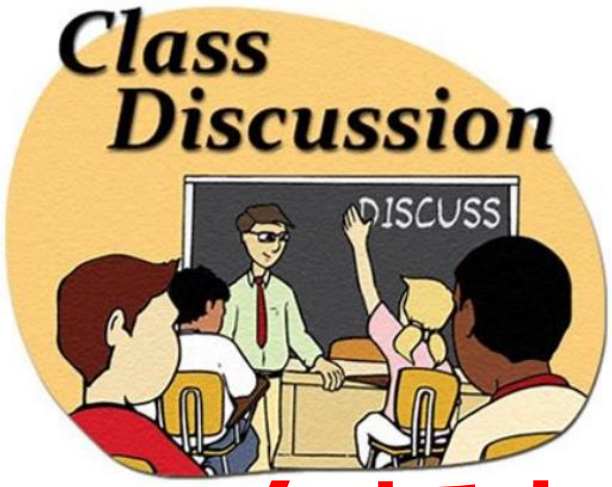
CS2001
软件工程

3. 个人开发技能
—代码质量

课堂讨论：代码质量



课堂讨论：回想一下，自己写过的代码里面有哪些质量问题

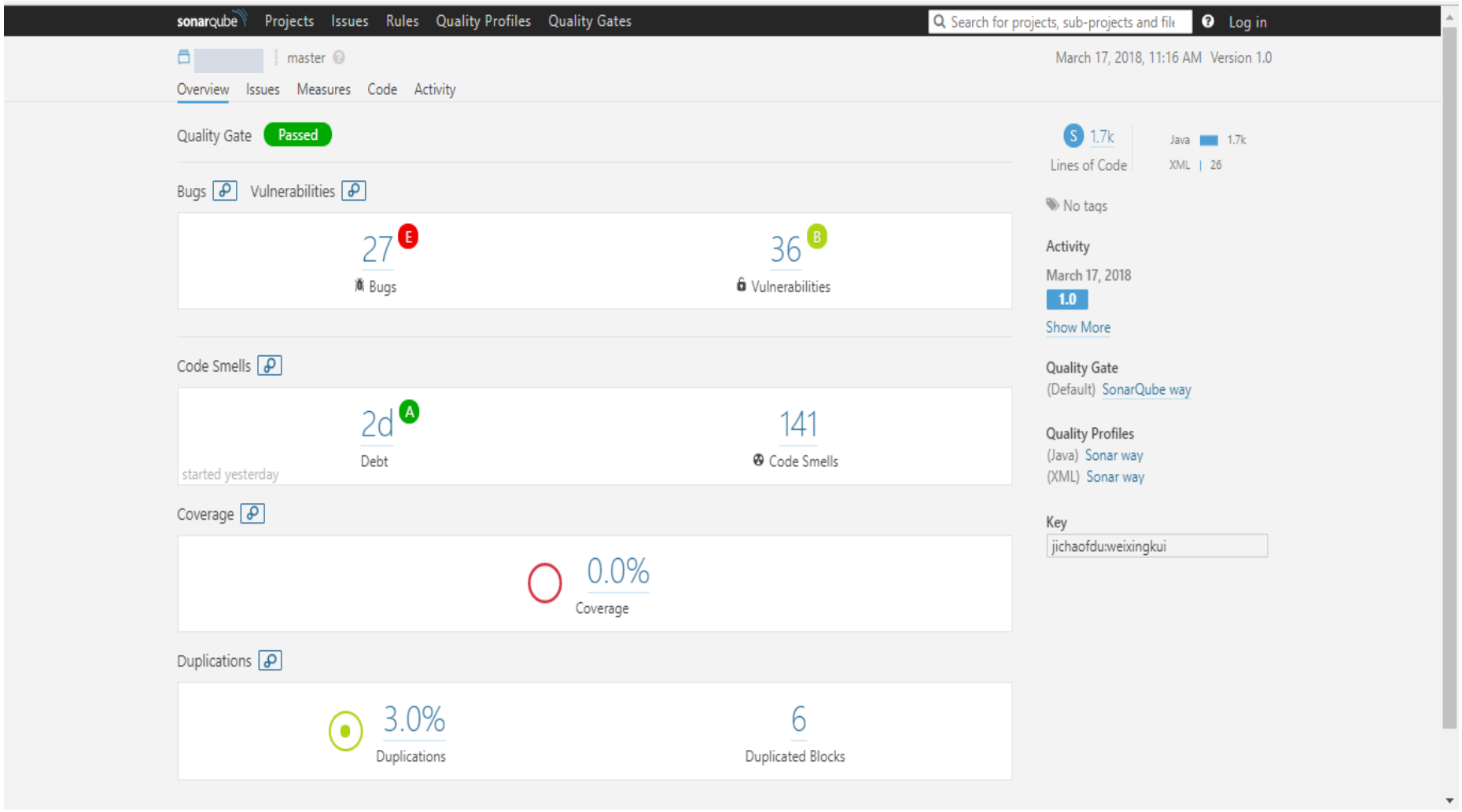
包括内部和外部质量两方面

影响代码的理解和修改

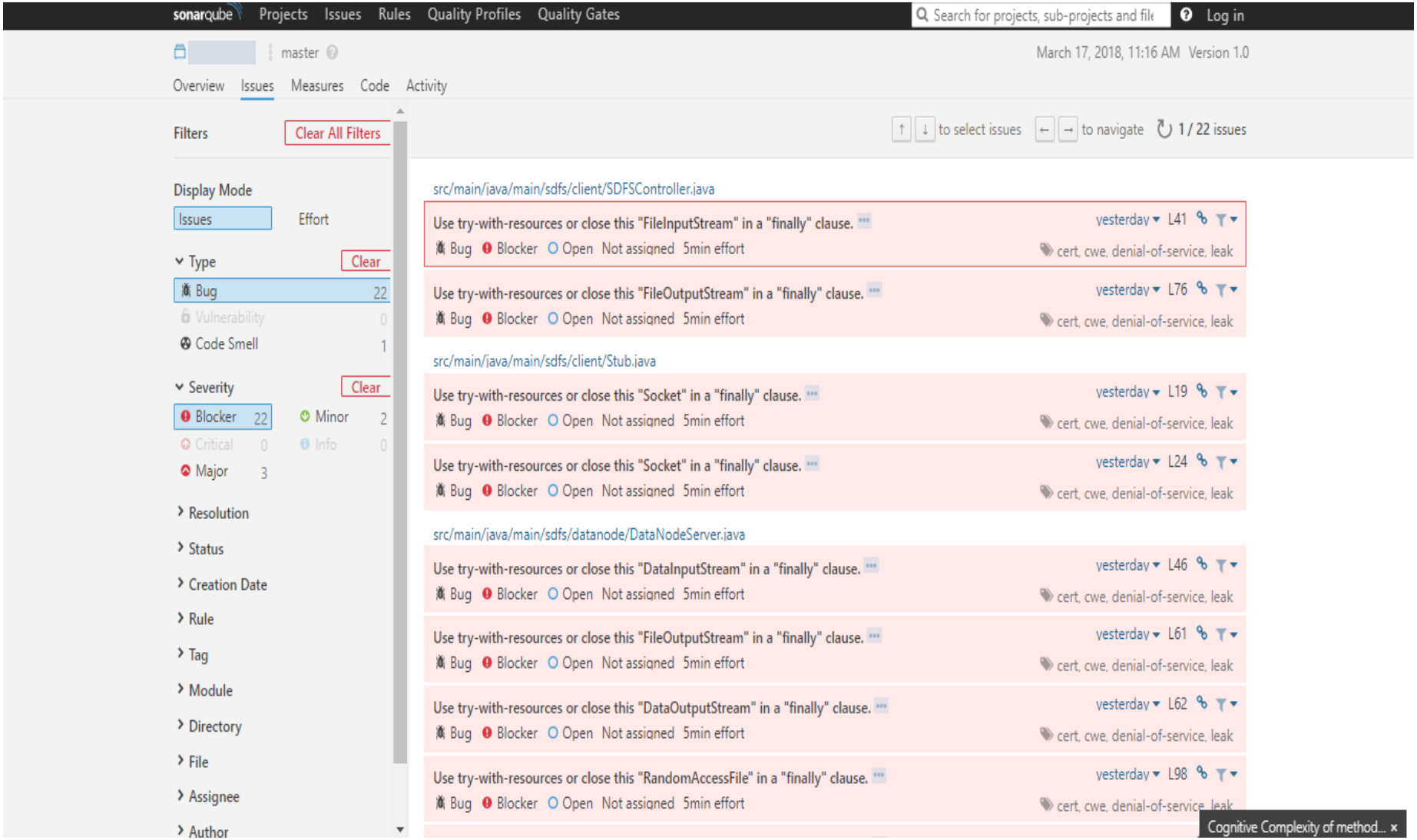
增加了程序出错的概率

影响程序运行性能和可靠性

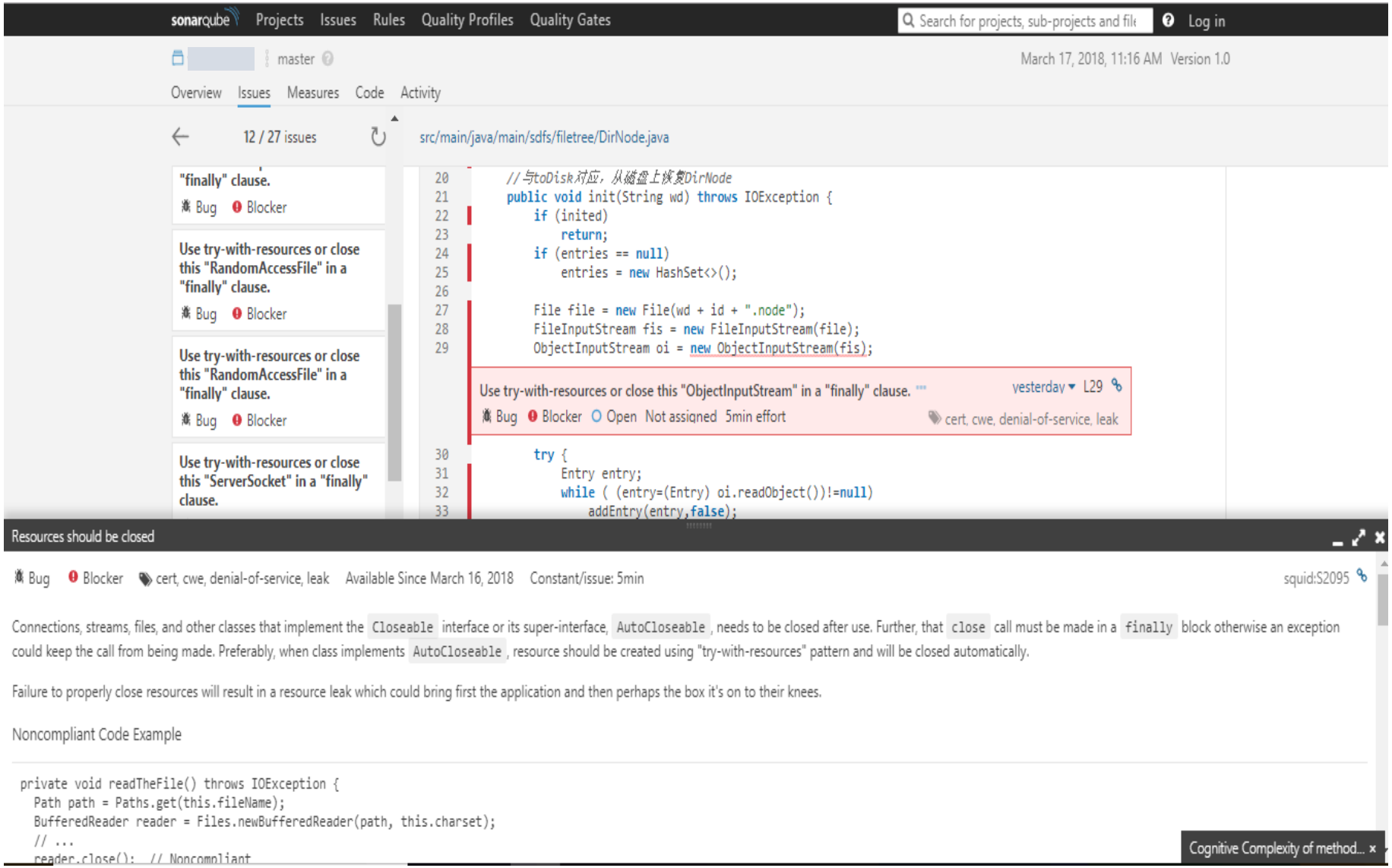
代码质量分析工具SonarCube-检测概览



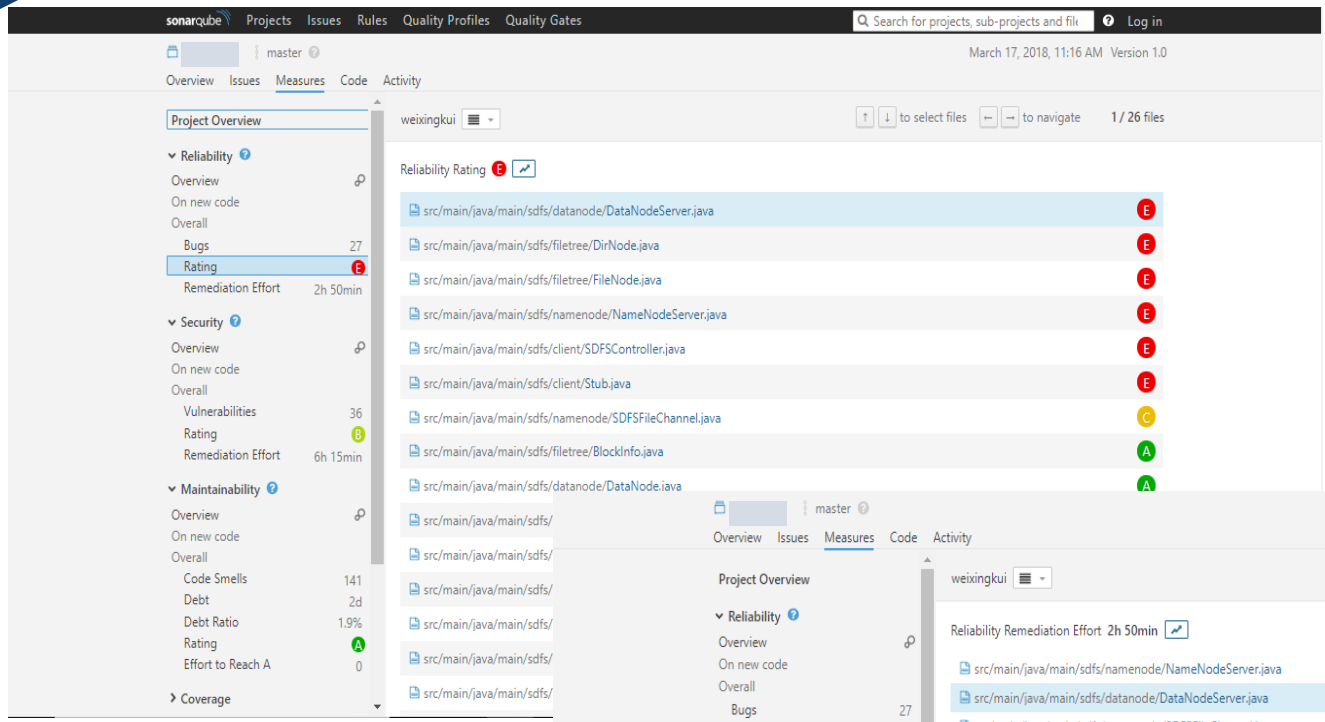
代码质量分析工具SonarCube-问题报告

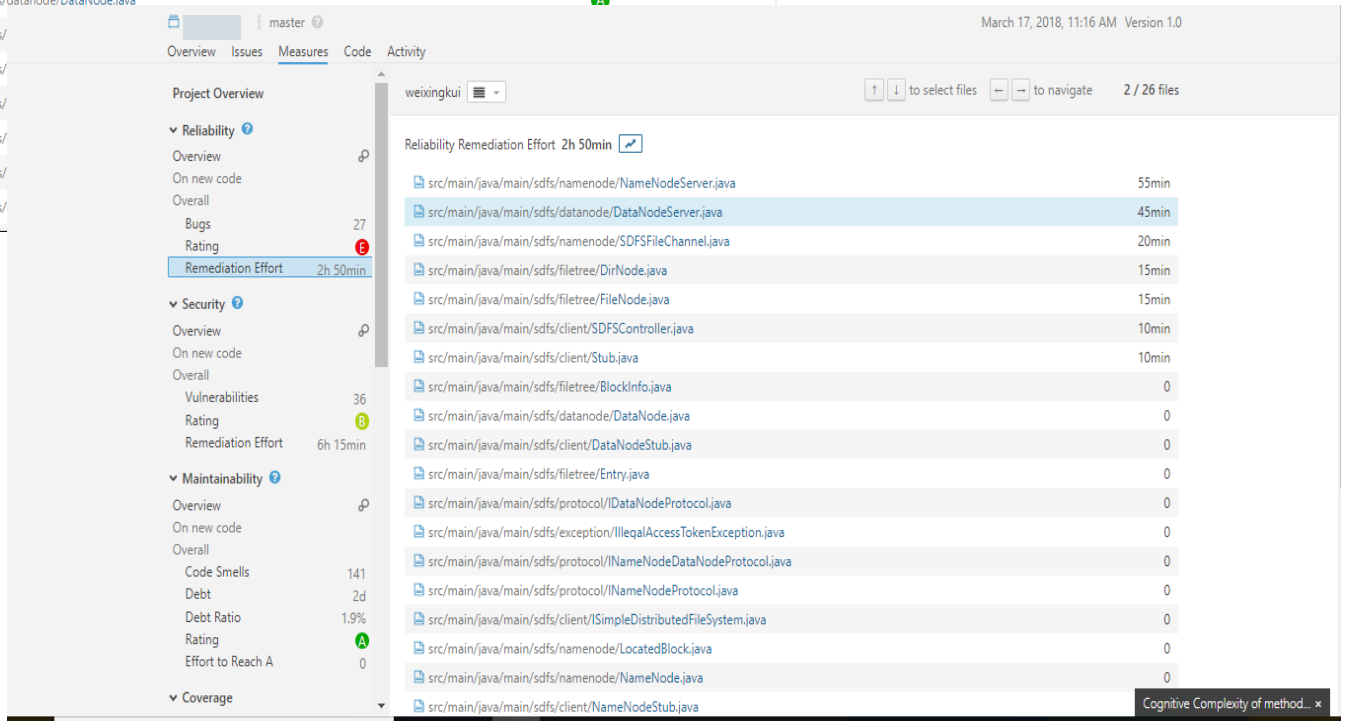


代码质量分析工具SonarCube-详细解释



代码质量分析工具SonarCube-问题评级





学生代码质量抽样分析

姓名	LOC	Bugs	Vulnerabilities	Code Smells	Duplications
S1	638	14	4	50	2.6%
S2	1000	1	0	82	35.8%
S3	2200	7	44	201	8.5%
S4	701	4	4	51	15.6%
S5	370	2	2	36	0.0%
S6	634	6	1	26	0.0%
S7	1700	27	36	141	3.0%
S8	1100	14	0	163	0.0%
S9	1300	15	31	121	1.4%
S10	932	12	0	251	13.2%
S11	657	0	0	42	9.6%
S12	1800	7	18	75	20.3%
S13	1000	17	16	157	13.4%
S14	277	2	0	35	0.0%
S15	488	2	8	37	0.0%

Bug示例-1

Stream对象使用后没有关闭

```
3           }
4           BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(nf));
```

Use try-with-resources or close this "BufferedOutputStream" in a "finally" clause. ··· 17 minutes ago ▾ L14

Bug ▾ Blocker ▾ Open ▾ Not assigned ▾ 5min effort Comment cert, cwe, denial-of-service, leak ▾

```
5           int subnum=countSubfile(f);
6
```

循环没有终止条件

```
        while (true) {
            player1animals = copyArray(historyPlayer1[currentStep - 1]);
            player2animals = copyArray(historyPlayer2[currentStep - 1]);
            moveAndJudge();
            ErrorInstruction();
            if (judgeInputFormat && judgeInputRange) {
                if (countStep) {
                    playerAnimals = player1animals;
                    playButWhichKind();
                    player1animals = copyArray(playerAnimals);
                } else {
                    playerAnimals = player2animals;
                    playButWhichKind();
                    player2animals = copyArray(playerAnimals);
                }
                if (!toHelp) {
                    setHistory();
                }
            }
            eat();
            System.out.println("玩家输入为:  \"" + playerInput + "\"");
            printHistory();
            countStep = !countStep;
            judgeWinNoAnimalLeft();
            judgeWinAnimalInOppositeHole();
            for (int i = 0; i < 7; i++) {
                for (int j = 0; j < 9; j++) {
                    if (animalsMap[i][j] != 0) {
                        if (j < 8) {
                            System.out.print(animalsMap[i][j]);
                        } else {
                            System.out.print(animalsMap[i][j]);
                            System.out.println();
                        }
                    }
                }
            }
        }
```

