



西安电子科技大学
XIDIAN UNIVERSITY

软件实现基础





内容

1. 软件实现概述

✓ 软件实现的任务、过程与原则

2. 软件实现语言

✓ 编程语言的类别和选择

3. 高质量编码

✓ 编码的原则和要求

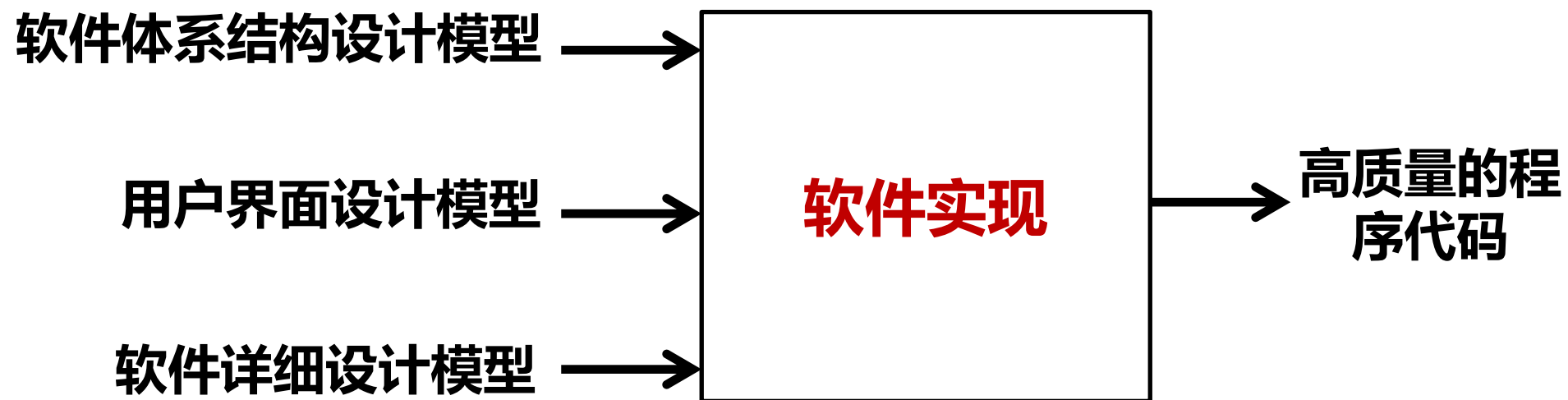




1.1 何为软件实现

- 根据**软件设计模型**，编写出目标软件系统的**程序代码**，并对代码进行必要的**测试**，以发现和纠正代码存在中的**缺陷**，并将可运行的目标代码部署到目标计算机上运行
- 软件实现不仅要编写出程序代码，还要确保代码的质量，因此软件实现涉及多方面的开发工作，如**编码、测试、调试等**

软件实现的任务





软件实现兼具创作和生产

□生产性活动

- ✓ 需要根据**软件设计规格说明书和软件设计模型**，生产出与之相符的软件制品，即程序代码
- ✓ 遵循设计文档和模型来编写程序，而且还要求程序员**遵循编码原则和风格**来编写出高质量的程序代码，并通过单元测试、集成测试、确认测试等一系列的软件测试活动来保证代码质量

□创作性活动

- ✓ 发挥软件开发工程师的**智慧和主观能动性**，创作出目标软件系统的程序代码。这一过程高度依赖于程序员的**编程经验、程序设计技能和素养**，以及软件测试工程师的**软件测试水平**



