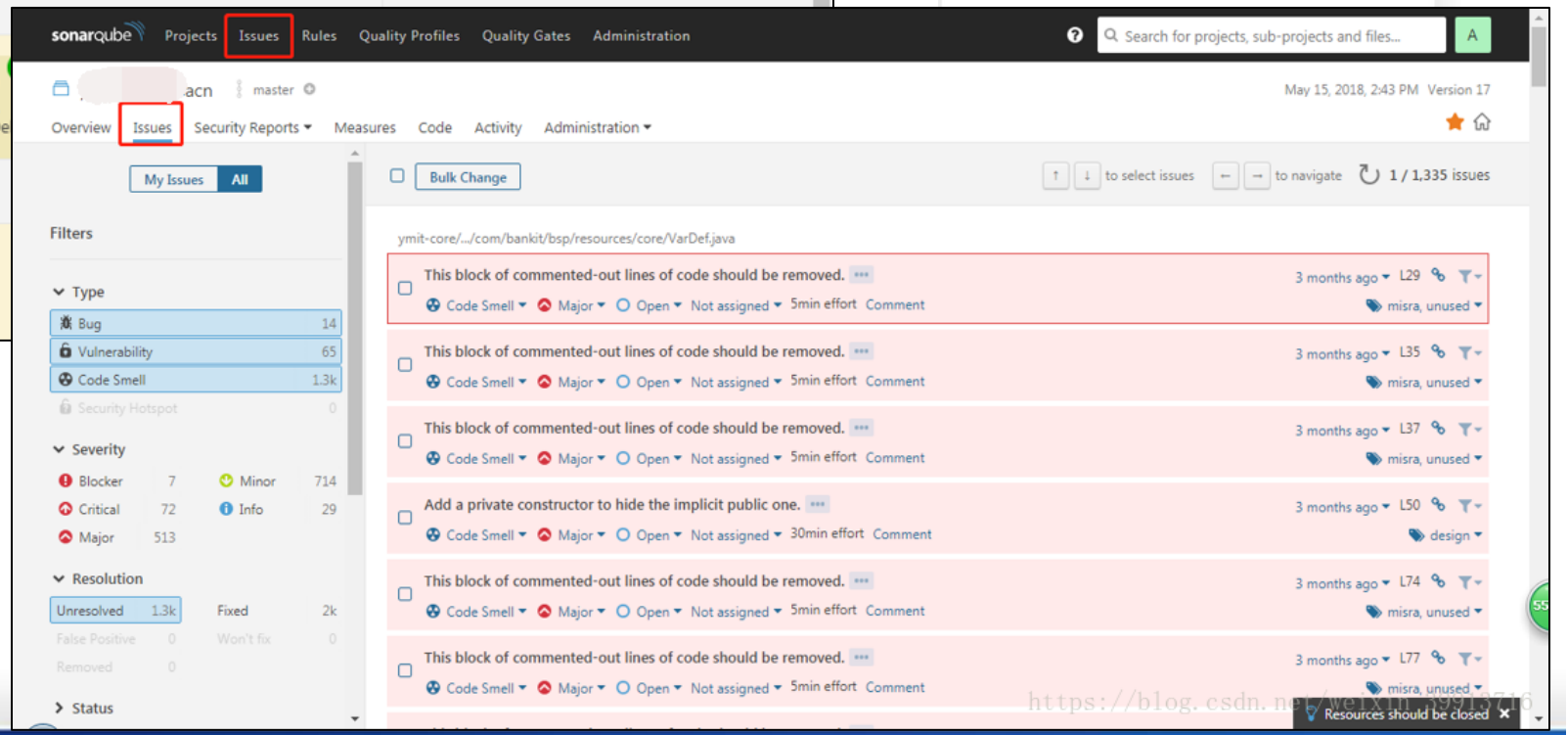
- **Vulnerability（脆弱点）**
  - ✓影响大
  - ✓如漏洞