Add an end condition to this loop. ··· 2 days ago ▾ L83

Bug Blocker Open Not assigned 15min effort cert

Bug示例-2

没有对有返回值的方法进行返回值检测，可能导致副作用

```
        bis.read(singleinteger);
```

Check the return value of the "read" call to see how many bytes were read. 21 minutes ago L12
Bug Minor Open Not assigned 15min effort Comment cert

```
        int subnum = bytesToInt(singleinteger, 0);
        if (subnum == 0) {
```

数字类型之间强制转换可能导致精度丢失等问题

```
        bufferedInputStream.skip(1 + treeLength);
```

Cast one of the operands of this addition operation to a "long". 22 minutes ago L118
Bug Minor Open Not assigned 5min effort cert, cwe, misra, overflow, sans-...

方法复写时没有复写其他可能一起使用的方法

```
    }

    @Override
    public boolean equals(Object obj) {
```

This class overrides "equals()" and should therefore also override "hashCode()". 1 hour ago L99
Bug Minor Open Not assigned 15min effort Comment cert, cwe

```
        return (obj instanceof Animal) && ((Animal) obj).power == this.power && ((Animal) obj).side == this.si
    }
}
```

Bug示例-3

永远不会被执行的else分支

```
            if (animal == null) {
            } else if ( 1 animal.getSide() == Animal.Side.LEFT) {
                sumPowerL += animal.getPower();
            } else if (animal.getSide() == Animal.Side.LEFT) {
```

This branch can not be reached because the condition duplicates a previous condition in the same sequence of "if/else if" statements ···

1 hour ago ▾ L432

Bug ▾ Major ▾ Open ▾ Not assigned ▾ 10min effort Comment

cert, pitfall, unused ▾

```
                sumPowerL += animal.getPower();
            }
        } catch (Exception ignored) {
```

For循环里的终止条件永远无法被触发，容易产生误解

```
    }

    private void buildHeap() {
        for (int i = currentSize / 2; i > 0; i++)
```

"i" is incremented and will never reach "stop condition". ···

1 minute ago ▾ L39

Bug ▾ Major ▾ Open ▾ Not assigned ▾ 5min effort Comment

cert ▾

```
            percolateDown(i);
```

Correct this loop's end condition. ···

1 minute ago ▾ L40

Bug ▾ Blocker ▾ Open ▾ Not assigned ▾ 15min effort Comment

cert ▾

```
    }
```

Bug示例-4

没有对可能为null的变量进行检验就直接使用

```
//            System.out.println("i 值:  " + i);
            Node y=x.children.get(i);
          1 Node z=null;
            if (i<=x.number) {
                if (i<x.number)
                    z = x.children.get(i + 1);
                if (i<x.number&&kv.key.equals(x.keys.get(i).key)) {
                    KV kv1;
                    if (y.number > t - 1) {
                        kv1 = BTsearchPredecessor(y);
                        BTdeleteNonone(y, kv1);
                        x.keys.set(i, kv1);
                    } else if ( 2 z.number > t - 1) {
```

A "NullPointerException" could be thrown; "z" is nullable here. ··· 1 minute ago ▾ L359