1.2 软件实现需考虑多方面的因素

□与多类不同的人员相关

- ✓包括程序员、软件测试工程师等

□程序员要考虑的因素

- ✓不仅要对照设计来编写代码，还需要通过遵循编码规范、程序设计原则等来提高代码的质量

□软件测试工程师需要考虑的因素

- ✓针对代码开展测试，不仅要发现代码中存在的功能性缺陷，如代码功能实现不正确，还要发现代码中存在的非功能性缺陷



软件实现与软件设计之间的关系

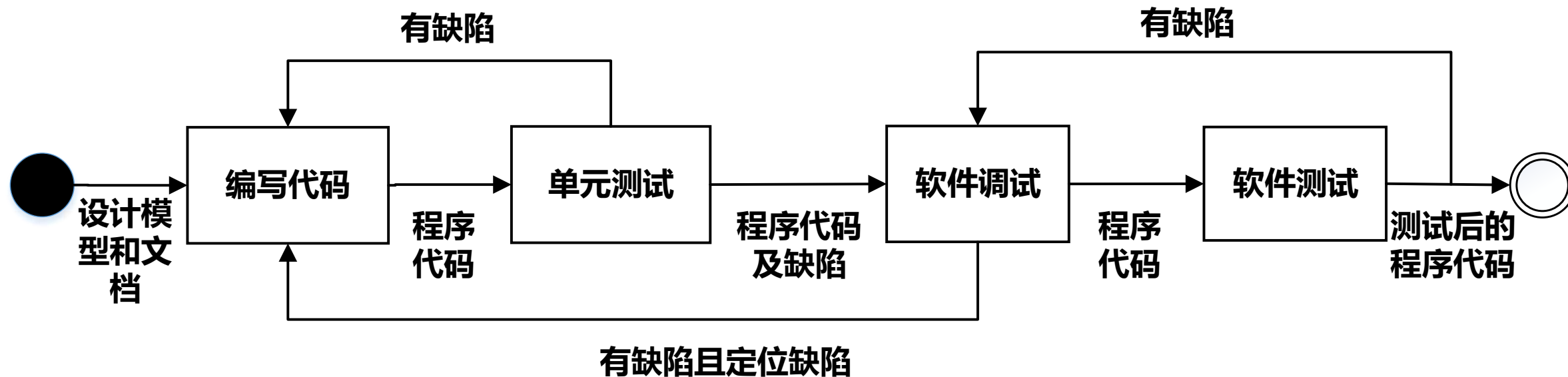
□基于软件设计来开展软件实现

- ✓照软件设计模型和文档来进行编码

□根据实现中发现的问题来纠正和完善软件设计

- ✓设计不够详细，程序员需要进行进一步的软件设计和程序设计，才能编写出程序代码
- ✓设计考虑不周全，软件设计时没有认真考虑编码实现的具体情况（如程序设计语言和目标运行环境的选择），导致有些软件设计不能通过程序设计语言加以实现
- ✓程序员需要根据编码阶段发现的问题回溯到软件设计模型和文档，对其进行纠正和完善

1.3 软件实现的过程



软件实现包含编码、测试、调试等一系列的开发活动

编码+单元测试+调试

□这三项工作均由程序员负责完成

□编码

- ✓基于软件设计模型和文档，采用选定的程序设计语言，编写出目标软件系统的程序代码

□单元测试

- ✓对自己编写的各个基本模块进行单元测试，以发现模块单元中存在的缺陷和问题

□调试

- ✓发现产生缺陷原因，定位缺陷位置，进而对代码缺陷进行修复



软件测试

□包括多项的软件测试工作

✓集成测试、确认测试、系统测试等

□这项工作由软件测试工程师来完成



1.4 软件实现要遵循的原则

□基于设计来编码

- ✓切忌抛开设计文档和模型，“拍脑袋”写程序

□质量保证贯穿全过程

- ✓要有非常强的“质量”意识
- ✓既要重视外部质量，也要重视内部质量



内容

1. 软件实现概述

✓ 软件实现的任务、过程与原则

2. 软件实现语言

✓ 编程语言的类别和选择

3. 高质量编码

✓ 编码的原则和要求





程序设计语言提供的支持

- 提供了**语法、语义和语用**三方面的要素
- 支持程序员来编写程序代码
- 人们提出了二千多种的程序设计语言，不同的语言适合于不同的应用开发



2.1 程序设计语言的类别 (1/3)

□ 机器语言

- ✓ 由 “0”、 “1” 所组成的机器指令
- ✓ 极为繁琐、费时费力的工作；软件开发效率非常低，而且程序代码可读性非常差，极易出错，不易于维护、移植性差；但程序代码的执行效率会非常高

□ 汇编语言

- ✓ 一种**低级语言**，用助记符代替机器指令的操作码，用地址符号或标号代替指令或操作数的地址
- ✓ 较为低级和复杂，程序可读性差，代码编写的效率低，对代码进行维护非常困难，程序调试也不容易，代码兼容性差
- ✓ 程序代码占用存储空间少、运行速度快、执行效率高



程序设计语言的类别 (2/3)