- **Code Smell（代码异味）**
  - ✓影响小
  - ✓如风格

SonarQube分析报告





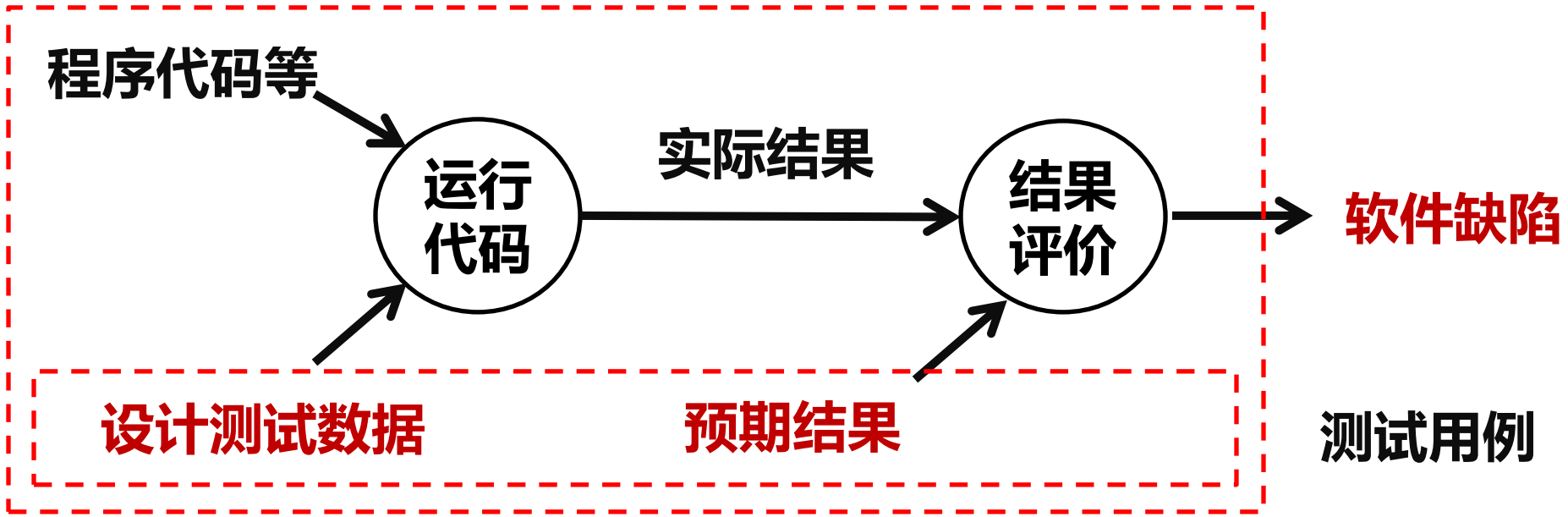


代码审查后要做的工作

- 理解发现和指出的问题
- 修改和更正有问题代码
- 对于无法很快更正的错误，要把错误的信息记录下来，以便适当的时候能够更正

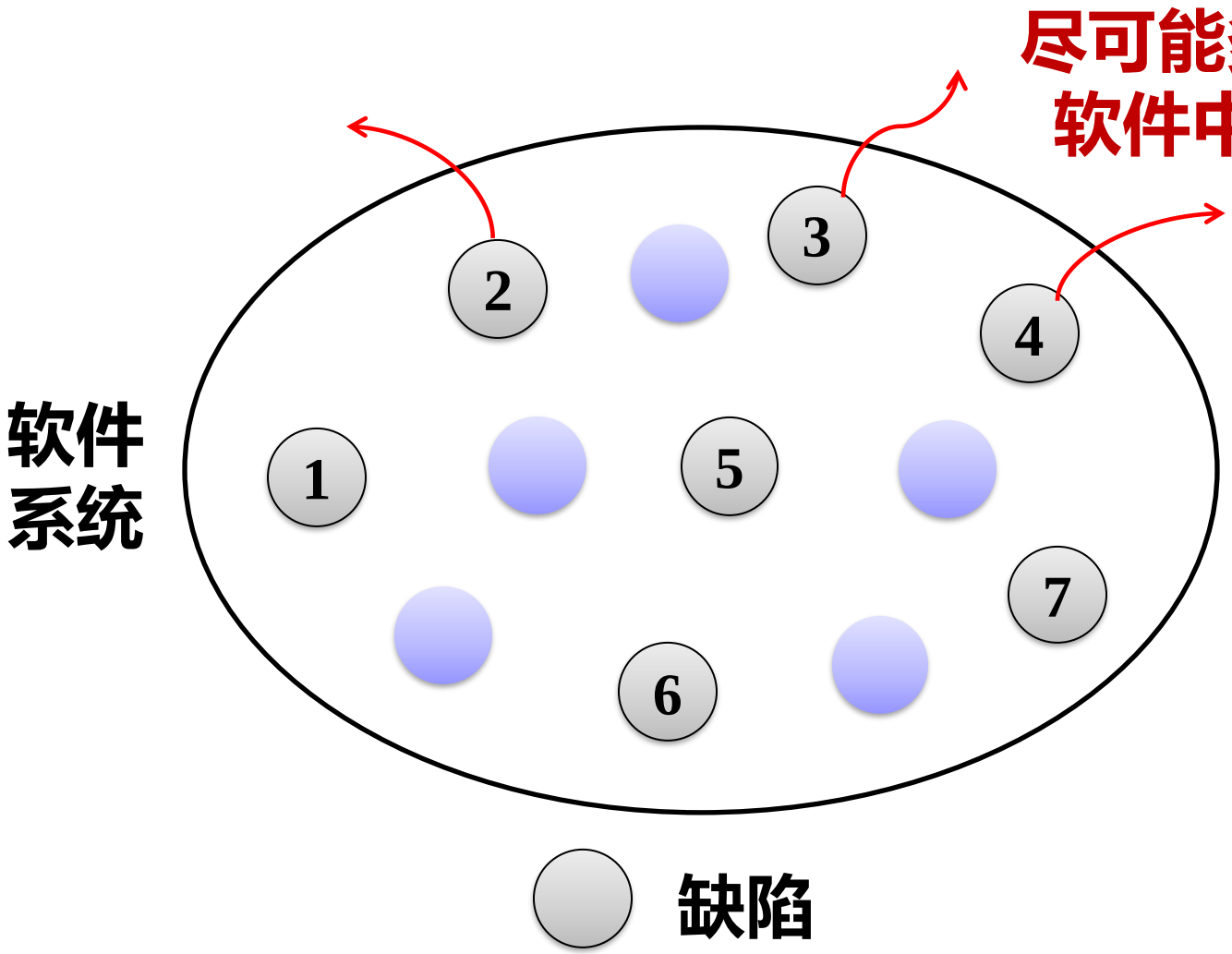
3.4 软件测试技术

- 程序本质上是对数据的处理
- 设计数据(测试用例) ➔ 运行测试用例(程序来处理数据) ➔ 判断运行结果(是否符合预期结果)



为软件测试而设计的数据称为测试用例(Test Case)

软件测试的目的和任务



软件缺陷可能隐藏的比较深，难以发现

软件测试示例

- **一个加法器程序**
  - ✓功能：给定二个数字，将其相加，然后输出
- **设计测试数据**
  - ✓<1,2,3><-1,1,0><0,0,0>
- **运行测试程序**
  - ✓输入数据，查看运行结果，判断是否与预期结果一致
  - ✓如果不一致就意味着有错误

思考和讨论

- 在你的编程实践中，你是采用何种方式来检查代码质量、发现代码问题的?

内容

1. 程序及质量要求
   - ✓程序及其内部和外部质量
2. 程序质量保证方法
   - ✓编码规范、设计方法、代码重用、结对编程
3. 程序质量的分析方法
   - ✓人工审查、自动化分析、代码测试
4. **编写程序需要解决的问题**

程序编写面临的挑战！

- **程序功能从何而来?**
  - ✓谁以及如何来确定软件功能?
- **软件规模很大怎么办?**
  - ✓500 LOC vs 1M LOC
- **软件功能变化如何改写代码?**
  - ✓在哪里改、如何改?
- **如何保证程序质量?**
  - ✓能否想到代码的质量问题，如何保证

光靠脑子思考能解决编写程序的问题吗?

思考和讨论

- 如何明确功能、划分模块?
- 如何编写代码、确保质量?
- 对于规模较大应用能行吗?
- 面临什么样的困难和问题?