Bug ▾ Major ▾ Open ▾ Not assigned ▾ 10min effort Comment cert, cwe ▾

```
                        kv1 = BTsearchSuccessor(z);
                        BTdeleteNonone(z, kv1);
                        x.keys.set(i, kv1);
```

未接收并利用方法返回值

```
    public String[] parseUri(String fileUri) throws URISyntaxException {
        fileUri.trim();
```

The return value of "trim" must be used. ··· 24 minutes ago ▾ L343

Bug Major Open Not assigned 10min effort cert, misra

```
        int len = fileUri.length();
```

Vulnerability示例-1

没有处理方法执行失败情况

```
        nf.createNewFile();
```

Do something with the "boolean" value returned by "createNewFile". 19 minutes ago L11

Vulnerability Minor Open Not assigned 15min effort Comment cert, cwe, error-handling, misra

没有使用logger来记录运行异常

```
            butteredReader.close();
        } catch (IOException e) {
            e.printStackTrace();
```

Use a logger to log this exception. 1 hour ago L292

Vulnerability Minor Open Not assigned 10min effort Comment error-handling

```
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
```

Use a logger to log this exception. 1 hour ago L295

Vulnerability Minor Open Not assigned 10min effort Comment error-handling

Vulnerability示例-2

类变量可能被恶意修改

Make bNumber a static final constant or non-public and provide accessors if needed. 1 hour ago L48

Vulnerability Minor Open Not assigned 10min effort Comment cwe

```
public static  int wNumber = 0;
```

公共变量放在了公共接口中且无访问控制

```
public interface something {
    Stack stacka = new Stack();//数组标签栈
```

Move "stacka" to a class and lower its visibility

Vulnerability Minor Open Not assigned 15min effort

```
    Stack stackb = new Stack();//数组标签栈
```

Move "stackb" to a class and lower its visibility

Vulnerability Minor Open Not assigned 15min effort

```
    int VOID = 0;//假空
```

Code Smell示例-1

条件语句中没有对某些可能发生的情况进行处理

```
if (nm.indexOf('.') >= 0||nm.indexOf("empty_file")>0) {
```

0 is a valid index, but is ignored by this check. ··· 24 minutes ago ▾ L49

Code Smell ▾ Critical ▾ Open ▾ Not assigned ▾ 2min effort Comment suspicious ▾

```
if(nm.indexOf("empty_file")>0){
```

0 is a valid index, but is ignored by this check. ··· 24 minutes ago ▾ L50

Code Smell ▾ Critical ▾ Open ▾ Not assigned ▾ 2min effort Comment suspicious ▾

```
df.createNewFile();}
else
```

过于复杂的分支判断语句

```
public static String [][] createAChessboard()//定义一个生成棋盘的函数
```

Refactor this method to reduce its Cognitive Complexity from 24 to the 15 allowed. ··· 54 minutes ago ▾ L6

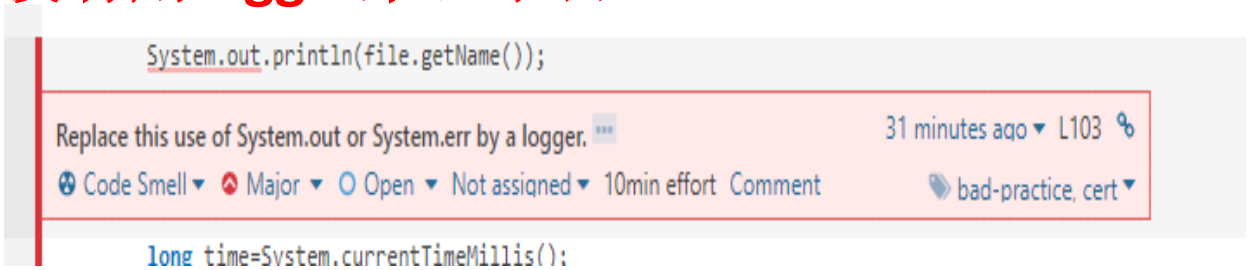
Code Smell ▾ Critical ▾ Open ▾ Not assigned ▾ 14min effort Comment brain-overload ▾

```
{
    String [][] chessboard = new String[17][17];//创建17*17的数组，防止有坐标越界的报错
 1 for (int row = 0; row < chessboard.length; row++)//利用循环来建立棋盘数组
    {
     2 for (int column = 0; column < chessboard[row].length; column++)
       {
         3 if (row == 1  4 && column == 1)
           {
               chessboard[row][column] = "┌";
           }
           else  5 if (row == chessboard.length - 2  6 && column == 1)
           {
               chessboard[row][column] = "└";
           }
           else  7 if (row == 1  8 && column == chessboard[row].length - 2)
           {
               chessboard[row][column] = "┐";
           }
           else  9 if (row == chessboard.length - 2  10 && column == chessboard[row].length - 2)
           {
               chessboard[row][column] = "┘";
           }
           else  11 if (row == 1  12 && column > 1 && column < chessboard[row].length - 2)
           {
               chessboard[row][column] = "┬";
           }
           else  13 if (row == chessboard.length - 2  14 && column > 1 && column < chessboard[row].leng
           {
               chessboard[row][column] = "┴";
           }else  15 if (column == 1  16 && row > 1 && row < chessboard.length - 2)
           {
```

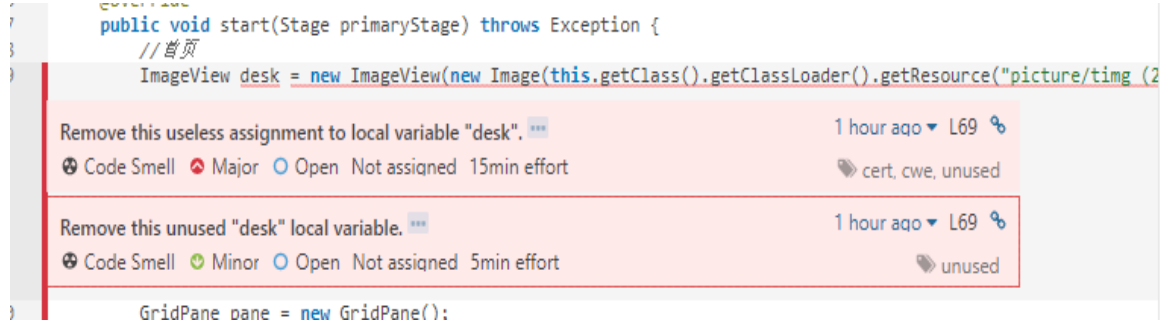
Cognitive Co

Code Smell示例-2

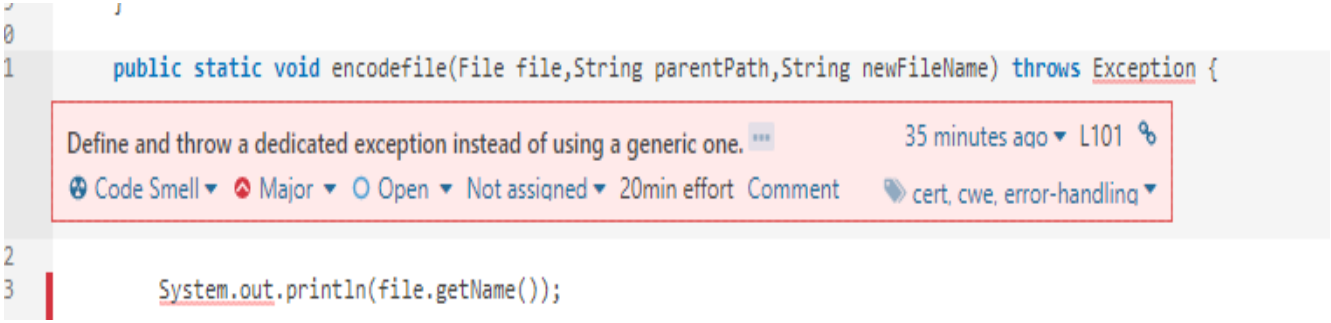
没有用logger来记录日志



未使用的变量和变量赋值



抛出一个笼统的、类型不明的异常



Code Smell示例-3

类、方法、变量命名不规范

```
public class fileStructure {
```

Rename this class name to match the regular expression '^[A-Z][a-zA-Z0-9]*$'. ... 38 minutes ago ▾ L3

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 5min effort Comment convention ▾

```
    FILE root;
    FILE[] subfile;
```

空语句

```
            }
            catch (Exception e){

            };
```

Remove this empty statement. ... 39 minutes ago ▾ L77

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 2min effort Comment cert, misra, unused ▾

不必要的import语句

```
import java.io.*;
import java.lang.String;
```

Remove this unnecessary import: java.lang classes are always implicitly imported. ... 41 minutes ago ▾ L2

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 2min effort Comment unused ▾

```
import java.lang.StringBuilder;
```

Remove this unnecessary import: java.lang classes are always implicitly imported. ... 41 minutes ago ▾ L3

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 2min effort Comment unused ▾

```
public class Encode {
    static int number=-1;
```

Code Smell示例-4

硬编码的路径分隔符

```
File f = new File(parentPath + "/" + newFileName);
```

Remove this hard-coded path-delimiter. 43 minutes ago ▾ L110

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 20min effort Comment android, cert ▾

重复出现的字符串未使用常量定义

```
menuBar.setPrefWidth(1000);
menuBar.setStyle( 1 "-fx-background-color:#66666666");
```

Define a constant instead of duplicating this literal "-fx-background-color:#66666666" 4 times. 1 hour ago ▾ L93

Code Smell Critical Open Not assigned 10min effort design

```
VBox vBox = new VBox(25, menuBar, new Time().TimePane());
```

不适当的代码排版导致令人误解的循环语句

```
}
for(int i=0;i<3;i++)
    mediaPlayer[i].stop();
    media1=0;
```

This line will not be executed in a loop; only the first line of this 2-line block will be. The rest will execute only once. 1 hour ago ▾ L80

Code Smell ▾ Major ▾ Open ▾ Not assigned ▾ 5min effort Comment cert, cwe ▾

Code Smell示例-5

早期同步化的类在性能上可能存在不足

```
37      MenuItem choice2 = new MenuItem("悔棋(b)");
38      static Stack stack= new Stack();//用于每下一个棋往栈中放入它的横纵坐标
```

Replace the synchronized class "Stack" by an unsynchronized one such as "Deque". ··· 1 hour ago ▾ L38

Code Smell ▾ Major ▾ Open ▾ Not assigned ▾ 20min effort Comment performance ▾

```
39      static Stack stack1 = new Stack();
```

声明了抛出运行时异常

```
    @Override
    public SDFSFileChannel openReadwrite(String fileUri) throws IndexOutOfBoundsException, IllegalStateExcept
```

Remove the declaration of thrown exception 'java.lang.IndexOutOfBoundsException' which is a runtime exception. ··· 6 minutes ago ▾ L35

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 5min effort Comment clumsy, redundant, unused ▾

Remove the declaration of thrown exception 'java.lang.IllegalStateException' which is a runtime exception. ··· 6 minutes ago ▾ L35

Code Smell ▾ Minor ▾ Open ▾ Not assigned ▾ 5min effort Comment clumsy, redundant, unused ▾

```
        Class<?>[] paraTypes = {String.class};
        Object[] parameters = {fileUri};
```

Code Smell示例-6

Switch语句缺少Default分支

```
for (int i = 0; i < 7; i++) {
    for (int j = 0; j < 7; j++) {
        // 要改一下，用来读存档
        int tile = Origins.readOriginal().getLandscapeInt()[i][j];
        switch (tile) {
```

Add a default case to this switch. ··· yesterday ▾ L94

Code Smell · Critical · Open · Not assigned · 5min effort · cert, cwe, misra

```
                case 0:
                    tiles[i][j].setType(Tile.TileType.LAND);
                    break;
                case 1:
                    tiles[i][j].setType(Tile.TileType.WATER);
                    break;
                case 2:
                    tiles[i][j].setType(Tile.TileType.TRAPLEFT);
                    break;
                case 3:
                    tiles[i][j].setType(Tile.TileType.HOMELEFT);
                    break;
                case 4:
                    tiles[i][j].setType(Tile.TileType.TRAPRIGHT);
                    break;
                case 5:
                    tiles[i][j].setType(Tile.TileType.HOMERIGHT);
                    break;
            }
        }
```