□结构化程序设计语言

- ✓以**过程或函数**作为基本的编程单元，采用三类控制结构（顺序、条件和循环）来刻画模块的处理过程和流程
- ✓属于**高级程序设计语言**，程序可读性、可理解性、可维护性等有了明显的提升；配套CASE工具较为完善，有结构化程序设计方法学的指导
- ✓不足：以过程和函数作为基本模块，模块的粒度小，可重用性差；程序代码抽象层次低，无法对问题域及其求解进行自然抽象
- ✓如C、Fortran、Pascal等



程序设计语言的类别 (3/3)

□面向对象程序设计语言

- ✓以类作为基本的模块单元，借助于**面向对象的一组概念和机制**来进行程序设计，
- ✓有系统的**方法学指导**，建立起可直观反映问题域、模块粒度更大、可重用性更好的程序代码，已经成为计算机领域的主流编程语言
- ✓如Java、C++等

□描述性程序设计语言

- ✓描述程序需要**解决什么样的问题**，无需在程序中显式地定义如何来解决问题
- ✓如Prolog、Lisp、ML等

2.2 程序设计语言的表达能力

程序设计方法及语言抽象层次越来越高，越来越贴近应用本身，基本模块单元越来越大，更好支持软件复用

编程语言的类别	平均代码量	编程语言	平均代码量
机器语言	320	C	128
汇编语言	107	Fortran	107
高级语言	80	C++ / Java	53



2.3 程序设计语言的选择 (1/3)

□ 软件的应用领域

- ✓ 不同应用领域的软件通常会选择不同的程序设计语言来加以实现
- ✓ 科学和工程计算领域选用Fortran、C等程序设计语言，数据库应用软件开发会选用Delphi、Visual Basic、SQL等程序设计语言，机器人等嵌入式应用选用C、C++、Python等程序设计语言，互联网应用开发选用Java、ASP等程序设计语言

□ 与遗留软件系统的交互

- ✓ 考虑待开发软件系统是否需要与遗留软件系统存在交互。如果有该方面的实际需要，那么程序员需要解决二个系统之间的互操作问题



程序设计语言的选择 (2/3)

□ 软件的**特殊功能及需求**

- ✓ 是否需要与底层的硬件系统进行交互，如果需要，可以考虑采用诸如C、汇编语言
- ✓ 是否需要丰富的软件库来支持功能的实现，如果需要，可以考虑具有丰富软件库的编程语言，如Python、Java等
- ✓ 是否需要相关的知识进行表示和推理，如果需要，可以考虑选用描述性的程序设计语言，如Prolog、Lisp等

程序设计语言的选择 (3/3)

□软件的目标平台

- ✓如果目标软件系统需要运行在特定的软件开发框架、软件中间件、基础设施之上，那么程序员还需要考虑目标平台对程序设计语言的支持，并依此来选定所需的编程语言
- ✓如果目标软件系统需要部署在J2EE架构之上，那么就需要选择Java编程语言；如果需要借助于ROS来开发机器人软件，那么建议选择C、C++和Python等编程语言

□程序员的编程经验

- ✓应该选择对于自己而言较为熟悉的语言，尽量避免选择没有使用过的程序设计语言



2.4 流行的程序设计语言

□2021年7月份程序设计语言的使用排行榜

排名	语言名称	使用占比
1	C	11.62%
2	Java	11.17%
3	Python	10.95%
4	C++	8.01%
5	C#	4.83%
6	Visual Basic	4.5%
7	Java Script	2.71%
8	PHP	2.58%
9	汇编语言	2.4%
10	SQL	1.53%



内容

1. 软件实现概述

✓ 软件实现的任务、过程与原则

2. 软件实现语言

✓ 编程语言的类别和选择

3. 高质量编码

✓ 编写代码的原则和要求





3.1 编写代码的原则 (1/3)

□易读，一看就懂

- ✓能够理解代码的语义和内涵，了解相关语句和代码的实现意图，方便修改和维护代码
- ✓采用缩进的方法来组织代码的显示，用括号来表示不同语句的优先级，对关键语句、语句块、方法等要加以注释

□易改，便于维护

- ✓或者在适当的位置增加新的代码以完善代码功能，或者对某些代码进行修改以便纠正代码中的缺陷和错误
- ✓对将来可能需要进行修改和维护的代码（包括常元、变量、方法等）进行单独的抽象、参数化和封装，以便将来对其修改时不会影响其他部分的代码

编写代码的原则 (2/3)