小结

- **程序的多种质量要求**
  - ✓外在和内在、语法和语义
- **确保代码质量的方法**
  - ✓编码规范、设计方法、代码重用、结对编程
- **分析、发现和审查代码**
  - ✓人工审查、静态分析、程序测试
- **学会编写高质量的程序代码**
  - ✓如何编写代码？如何确保质量？

课后的实践任务

- **开展基于结对的代码分析和维护实践**
- **阅读和掌握编码风格**
  - ✓Java、C/C++编程风格，要求学以致用
- **熟练掌握SonarQube工具**
  - ✓安装和使用，分析开源软件的代码质量
- **开展课程实践**
  - ✓阅读、理解和分析开源软件的代码质量
  - ✓提交代码的质量分析报告
- **思考：如何编写高质量的程序代码**

综合实践一

- **任务：选取或指定待阅读、分析和维护的开源软件。**
- **方法**
  - ✓访问Github、码云Gitee、SourceForge等开源软件托管平台，从中检索到符合上述要求的开源软件，下载或克隆开源软件代码，阅读开源软件的相关文档来安装、部署和运行开源软件。以二人为一组、采用结对方式来开展本综合实践
- **要求**
  - ✓所选取或指定的开源软件要求功能易于理解、代码质量高、规模适中（5000-20000行代码量），也可以直接指定“MiNote”便签管理开源软件作为阅读、分析和维护的对象。
- **结果：获得开源软件源代码，并可运行和操作该开源软件**

综合实践二

- **任务：查看和分析开源软件**
- **方法**
  - ✓访问Github、SourceForge、Gitee等开源软件托管平台或Apache、Eclipse等开源软件基金会平台，从中检索自己感兴趣的开源软件，阅读相关的软件文档，下载安装开源软件
- **要求**
  - ✓结合自己的兴趣，查看有那些开源软件，分析这些软件的功能和定位、存在的缺陷和不足
- **结果：掌握开源软件托管平台的使用方法，大致了解感兴趣的开源软件情况**

思考和讨论

你是否意识到程序质量的重要性？你认为高质量的程序应该是什么样的？

问题和讨论