适合使用if-else的Switch语句

```
while (true) {
    cmd = input.nextLine();
    if (!cmd.contains(" ")) {
        switch (cmd) {
```

Add a default case to this switch. ··· yesterday ▾ L15

Code Smell · Critical · Open · Not assigned · 5min effort · cert, cwe, misra

Replace this "switch" statement by "if" statements to increase readability. ··· yesterday ▾ L15

Code Smell · Minor · Open · Not assigned · 5min effort · bad-practice, misra

```
                case "EXIT":
                    System.exit(0);
                    break;
                case "DUMP":
                    b_dictionary.dump();
            }
        } else {
            command = cmd.split(" ")[0];
```

Code Smell示例-7

Try语句块过长，影响对整个异常处理逻辑的理解

```
2           try {
```

Extract this nested try block into a separate method. ··· 8 minutes ago ▾ L32
Code Smell ▾ Major ▾ Open ▾ Not assigned ▾ 20min effort Comment confusing ▾

```
3               if (args.length!=3){
4                   System.out.println(usage);
```

Replace this use of System.out or System.err by a logger. ··· 8 minutes ago ▾ L34
Code Smell ▾ Major ▾ Open ▾ Not assigned ▾ 10min effort Comment bad-practice, cert ▾

```
5                   System.exit(0);
6               }
7
8               String uri = getUri(args[2]);
9               File input = new File(args[1]);
0
1               FileChannel fi = new FileInputStream(input).getChannel();
2
3               SDFSFileChannel fileChannel = simpleDistributedSystem.create(uri);
4               ByteBuffer byteBuffer = ByteBuffer.allocate(100000);
5               while (fi.read(byteBuffer) >0){
6                   byteBuffer.flip();
7                   fileChannel.write(byteBuffer);
```

Code Smell示例-8

方法只有一种返回值

Refactor this method to not always return the same value. 1 hour ago L9
Code Smell Blocker Open Not assigned 16min effort No tags

Not covered by tests

```
        if (GameController.historyBoard.animalBoardRecord.size() == 2) {
            animalChosen = Board.animalBoard[0][6];
            animalChosen.go(new int[]{0, -1});
            return true;
        }
        if (GameController.historyBoard.animalBoardRecord.size() == 4) {
            animalChosen = Board.animalBoard[4][6];
            animalChosen.go(new int[]{0, 1});
            return true;
        }
        if (GameController.historyBoard.animalBoardRecord.size() == 6 && !Judge.ifDanger(false)) {
            animalChosen = Board.animalBoard[0][5];
            animalChosen.go(new int[]{0, -1});
            return true;
        }


        if (Eat()) {
            return true;
        }

        if (Escape()) {
            return true;
        }

        if (GameController.historyBoard.animalBoardRecord.size() < 16 && Board.animalBoard[2][7].getCamp() !=
            animalChosen = Board.animalBoard[1][7];
            animalChosen.go(new int[]{1, 0});
            return true;
        }
```

Cognitive Complexity of method... × Mutal

没有对空方法做出解释并抛出相应异常

```
    public void jumpRiverDirection(int[] direction) {
```

Add a nested comment explaining why this method is empty, throw an UnsupportedOperationException or complete the implementation. 1 hour ago L75
Code Smell Critical Open Not assigned 5min effort suspicious

```
    }

}
```

Code Smell示例-9

临时变量与类属性重名

```
    private void save(GridPane functionPane, GridPane picturePane, GridPane animalPane, Timeline timeline, Ti
        Label save = new Label("", new ImageView(new Image("file:pic/save.png")));
```

Rename "save" which hides the field declared at line 30. ··· 1 hour ago ▾ L361

Code Smell Major Open Not assigned 5min effort cert, pitfall

```
        functionPane.add(save, 2, 0);
        Label saveBackground = new Label("", new ImageView(new Image("file:pic/savePane.png")));
        picturePane.add(saveBackground, 0, 0);
        saveBackground.setVisible(false);
        GridPane savePane = new GridPane();
        savePane.setAlignment(Pos.CENTER);
        savePane.setVgap(12);
        savePane.setVisible(false);
        Label save1 = new Label("", new ImageView(new Image("file:pic/save1.png")));
        save1.setPadding(new Insets(120, 0, 0, 0));
        Label save2 = new Label("", new ImageView(new Image("file:pic/save2.png")));
        Label save3 = new Label("", new ImageView(new Image("file:pic/save3.png")));
        Label back = new Label("", new ImageView(new Image("file:pic/close2.png")));
        GridPane.setConstraints(save1, 0, 0);
        GridPane.setConstraints(save2, 0, 1);
        GridPane.setConstraints(save3, 0, 2);
```

在非静态方法中修改静态变量（有并发冲突可能）

```
            new Board();
            player = true;
```

Make the enclosing method "static" or remove this set. ··· 1 hour ago ▾ L440

Code Smell Critical Open Not assigned 20min effort multi-threading

```
            View newView = new View();
            newView.start(primaryStage);
        });

        back.setOnMouseClicked(event -> {
            new Board();
            player = true;
```

Make the enclosing method "static" or remove this set. ··· 1 hour ago ▾ L447

Code Smell Critical Open Not assigned 20min effort multi-threading

```
            Start start = new Start();
            start.start(primaryStage);
        });
```

Code Smell示例-10

没有使用StringBuilder而是直接拼接字符串

```
for (int i = 0; i < rootNameLength; i++) {
    rootName += (char) bufferedInputStream.read();
```

Use a StringBuilder instead. 1 hour ago L34

Code Smell Minor Open Not assigned 10min effort performance

```
    count++;
}
```

使用了将要废弃的方法

```
if (bufferedInputStream.available() > 0) {
    for (int i = 0; i < codeByteSize; i++) {
        Integer integer = new Integer(bufferedInputStream.read());
```

Remove this "Integer" constructor 1 hour ago L46

Code Smell Major Open Not assigned 5min effort performance

没有使用最合适的方法

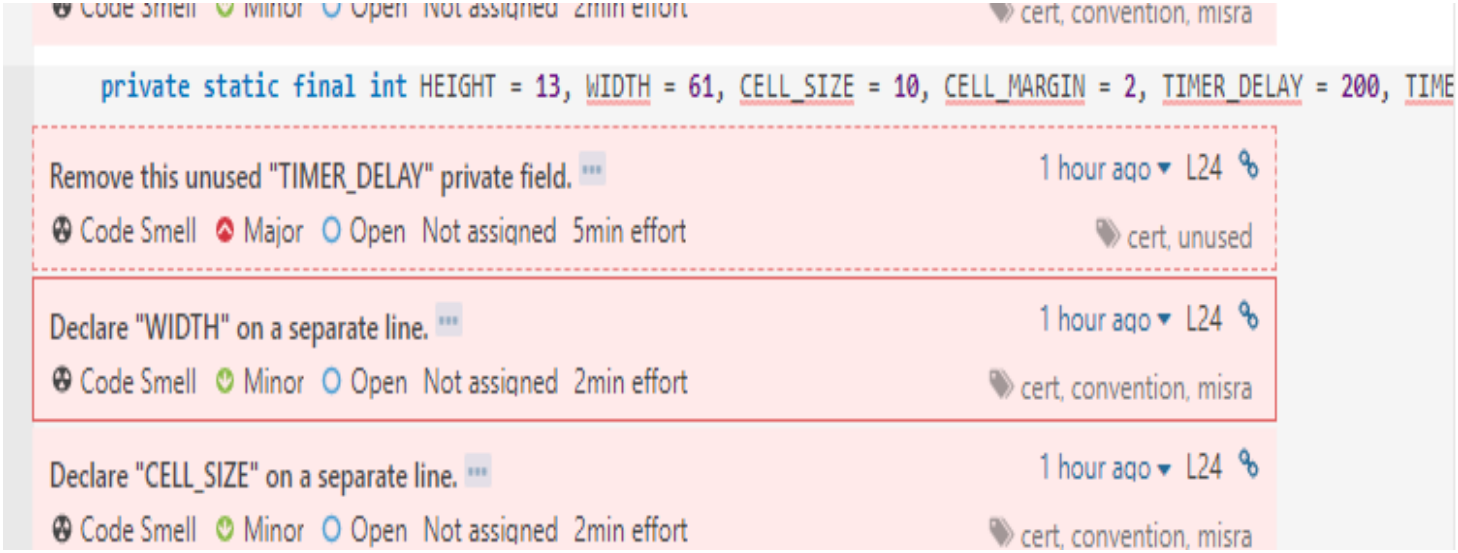
```
        new GameRule("Carnival shuttle -- Robert Wainwright 1984", 38, 7, "33bo3bo$2o3b2o26b5o$bob
this.gameRule = gameRules[(int) (Math.random() * gameRules.length)];
```

Use "java.util.Random.nextInt()" instead. 1 hour ago L209

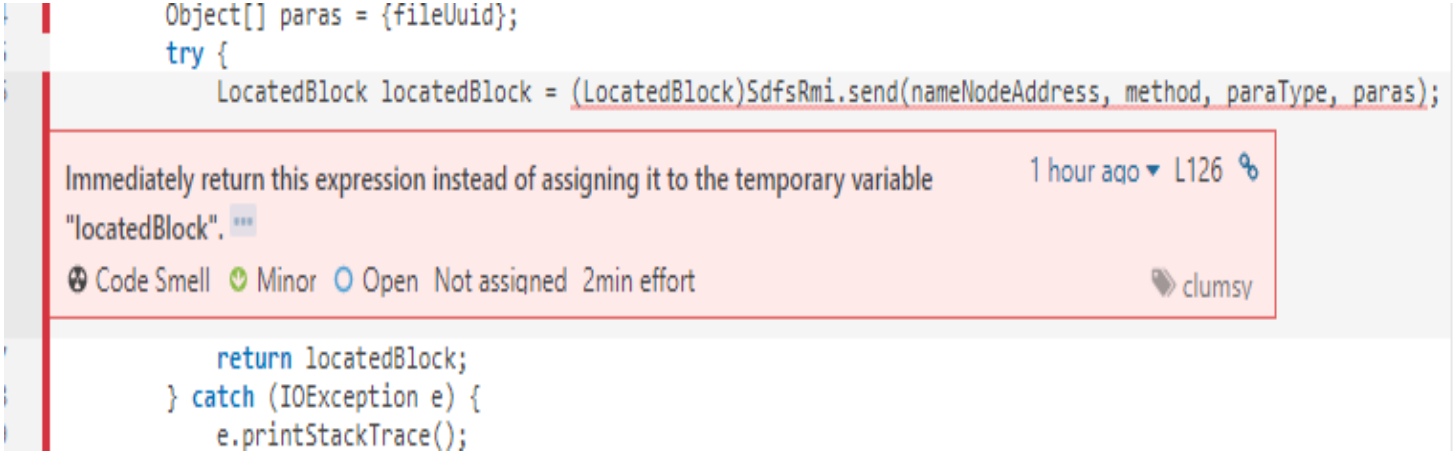
Code Smell Minor Open Not assigned 5min effort clumsy