□降低代码的复杂度

- ✓ 将一个类代码组织为一个文件，并用统一的命名规则来命名文件
- ✓ 在代码中适当的增加注释以加强对代码的理解，不用“goto”语句，慎用嵌套或者减少嵌套的层数，尽量选用简单的实现算法

□尽可能地开展软件重用和编写可重用的程序代码

- ✓ 尽可能地重用已有的软件制品，如函数库、类库、软构件、开源软件、甚至代码片段等等
- ✓ 在编码时要考虑所编写代码的可重用性，使得所编写的代码能为他人或者在其它软件系统开发中被再次使用

编写代码的原则 (3/3)

□要有处理异常和提高代码的容错性

- ✓编写必要的异常定义和处理代码，使得程序能够对异常情况进行必要的处理，防止由于异常而导致的程序终止或崩溃
- ✓编写程序代码以支持故障检测、恢复和修复，确保程序在出现严重错误时仍然能够正常运行，或者当崩溃时能尽快恢复执行

□代码要与模型和文档相一致

- ✓程序员在编写代码的同时要同步修改和完善相应的软件设计模型和文档，确保代码、模型和文档三者之间保持一致



3.2 遵循编码风格 (1/4)

□ 格式化代码的布局，尽可能使其清晰、明了

- ✓ 充分利用水平和垂直两个方向的编程空间来组织程序代码，便于读者阅读代码
- ✓ 适当地插入括号 “{ }”，使语句的层次性、表达式运算次序等更为清晰直观
- ✓ 有效地使用空格符，以显式地区别程序代码的不同部分（如程序与其注释）

```
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);
    }
}
```



遵循编码风格 (2/4)

□ 尽可能提供简洁的代码，不要人为地增加代码的复杂度

- ✓ 使用简单的数据结构，避免使用难以理解和难以维护的数据结构（如多维数组、指针等）
- ✓ 采用简单而非复杂的实现算法
- ✓ 简化程序中的算术和逻辑表达式
- ✓ 不要引入不必要的变元和动作
- ✓ 防止变量名重载
- ✓ 避免模块的冗余和重复

```
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);
    }
}
```



遵循编码风格 (3/4)

□对代码辅之以适当的文档，以加强程序的理解

- ✓有效、必要、简洁的代码注释
- ✓代码注释的可理解性、准确性和无二义性
- ✓确保代码与设计模型和文档的一致性

```
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);
    }
}
```




遵循编码风格 (4/4)