Code Smell示例-11

在一行上声明了太多的变量



声明了一个马上就被返回的变量



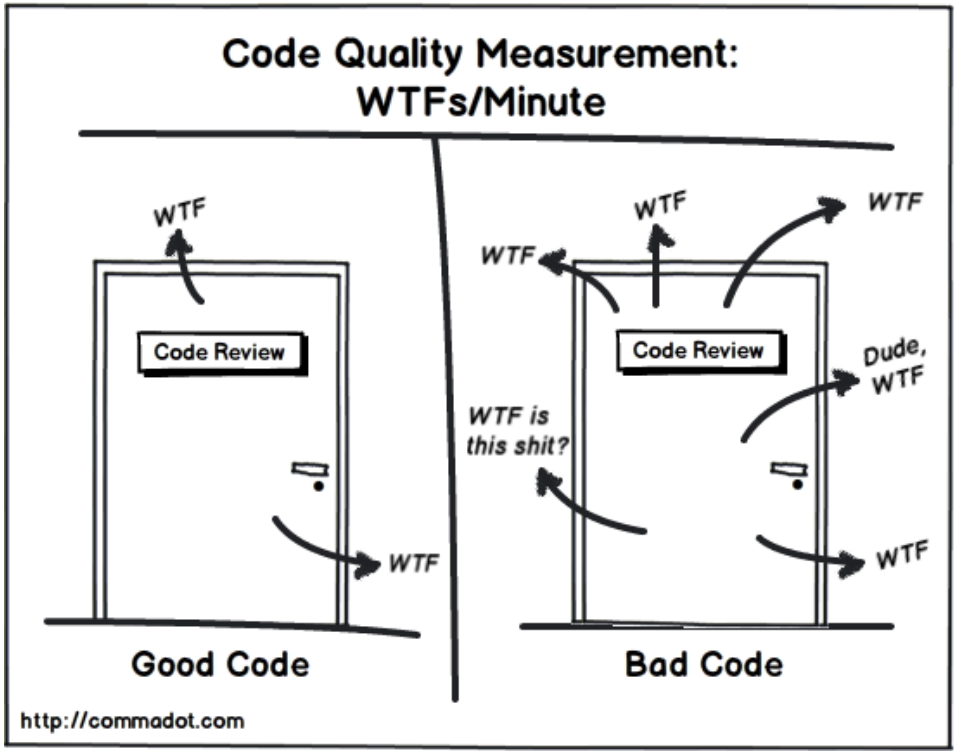
问题总结-1

- API及相关资源的使用不够严谨
  - ✓ API的规范使用模式，如资源使用后确保释放
  - ✓ API的同步和性能特性不够了解
- 程序逻辑不够严密
  - ✓ 条件分支组合考虑不周全，循环终止条件考虑不周
  - ✓ 未考虑方法调用返回值所代表的特殊含义（如返回null）
  - ✓ 异常的理解和处理不当
- 未考虑外部可能的恶意或错误使用
  - ✓ 属性或其他内部信息封装不够
  - ✓ 对于外部输入值的检验和使用不恰当

问题总结-2

- 未考虑“他人”阅读理解代码的问题
  - ✓ 不规范或难以理解的标识符
  - ✓ 令人误解的排版
  - ✓ 空语句或过于复杂的语句（如嵌套条件和嵌套循环）
- 未考虑未来的扩展和修改
  - ✓ 写死的变量，不善于利用配置文件和字符串常量等
  - ✓ 复杂的条件分支，不善于利用继承、多态和设计模式
- 程序各处处理逻辑的统一性不够
  - ✓ 不善于使用logger进行日志记录
  - ✓ 各处的异常处理策略不统一

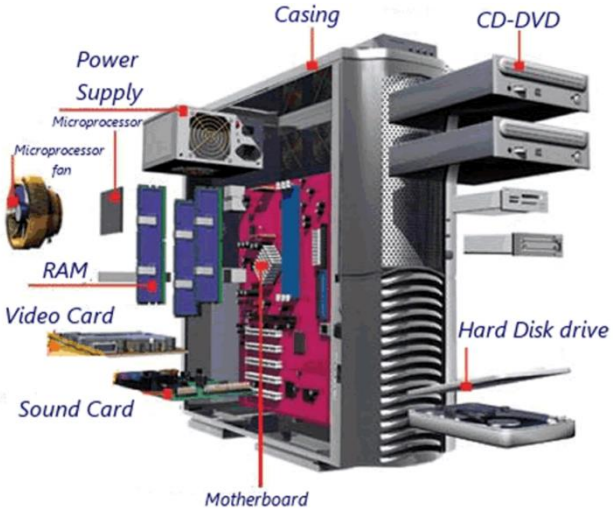
代码质量



程序写出来是给人看的，
只是顺便用作机器执行

—《计算机程序的构造与解释》

代码质量的两面



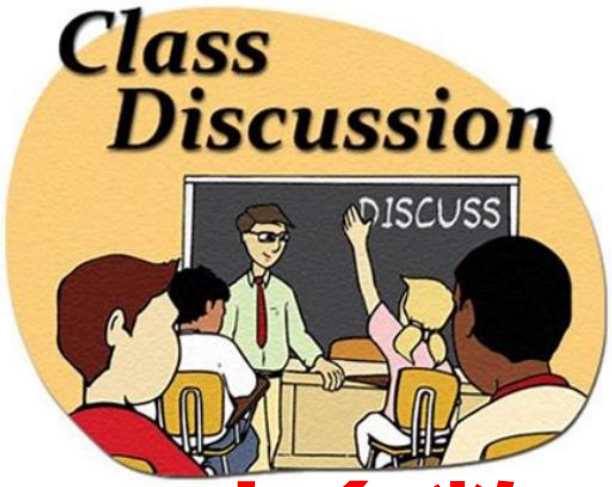
机器（计算机）

代码的运行质量：
功能、性能、可靠性...

人（程序员）

代码的维护质量：
易理解、修改、扩展...

课堂讨论：排序算法



课堂讨论：你认为最好的排序算法是哪一个?

大多数时候的选择：冒泡排序

逻辑简单

广泛使用

易理解

代码质量的含义

- 易于理解和修改
- 易于扩展
- 不容易出错
- 不容易被外部破坏
- 性能和可靠性好
- 实现决策的整体性和统一性

代码质量的几个方面

- 代码风格
- 代码逻辑
- 代码设计（软件设计部分）

代码风格示例

```
#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]);
}
```

```
//                    _ooOoo_
//                   o8888888o
//                   88" . "88
//                   (| -_- |)
//                    O\ = /O
//                ____/`---'\____
//              .   ' \\| |// `.
//               / \\||| : |||// \
//             / _||||| -:- |||||- \
//               | | \\\ - /// | |
//             | \_| ''\---/'' | |
//              \ .-\__ `-` ___/-. /
//           ___`. .' /--.--\ `. . __
//        ."" '< `.___\_<|>_/___.' >'"".
//       | | : `- \`.;`\ _ /`;.`/ - ` : | |
//         \ \ `-. \_ __\ /__ _/ .-` / /
// ======`-.____`-.___\_____/___.-`____.-'======
//                    `=---='
//
// .............................................
//          佛祖保佑             永无BUG
//     佛曰:
//          写字楼里写字间，写字间里程序员；
//          程序人员写程序，又拿程序换酒钱。
//          酒醒只在网上坐，酒醉还来网下眠；
//          酒醉酒醒日复日，网上网下年复年。
//          但愿老死电脑间，不愿鞠躬老板前；
//          奔驰宝马贵者趣，公交自行程序员。
//          别人笑我忒疯癫，我笑自己命太贱；
//          不见满街漂亮妹，哪个归得程序员？
```

聪明的代码？
艺术的代码？
想想维护者的感受就行！

代码风格规范

- 缩进
- 括号
- 分行
- 大小写
- 命名
- 注释

…

基本原则：

1）排版反映逻辑结构

2）命名遵循规范，通俗易懂

3）利用注释充分揭示代码意图

4）风格保持一致

详见《构建之法》4.1-4.2

《代码大全》第11、19章

不良代码风格示例-1

```
function createJxbObj(xnxqdm, jxbid, zjxbid, kcdm, kcmc, bjmc, kkdw, kkdwmc,
 xs, zxs, xf, bjrnrs, xqdm, xqmc, zjls, skxx, xxkcxx, sfbhysb,sftykczx,
kcxzdm,kclbdm,yjkcdm) {
        var jxbObj = new Object();
        jxbObj.xnxqdm = xnxqdm;
        jxbObj.jxbid = jxbid;
        jxbObj.zjxbid = zjxbid;
        jxbObj.kcdm = kcdm;
        jxbObj.kcmc = kcmc;
        jxbObj.bjmc = bjmc;
        jxbObj.kkdw = kkdw;
        jxbObj.kkdwmc = yxObjList[kkdw];
        jxbObj.xs = xs;
        jxbObj.zxs = zxs;
        jxbObj.xf = xf;
        jxbObj.bjrnrs = bjrnrs;
        jxbObj.xqdm = xqdm;
        jxbObj.xqmc = xqiuMap[xqdm];
        jxbObj.zjls = zjls;
        jxbObj.skxx = skxx;
        jxbObj.xxkcxx = xxkcxx;
        jxbObj.sfbhysb = sfbhysb;
        jxbObj.pkxxlist = new Array();
```

拼音缩写命名、参数过多

```
    type     :'POST',
    url      : "../model/IVLEDataMa
    data     : {
        module_code : code,
        CourseStaff : lecturers,
        AcaYear : academicYear,
        Sem : semester
    },
    success : function (data) {
```

命名风格不统一

```
enum FiveLine{
  Jin,
  Wood,
  Water,
  Huo,
  Earth,};
```

拼音与英文命名混杂

不良代码风格示例-2

```
1
2
3
4   //This loop starts the i from 0 to len, in each step, it
5   //does Something
6   for(i = 0; i < len; i++){
7       DoSomeThing();
8   }
9
```

无意义的注释（应当解释高层意图而非实现本身）

```
17              if (rf.getVersion().equals("最新价格"))
18              {// 实时查询
1257            else
1258            {// 常规查询
2948            db.endTransaction();
```

将近1700行的else分支

```
if (condition) {
    if (conditionA) {
        if (conditionB) {
            if (conditionC) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
} else {
    return false;
}
```

复杂的嵌套结构

直截了当而不是故弄玄虚

```
//使用空语句
//方法一：让空语句中的分号自占一行，并且加以缩进，就像对待其他的语句一样
while ( recordArray.Read( index++ ) != recordArray.EmptyRecord() ) {
    ;
}

//方法二：用一组空的括号强调该空语句
while ( recordArray.Read( index++ ) != recordArray.EmptyRecord() ) {}

//方法三：为空语句创建一个DoNothing()函数.
//        这条语句什么也不做，却可以毫无争议地表明“这里不希望做任何事情”的用意
while ( recordArray.Read( index++ ) != recordArray.EmptyRecord()){
    doNothing();
}

//方法四：更加清晰的重写后的代码，采用非空循环体
RecordType record = recordArray.Read( index );
index++;
while( record != recordArray.EmptyRecord() ) {
    record = recordArray.Read( index );
    index++;
}
```

代码逻辑考虑

- 控制代码复杂度
- 控制代码重复率
- 高质量的子程序
- 防御式编程

控制复杂度

- 应当以某种方式去组织程序，使得我们在同一时刻可以只关注于一个特定的部分
- 需要同时关注的东西越多，越容易犯错误
- 基本手段：分解和抽象

没有谁的大脑能容得下一个现代的计算机程序。

— Edsger Dijkstra

控制代码复杂度

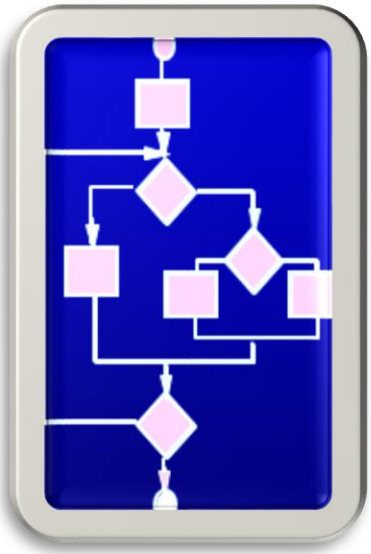
- 控制代码长度
  - ✓ 缺少抽象和分解，需要一下子理解很长的代码
  - ✓ 例如：长类、长方法、长代码块（如条件分支）
- 控制嵌套语句的层数
  - ✓ 嵌套的循环和/或条件分支语句
  - ✓ 很少有人能轻松理解超过3层的嵌套语句

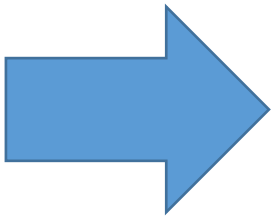
控制代码长度

- 消除冗余代码和死代码（dead code）
- 适当分解，提取公共部分消除重复代码
- 在各个类和方法之间适当分配职责（如界面和后台逻辑分离）

代码复杂度

- 代码长度（类/方法）
- 圈复杂度
  - ✓ 程序流图中简单判断结点数量加1
  - ✓ 或者是程序流图中封闭区域数量加1





V(G) = 3+1 = 4

控制圈复杂度

- 容忍范围
  - ✓ 0-5　还不错
  - ✓ 6-10　得想办法简化了
  - ✓ 10+　要考虑分拆子程序了
- 整个程序的复杂度保持不变，但是同一时间需要关注的复杂度下降了
- 10+非绝对标准，主要取决于人的整体复杂度认知
  - ✓ 如大量并列的简单case分支也许可以接受