□加强程序代码的结构化组织，提高代码的可读性

- ✓按一定的次序来说明数据
- ✓按字母顺序说明对象名
- ✓避免使用嵌套循环结构和嵌套分支结构
- ✓使用统一的缩进规则
- ✓确保每个模块内部的代码单入口、单出口

```
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;

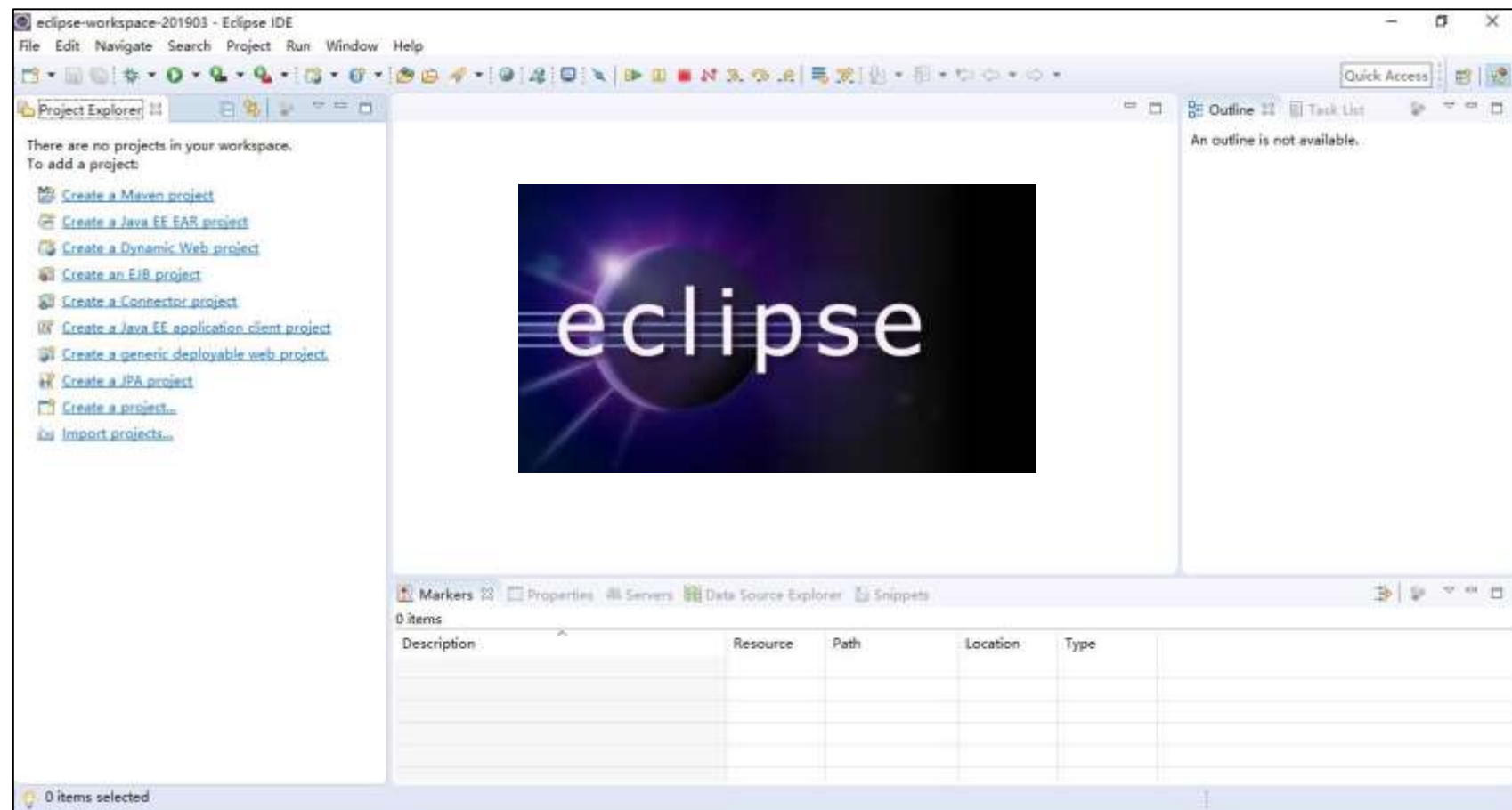
    private static final int URI_SEARCH = 5;
    private static final int URI_SEARCH_SUGGEST = 6;

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



3.3 支持软件实现的CASE工具

- 编辑器
- 编译器
- 调试器
- 测试工具
- 集成开发环境



Eclipse集合开发环境

3.4 软件实现的输出

- 源程序代码
- 部署在不同计算节点上的可执行程序代码
- 软件测试报告等



小结

□软件实现

- ✓软件实现包括编码、测试、调试、部署等一系列的活动
- ✓基于软件设计模型，编写出目标软件系统的程序代码，并对代码进行必要的测试，以发现和纠正代码存在中的缺陷，并将目标代码部署到计算机上运行

□编程语言的选择

- ✓要根据软件所属的应用领域、与遗留软件系统的交互、程序员的经验等多个方面，考虑选择什么样的程序设计语言来进行编程

□程序员遵循编码的原则和规范来编写出高质量的程序代码



西安电子科技大学
XIDIAN UNIVERSITY

编写代码





内容

1. 编写代码

- ✓ 任务、过程和方法
- ✓ 代码片段的重用

2. 软件缺陷和调试

- ✓ 软件缺陷、错误和失效
- ✓ 代码缺陷的应对方法及调试

3. 解决编程和调试问题

- ✓ 开源技术问答社区
- ✓ 群智知识的利用





1.1 编写代码的任务

□根据软件设计信息，借助于程序设计语言，编写出目标软件系统的**源程序代码**，开展**程序单元测试**、**代码审查**等质量保证工作

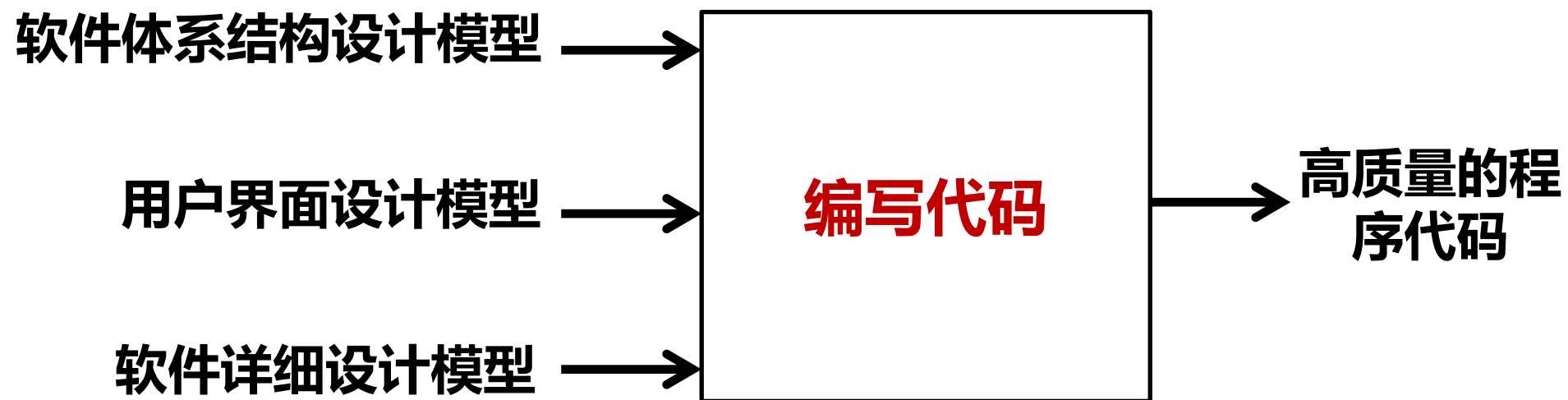
✓编写代码既是一个生成代码的过程，也是对生成的代码进行质量保证的过程

□兼具**软件创作**和**软件生产**的过程

✓自由地开展代码创作，编写出满足要求的程序代码，发挥其创新性和主观能动性，创作出算法精巧、运行高效的代码

✓按照软件质量保证的规范和要求，生产出高质量的代码。程序员需要约束其编程行为，防止随意性、自由性的编程活动，确保其编程活动及其所产生的程序代码满足工程化开发的要求

编写代码的任务



- 编写类代码
- 编写用户界面代码
- 编写数据设计代码



1.1.1 编写类代码

- ① 编写实现类的代码
- ② 编写实现类方法的代码
- ③ 编写实现类间关联的代码
- ④ 编写实现设计类间聚合和组合关系的代码
- ⑤ 编写实现接口关系的代码
- ⑥ 编写实现继承关系的程序代码
- ⑦ 编写实现包的代码



(1) 编写实现类的代码

□ **设计模型**（如设计类图）详细描述了软件系统中类的详细设计信息，包括可见性、类名、属性、方法等等

□ 程序员需要将这些设计信息直接转换为用程序设计语言表示的**实现结构和代码**