糟糕的深层嵌套代码

```
//糟糕的深层嵌套代码
if ( inputStatus == inputStatus_Success ) {
    //...Lots of Code
    if ( pringterRoutine != NULL ) {
        //...Lots of Code
        if( SetupPage() ) {
            //...Lots od Code
            if ( AllocMem( &printData ) ) {
                //...Lots od Code
            }
        }
    }
}
```

消除深层嵌套代码

```
//通过重复检测条件中的某一部分来简化嵌套的if语句
if ( inputStatus == inputStatus_Success ) {
    //...Lots of Code
    if ( pringterRoutine != NULL ) {
        //...Lots of Code
    }
}

if ( inputStatus == inputStatus_Success &&
    (pringterRoutine != NULL) && SetupPage() ) {
        //...Lots of Code
        if ( AllocMem( &printData ) ) {
            //...Lots of Code
        }
}
```

通过重复检测条件中的某一部分来简化嵌套的if语句

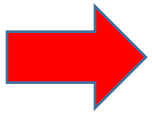
消除深层嵌套代码

```
//使用break块来简化嵌套if
do {
    //begin break block
    if ( inputStatus != inputStatus_Success ) {
        break; //break out of block
    }
    //...Lots of Code
    if ( pringterRoutine == NULL ) {
        break; //break out of block
    }
    //...Lots of Code
    if( !SetupPage() ) {
        break; //break out of block
    }
    //...Lots of Code
    if ( !AllocMem( &printData ) ) {
        break; //break out of block
    }
    //...Lots of Code
} while (false);
```

用break块来简化嵌套的if语句

消除深层嵌套代码

```
//茂盛的决策树
if ( 10 < quantity ) {
    if ( 100 < quantity ) {
        if ( 1000 < quantity ) {
            discount = 0.10;
        } else {
            discount = 0.05;
        }
    } else {
        discount = 0.025;
    }
} else {
    iscount = 0.0;
}
```



```
//将嵌套的if语句转换为一组if-then-else语句
if ( 1000 < quantity ) {
    discount = 0.10;
} else if ( 100 < quantity ) {
    discount = 0.05;
} else if ( 10 < quantity ) {
    discount = 0.025;
} else {
    discount = 0;
}
```

将嵌套if语句转换为if-then-else结构

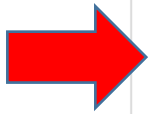
消除深层嵌套代码

```
//VB语言 将嵌套if语句转换为case语句
Select Case quantity
    Case 0 To 10
        discount = 0.0
    Case 11 To 100
        discount = 0.025
    Case 101 To 1000
        discount = 0.05
    Case Else
        discount = 0.10
End Select
```

将嵌套if语句转换为case语句

消除深层嵌套代码

```
//需要分入子程序的嵌套代码
while ( !TransactionsComplete() ) {
    // read transaction record
    transaction = ReadTransaction();
    // process transaction depending on type of transaction
    if ( transaction.Type == TransactionType_Deposit ) {
        // process a deposit
        if ( transaction.AccountType == AccountType_Checking ) {
            if ( transaction.AccountSubType == AccountSubType_Business )
                MakeBusinessCheckDep( transaction.AccountNum, transaction.Amount );
            else if ( transaction.AccountSubType == AccountSubType_Personal )
                MakePersonalCheckDep( transaction.AccountNum, transaction.Amount );
            else if ( transaction.AccountSubType == AccountSubType_School )
                MakeSchoolCheckDep( transaction.AccountNum, transaction.Amount );
        } else if ( transaction.AccountType == AccountType_Savings )
            MakeSavingsDep( transaction.AccountNum, transaction.Amount );
        else if ( transaction.AccountType == AccountType_DebitCard )
            MakeDebitCardDep( transaction.AccountNum, transaction.Amount );
        else if ( transaction.AccountType == AccountType_MoneyMarket )
            MakeMoneyMarketDep( transaction.AccountNum, transaction.Amount );
        else if ( transaction.AccountType == AccountType_Cd )
            MakeCDDep( transaction.AccountNum, transaction.Amount );
    } else if ( transaction.Type == TransactionType_Withdrawal ) {
        // process a withdrawal
        if ( transaction.AccountType == AccountType_Checking )
            MakeCheckingWithdrawal( transaction.AccountNum, transaction.Amount );
        else if ( transaction.AccountType == AccountType_Savings )
            MakeSavingsWithdrawal( transaction.AccountNum, transaction.Amount );
        else if ( transaction.AccountType == AccountType_DebitCard )
            MakeDebitCardWithdrawal( transaction.AccountNum, transaction.Amount );
    } else if ( transaction.Type == TransactionType_Transfer ) {
        MakeFundsTransfer(
            transaction.SourceAccountType,
            transaction.TargetAccountType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else {
        // process unknown kind of transaction
        LogTransactionError( "Unknown Transaction Type", transaction );
    }
}
```

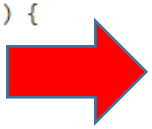


```
//将嵌套代码分解到子程序后的好代码
while ( !TransactionsComplete() ) {
    // read transaction record
    transaction = ReadTransaction();
    // process transaction depending on type of transaction
    if ( transaction.Type == TransactionType_Deposit ) {
        ProcessDeposit(
            transaction.AccountType,
            transaction.AccountSubType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else if ( transaction.Type == TransactionType_Withdrawal ) {
        ProcessWithdrawal(
            transaction.AccountType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else if ( transaction.Type == TransactionType_Transfer ) {
        MakeFundsTransfer(
            transaction.SourceAccountType,
            transaction.TargetAccountType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else {
        // process unknown transaction type
        LogTransactionError("Unknown Transaction Type", transaction );
    }
}
```

将深层嵌套代码抽取出来放入单独的子程序

消除深层嵌套代码

```
//将嵌套代码分解到子程序后的好代码
while ( !TransactionsComplete() ) {
    // read transaction record
    transaction = ReadTransaction();
    // process transaction depending on type of transaction
    if ( transaction.Type == TransactionType_Deposit ) {
        ProcessDeposit(
            transaction.AccountType,
            transaction.AccountSubType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else if ( transaction.Type == TransactionType_Withdrawal ) {
        ProcessWithdrawal(
            transaction.AccountType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else if ( transaction.Type == TransactionType_Transfer ) {
        MakeFundsTransfer(
            transaction.SourceAccountType,
            transaction.TargetAccountType,
            transaction.AccountNum,
            transaction.Amount
        );
    } else {
        // process unknown transaction type
        LogTransactionError("Unknown Transaction Type", transaction );
    }
}
```



```
//分解后的嵌套代码 - 使用case语句
while ( !TransactionsComplete() ) {
    // read transaction record
    transaction = ReadTransaction();
    // process transaction depending on type of transaction
    switch ( transaction.Type ) {
        case ( TransactionType_Deposit ):
            ProcessDeposit(
                transaction.AccountType,
                transaction.AccountSubType,
                transaction.AccountNum,
                transaction.Amount
            );
            break;
        case ( TransactionType_Withdrawal ):
            ProcessWithdrawal(
                transaction.AccountType,
                transaction.AccountNum,
                transaction.Amount
            );
            break;
        case ( TransactionType_Transfer ):
            MakeFundsTransfer(
                transaction.SourceAccountType,
                transaction.TargetAccountType,
                transaction.AccountNum,
                transaction.Amount
            );
            break;
        default:
            // process unknown transaction type
            LogTransactionError("Unknown Transaction Type", transaction );
            break;
    }
}
```

进一步转化为清晰的case语句

消除深层嵌套代码

```
//使用多态机制的好代码
TransactionData transactionData;
Transaction *transaction;
while ( !TransactionsComplete() ) {
    // read transaction record
    transactionData = ReadTransaction();
    // create transaction object, depending on type of transaction
    switch ( transactionData.Type ) {
        case ( TransactionType_Deposit ):
            transaction = new Deposit( transactionData );
            break;
        case ( TransactionType_Withdrawal ):
            transaction = new Withdrawal( transactionData );
            break;
        case ( TransactionType_Transfer ):
            transaction = new Transfer( transactionData );
            break;
        default:
            // process unknown transaction type
            LogTransactionError("Unknown Transaction Type", transactionData );
            return;
    }
    transaction -> Complete();
    delete transaction;
}
```

使用多态等面向对象特性

消除深层嵌套代码

```
//使用多态机制和Object Factory的好代码
TransactionData transactionData;
Transaction *transaction;
while ( !TransactionsComplete() ) {
    // read transaction record and complete transaction
    transactionData = ReadTransaction();
    transaction = TransactionFactory.Create( transactionData );
    transaction->Complete();
    delete transaction;
}
Transaction *TransactionFactory::Create(TransactionData transactionData) {
        // create transaction object, depending on type of transaction
        switch ( transactionData.Type ) {
            case ( TransactionType_Deposit ):
                return new Deposit( transactionData );
                break;
            case ( TransactionType_Withdrawal ):
                return new Withdrawal( transactionData );
                break;
            case ( TransactionType_Transfer ):
                return new Transfer( transactionData );
                break;
            default:
                // process unknown transaction type
                LogTransactionError( "Unknown Transaction Type", transactionData );
                return NULL;
}
```

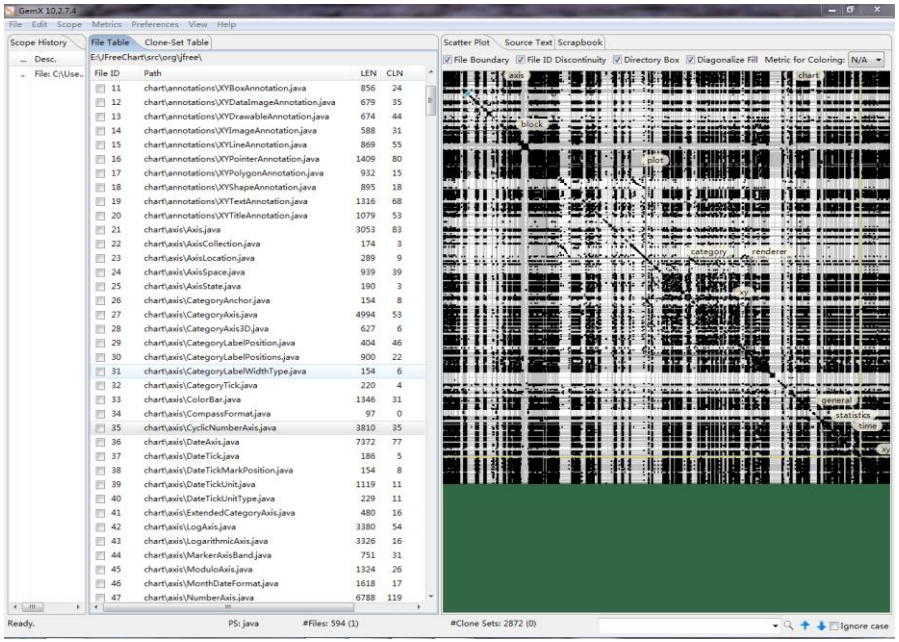
进一步使用面向对象的工厂（Factory Pattern）模式

善用面向对象特性

- case语句的使用经常说明代码的分解没有做好
  - ✓ 例如，可以通过多态消除case语句
- 深层嵌套的出现是一个警告，说明应该进行子程序分拆或重新设计

重复代码（代码克隆）

- 相同或相似的代码片段
  - ✓ 有意识的克隆：通过代码复制实现复用
  - ✓ 偶然克隆：模式化的解决方案导致相似的实现
  - ✓ 恶意克隆：出于增加代码长度等目的而复制代码
- 危害：增加复杂度、一致性维护负担、传播缺陷



克隆侦测工具CCFinder

```
0  PertProject.java:
1  public void init() {
2    super.init();
3    ...
4    view.setDOMFactory(
5      new PertFactory());
6    undo = new UndoRedoManager();
7    view.setDrawing(
8      new DefaultDrawing());
9    view.getDrawing().addUndoableEditListener
           (undo);
0    ...
1  }
```

```
NetProject.java:
public void init() {
  super.init();
  ...
  view.setDOMFactory(
    new NetFactory());
  undo = new UndoRedoManager();
  view.setDrawing(
    new DefaultDrawing());
  view.getDrawing().addUndoableEditListener
         (undo);
  ...
}
```

```
SVGProject.java:
public void init() {
  super.init();
  ...
  view.setDOMFactory(
    new SVGFigureFactory());
  undo = new UndoRedoManager();
  view.setDrawing(
    new SVGDrawing());
  view.getDrawing().addUndoableEditListener
         (undo);
  ...
}
```

JHotDraw中的代码克隆片段

控制代码重复率

- 提取公共部分进行方法或类封装
- 提取抽象类或接口
- 使用模板类或模板方法

重复代码也是一种灵活的代码复用手段，
绝对消除重复代码不太现实

子程序的作用

- 子程序：方法、过程或函数
- 子程序的作用
  - ✓ 代表一种逻辑分解（同时也是一种功能抽象）
  - ✓ 将相同的实现合并在一起，便于编写、调试、维护
- 创建子程序的理由
  - ✓ 引入一种易理解的中间抽象
  - ✓ 避免代码重复
  - ✓ 支持子类扩展
  - ✓ 隐藏复杂实现细节，如执行顺序、指针操作、复杂判断等
  - ✓ 提高可移植性，将可移植与不可移植的部分区分开

高质量的子程序

- 承担单一而明确的职责（功能内聚）
- 良好的子程序命名
- 控制子程序的长度
- 仔细考虑子程序的参数

良好的子程序命名-1

- 尽量使用语气强烈的动宾结构
  - ✓ 反映动作及其施加对象
  - ✓ 例如，使用printDocument()、checkOrderInfo()
- 描述子程序所做的所有事情
  - ✓ 例如，使用computeReportTotalsAndOpenOutputFile()而不是computeReportTotals()
  - ✓ 过长的名字表示子程序担负的职责太多，应该考虑分解
  - ✓ 类名/对象名已经为子程序提供了部分信息
- 不要使用数字来区分子程序
  - ✓ 例如，避免使用processData1()、processData2()
  - ✓ 按照功能逻辑而不是物理因素（如代码长度）进行分解
- 避免使用无意义或模糊不清的动词
  - ✓ 例如，避免使用performService()、handleInput()
  - ✓ 明确表明子程序的功能，如功能本身很模糊则要考虑调整

良好的子程序命名-2

- 对返回值有所描述
  - ✓ 让别人很容易理解子程序执行完之后返回结果的含义
  - ✓ 例如，cos()、isReady()、getCurrentColor()
- 准确使用对仗词
  - ✓ 对于成对出现的子程序使用对仗词进行命名
  - ✓ 例如，add/remove、lock/unlock、start/stop、up/down、get/set、open/close、show/hide 、create/destroy
- 为常用操作建立规范
  - ✓ 尽量以统一的方式提供常用操作的访问入口
  - ✓ 例如，对于学生类Student和导师类Advisor
    - ■ 统一的访问：Student.getAdvisor()及Advisor.getName()
    - ■ 不统一的访问： Student.getAdvisorName()及Advisor.getName()

控制子程序的长度

- 一般不应该超过200行
- 同时考虑以下因素
  - ✓ 子程序的功能内聚性
  - ✓ 子程序代码的嵌套层数
  - ✓ 子程序的变量数量
  - ✓ 子程序的决策点（条件判断）数量
  - ✓ 解释子程序意图所需要的注释数量

仔细考虑子程序的参数

- 子程序间的接口是最容易出错的部分之一，与参数的传递和使用密切相关
- 参数定义及使用
  - ✓ 类似参数的排列顺序保持一致
  - ✓ 使用所有的参数
  - ✓ 不要把参数用作工作变量
  - ✓ 对于参数的假定要加以明确（注释、断言）：契约式设计、防御式编程
  - ✓ 参数的个数不要太多（一般不超过7个）
    - ■ 参数太多说明子程序之间耦合过紧，也许需要调整设计结构
    - ■ 使用对象来封装所传递的参数

不要把参数用作工作变量

输入参数inputVal被用作工作变量

Java Example of Improper Use of Input Parameters

```
int Sample( int inputVal ) {
   inputVal = inputVal * CurrentMultiplier( inputVal );
   inputVal = inputVal + CurrentAdder( inputVal );
   ...
   return inputVal;
}
```

At this point, *inputVal* no longer contains the value that was input.

声明一个新的工作变量workingVal

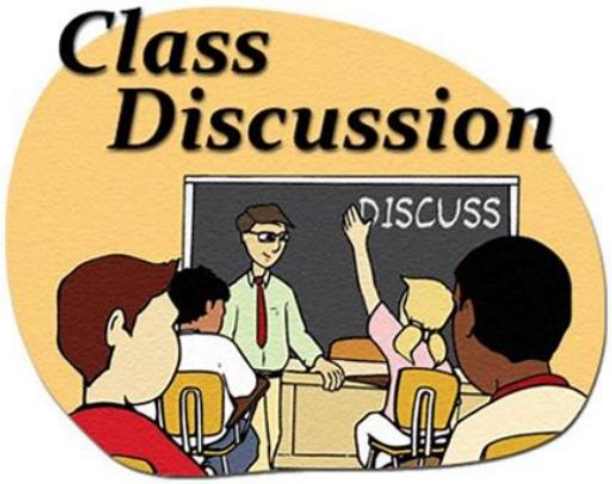
Java Example of Good Use of Input Parameters

```
int Sample( int inputVal ) {
   int workingVal = inputVal;
   workingVal = workingVal * CurrentMultiplier( workingVal );
   workingVal = workingVal + CurrentAdder( workingVal );
   ...

   ...
   return workingVal;
}
```

If you need to use the original value of *inputVal* here or somewhere else, it's still available.

课堂讨论：使用对象为子程序传递参数



课堂讨论：一个子程序需要访问一个包含10项属性的对象的3项属性，那么应该如何传递参数？

1）直接将3项属性作为参数传递

2）传递整个对象

方案1：尽量减少子程序之间的耦合，更加容易理解和复用

方案2：参数更加简洁，被调用子程序具有使用属性的灵活度

判断依据：被调用子程序的逻辑依赖到底是整个对象还是3项数据？

方案1：依赖3项数据，这些数据可以来自各种不同的来源（例如传递数据时临时创建一个仅为了数据传输的对象）

例如：利用一个加法子程序为一个学生对象计算总成绩

方案2：依赖整个对象，这些数据只能来自这个对象且未来有可能要利用对象的其他属性（例如需要经常修改参数表但参数都来自同一对象）

例如：利用一个学籍打印子程序为一个学生对象打印学籍信息

防御式编程

- 基本思想：子程序不应该由于传入错误的数据（由外部接口或其他子程序传入）而被破坏
- 程序员可以决定自己的程序逻辑但无法决定用户或其他子程序提供什么样的输入
- 但仍要竭尽所能保护自身程序的正确运行

防御性驾驶

预估风险
放眼远方
顾全大局
留有余地
引人注意

你永远不知道别人要做什么！

应对不可控的外部输入

- 检查来自用户、文件、网络或其他外部接口的输入
  - ✓ 数值：范围在可接受范围内
  - ✓ 字符串：长度在规定范围内
  - ✓ ID等标识符：在合法列表中
  - ✓ 注意潜在的外部攻击：可能导致缓冲区溢出、SQL命令/HTML/XML注入、整数溢出、将传递给系统调用的数据
- 检查来自其他子程序的输入：与外部输入相似
- 以适当的方式处理非法的外部输入

断言（assertion）

- 在开发期间加入的，让程序在运行时进行自检的代码（通过子程序或宏）
  - ✓ 例如，输入参数取值或返回值满足一定的条件
  - ✓ 断言为真：程序运行符合预期
  - ✓ 断言为假：程序运行发生了意料之外的错误
- 作用：快速定位问题

断言示例（Java）：包含断言为真时的条件以及为假时的显示消息

```
assert denominator != 0: "denominator is unexpectedly equal to 0.";
```

断言的使用

- 常用检查条件
  - ✓ 输入或输出参数的取值处于预期的范围内
  - ✓ 指针或对象引用非空
  - ✓ 子程序开始或结束执行时相关资源处于合理的状态（如文件或流打开或关闭）
  - ✓ 输入变量的值没有被修改
  - ✓ 多种不同计算方式的结果相同
- 主要用于开发和维护阶段，通常不应进入产品代码，以免影响系统性能和正常运行
- 可以理解为一种可以执行的“注释”

进一步的断言使用建议

- 避免把需要执行的代码放到断言中
- 用错误处理代码来处理预期会发生的情况，用断言来处理不应该发生的状况
  - ✓ 错误处理代码：为运行时可能出现的错误情况提供处理
  - ✓ 断言：如果被触发则意味着程序有问题，需要进行修复
- 用断言来注解并验证前置和后置条件
  - ✓ 契约式设计（design by contract）的一部分
  - ✓ 前置条件：调用方在调用子程序时需要满足的条件，表示调用方应该承担的义务
  - ✓ 后置条件：子程序在执行结束后需要满足的条件，表示被调用的子程序应该承担的义务

断言与错误处理代码示例

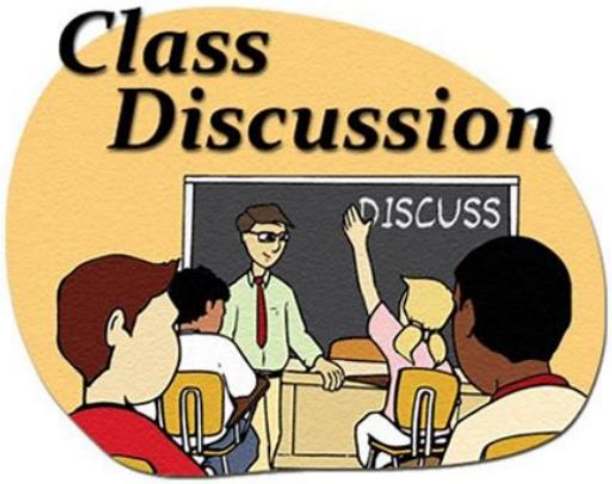
```
4   public void showScene(int latitude, int longitude, int elevation) {
5       assert -90 <= latitude && latitude <= 90: "unexpected latitude";
6       assert 0 <= longitude && longitude <= 360: "unexpected longitude";
7       assert -500 <= elevation && elevation <= 75000: "unexpected elevation";
8       ...
9   }
```

断言

```
12  public void showScene(int latitude, int longitude, int elevation) {
13      if(-90 > latitude)
14          latitude = -90;
15      else if(latitude > 90)
16      latitude = 90;
17
18      if(0 > longitude)
19          longitude = 0;
20      else if(longitude > 360)
21      longitude = 360;
22
23      if(-500 > elevation)
24          elevation = -500;
25      else if(elevation > 75000)
26      elevation = 75000;
27      ...
28  }
```

错误处理代码

课堂讨论：错误处理代码和断言的使用



课堂讨论：以下两种情况下应当使用错误处理代码还是断言来处理超范围的输入值？

1）输入值来自用户输入

2）输入值来自本程序中其他方法

情况1：使用错误处理代码（例如提示重新输入）

情况2：使用断言

情况1：用户提供了错误的输入（例如，错误格式的身份证号、不符合要求的用户名或密码等）是可预期的，且应该进行适当处理（例如，提示重新输入）

情况2：程序自身出错，表明各个子程序之间对于交互契约的理解不一致或者程序此前发生了内部错误

错误处理方法

- 重试，例如重新读取、请用户重新输入
- 使用中立值替换，如0、空字符串等
- 使用与前次相同的数据
- 换用最接近的合法值
- 将警告信息记录到日志文件中
- 返回一个错误码或显示出错消息
- 调用统一的错误处理子程序
- 关闭程序

注意考虑采取本地化还是全局错误处理策略

选取适当的错误处理方法

根据具体功能逻辑和使用场景选取合适的错误处理方法，否则…

显示病人X光检查结果的软件使用缺省值显示片子

AMT机读取卡号失败时使用上一次读取的银行卡号

错误处理决策的两种倾向

- 正确性（Correctness）：永远不要返回不正确的结果，哪怕不返回结果
  - ✓ 典型代表：人身安全攸关的软件
  - ✓ 例如，放射线治疗仪宁愿停止工作也不能按照一个错误的放射量对病人进行治疗
- 健壮性（Robustness）：保证软件可以持续运转，哪怕偶尔返回一些不够准确的结果
  - ✓ 典型代表：面向终端消费者的软件
  - ✓ 例如，用户能够容忍字处理软件显示文档时暂时缺少部分内容（刷新后重试）却不太容忍其异常退出

错误处理方式的高层考虑

- 错误信息会在不同层次的子程序间传递
- 需要在高层的体系结构设计层面上确定统一的错误处理策略
  - ✓ 例如，底层子程序负责返回错误码，高层子程序负责最终的处理
  - ✓ 需明确各个层次上的子程序在错误处理中的角色
- 确保对于子程序返回值进行检查和处理
  - ✓ 不要忽略可能的错误信息，哪怕你确信子程序不会出错
  - ✓ 防御式编程要求我们预防意想不到的错误（你不可能永远掌控程序的每一个部分）

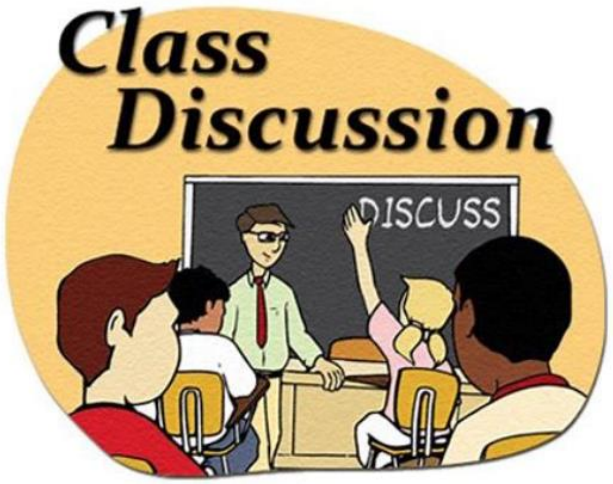
异常（Exception）

- 将代码中的错误或异常事件传递给调用方代码的一种手段
- 将问题处理的控制权交给系统中其他可以更好解释问题并采取措施的部分
- 审慎明智的使用可以降低复杂度，而草率粗心的使用会让代码变得无法理解
  - ✓ 弱化了封装性：调用代码需要了解被调用代码中可能抛出的异常
  - ✓ 增加了复杂度：需要考虑对各种异常的处理

将异常当作正常处理逻辑的一部分的那种程序，都会遭受与所有典型的意大利面条式代码同样的可读性和可维护性问题。

— Andy Hunt、Dave Thomas

课堂讨论：异常与错误处理



课堂讨论：以下两种银行交易处理过程中的错误应当使用异常还是错误处理代码（返回错误码）？

1）账户号不存在

2）网络传输超时

情况1：返回错误码

情况2：抛出异常

情况1：错误码表示可以预见的错误情况，往往属于业务处理的一部分，后续可以请用户重新输入

情况2：异常表示真正的意外情况（如网络、文件、数据库访问问题），需要进行技术处理

异常的使用方式-1

- 用异常通知程序其他部分，发生了不可忽略的错误
- 不能用异常来推卸责任，仅当在本地无法处理错误时抛出异常
- 避免在构造/析构函数中抛出异常，除非在同一个地方捕获处理
- 在异常消息中加入理解异常抛出原因所需要的全部信息

异常的使用方式-2

- 在适当的抽象层次抛出异常

一个关于雇员信息的类将一个底层的文件访问异常抛给了调用方，破坏了封装性和程序的逻辑层次，同时导致调用方代码与底层文件处理异常耦合

Here is the declaration of the exception that's at an inconsistent level of abstraction.

Bad Java Example of a Class that Throws an Exception at an Inconsistent Level of Abstraction

```
class Employee {
   ...
   public TaxId GetTaxId() throws EOFException {
      ...
   }
   ...
}
```

将底层文件访问异常映射为雇员数据不可用的高层异常，从而与当前的抽象层次相符

Here is the declaration of the exception that contributes to a consistent level of abstraction.

Good Java Example of a Class that Throws an Exception at a Consistent Level of Abstraction

```
class Employee {
   ...
   public TaxId GetTaxId() throws EmployeeDataNotAvailable {
      ...
   }
   ...
}
```

异常的使用方式-3

- 避免使用空的catch语句

要么try语句块有问题（无故抛出一个异常），要么catch语句块有问题（没能有效处理一个异常）

Bad Java Example of Ignoring an Exception

```
try {
    ...
    // lots of code
   ...
} catch ( AnException exception ) {
}
```

偶尔会遇到一个无法在当前抽象层次上处理的底层异常，此时应该明确进行文档化说明（通过注释、日志等）

Good Java Example of Ignoring an Exception

```
try {
    ...
    // lots of code
   ...
} catch ( AnException exception ) {
    LogError( "Unexpected exception" );
}
```

异常的使用方式-4

- 将异常作为接口声明的一部分，同时了解所使用的API或函数库所抛出的异常

java.io

Class BufferedInputStream

java.lang.Object
  java.io.InputStream
    java.io.FilterInputStream
      java.io.BufferedInputStream

All Implemented Interfaces:

Closeable, AutoCloseable

read

```
public int read()
         throws IOException
```

See the general contract of the `read` method of `InputStream`.

Overrides:

`read` in class `FilterInputStream`

Returns:

the next byte of data, or `-1` if the end of the stream is reached.

Throws:

`IOException` - if this input stream has been closed by invoking its `close()` method, or an I/O error occurs.

See Also:

`FilterInputStream.in`

异常的使用方式-5

- 考虑创建一个集中的异常报告机制

Visual Basic Example of a Centralized Exception Reporter, Part 1

集中定义的异常报告机制

```
Sub ReportException( _
    ByVal className, _
    ByVal thisException As Exception _
)
    Dim message As String
    Dim caption As String

    message = "Exception: " & thisException.Message & "." & ControlChars.CrLf & _
        "Class:     " & className & ControlChars.CrLf & _
        "Routine: " & thisException.TargetSite.Name & ControlChars.CrLf
    caption = "Exception"
    MessageBox.Show( message, caption, MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation )

End Sub
```

Visual Basic Example of a Centralized Exception Reporter, Part 2

使用集中定义的异常报告机制

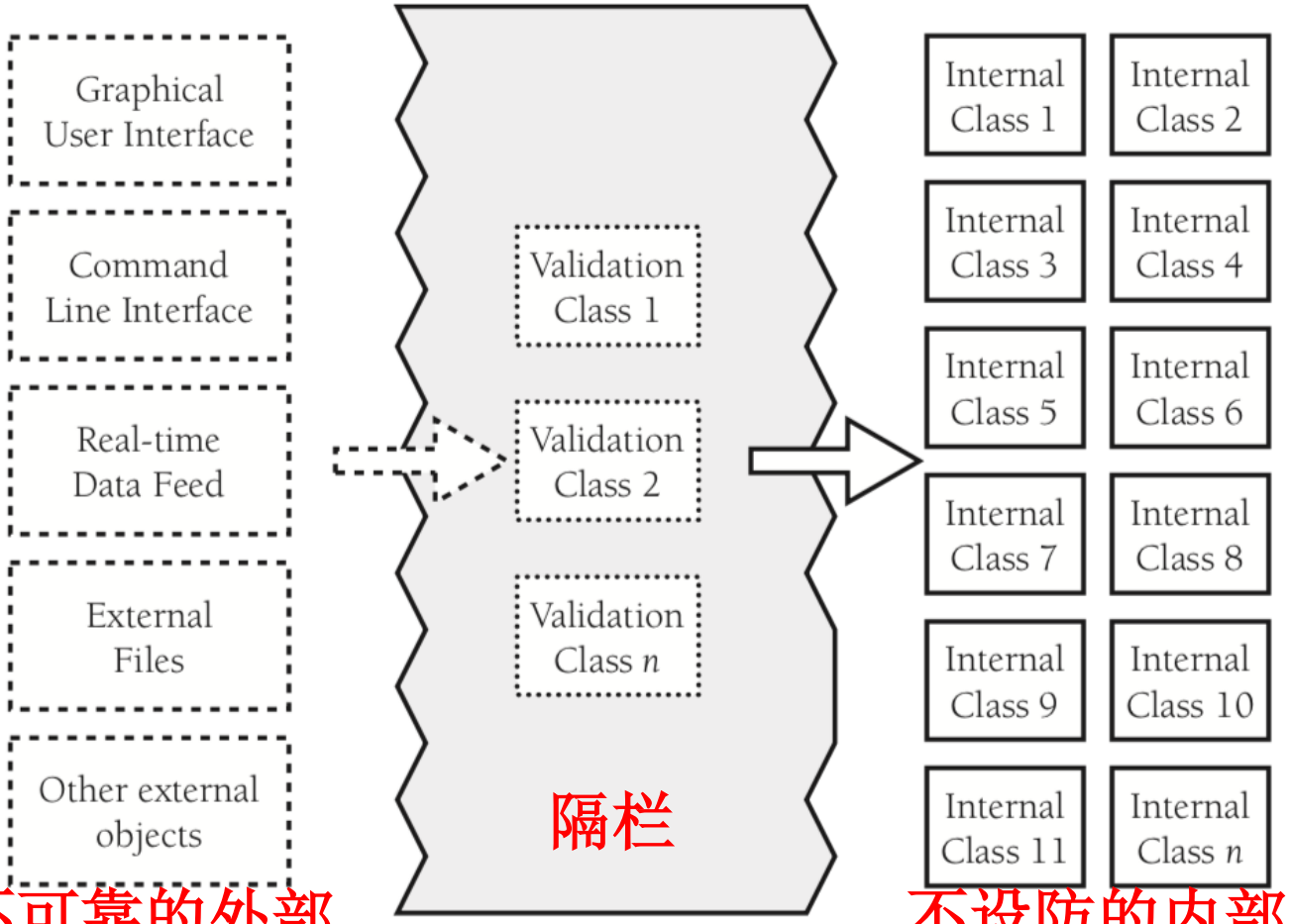
```
Try
    ...
Catch exceptionObject As Exception
    ReportException( CLASS_NAME, exceptionObject )
End Try
```

异常的使用方式-6

- 将项目中对异常的使用标准化
  - ✓ 规定可以抛出哪些类型的异常（例如，只抛出从Exception基类派生出的对象）
  - ✓ 为当前项目创建一个专门的异常类作为所有可能抛出的异常类的基类，从而便于异常处理操作的集中处理
  - ✓ 规定在哪些情况下允许使用catch语句进行异常的局部处理
  - ✓ 规定在哪些情况下允许抛出不在局部处理的异常
  - ✓ 确定是否要使用集中的异常报告机制
  - ✓ 规定是否允许在构造/析构函数中使用异常

隔栏（barricade）机制

让隔栏部分处理“不干净的”数据，而让内部代码处理“干净的”数据，这样大部分代码无需再担负检查错误数据的职责



隔栏机制的应用

- 类层次的一种应用方式：公共方法负责进行数据检查和清理（隔栏），私有方法假设数据都是安全的
- 在更高层次上规定各个部分的职责：哪些部分在安全区外、哪些负责数据清理、哪些部分在安全区内
- 隔栏与断言的关系
  - ✓ 隔栏外部的程序应该使用错误处理技术
  - ✓ 隔栏内部的程序应该使用断言技术
- 确定隔栏内外的代码之分是一种设计决策

在产品发布中保留多少防御式代码

- 保留那些检查重要错误的代码
- 去掉处理细微错误的代码（可以记录在日志中）
- 去掉可能导致程序硬性崩溃的代码（不能导致用户数据丢失）
- 保留可以让程序稳妥地崩溃的代码
- 为技术支持人员记录必要的错误信息
- 确保留在代码中的错误消息是友好的（例如报告“内部错误”并提供自动报告机制）

阅读建议

- 《构建之法》4.1-4.2
- 《代码大全》第7、8、11、19章

快速阅读后整理问题
在QQ群中提出并讨论

CS2001
软件工程

End

3. 个人开发技能
—代码质量