```
public class User {  
    private String account; //用户的账号  
    private String password; //用户的密码  
    private String name; //用户的名字  
    private String mobile; //用户的移动手机号  
    private int type; //用户的类别  
  
    public void User(String account, String password); //构造函数  
    public void User(String account, String password,   
        String name, String mobile, int type);  
    public String getName(); //获取用户的名字  
    public String getAccount(); //获取用户的账号  
    public int getType(); //获取用户的类别  
    public String getMobile(); //获取用户的手机号  
    public void setPsw(String userPsw); //设置用户的密码  
}
```




(2) 编写实现类方法的代码

□ 基于类方法的设计描述 (UML的活动图表示), 程序员可以依此为依据来编写类方法的实现代码

```
public int login(String account, String password) {  
    final int ERROR_ACCOUNT_EMPTY = 1; //表示账号为空的错误代码  
    final int ERROR_PASSWORD_EMPTY = 2; //表示密码为空的错误代码  
    final int ERROR_INVALID_USER = 3; //表示用户非法的错误代码  
    final int LOGIN_SUCCESS = 0; //表示用户合法的代码  
    int result;  
  
    If (account.getLength() == 0) { //检查 account 是否为空串  
        result = ERROR_ACCOUNT_EMPTY; //表示账号为空  
    } else if (password.getLength() == 0) { //检查 password 是否为空串  
        result = ERROR_PASSWORD_EMPTY; //表示密码为空  
    } else {  
        //向 UserLibrary 对象发消息以验证用户的身份是否合法  
        boolean validUser = userLib.isUserValid(account, password);  
        If (validUser) {  
            result = LOGIN_SUCCESS;  
        } else {  
            result = ERROR_INVALID_USER;  
        }  
    }  
    return result;  
}
```

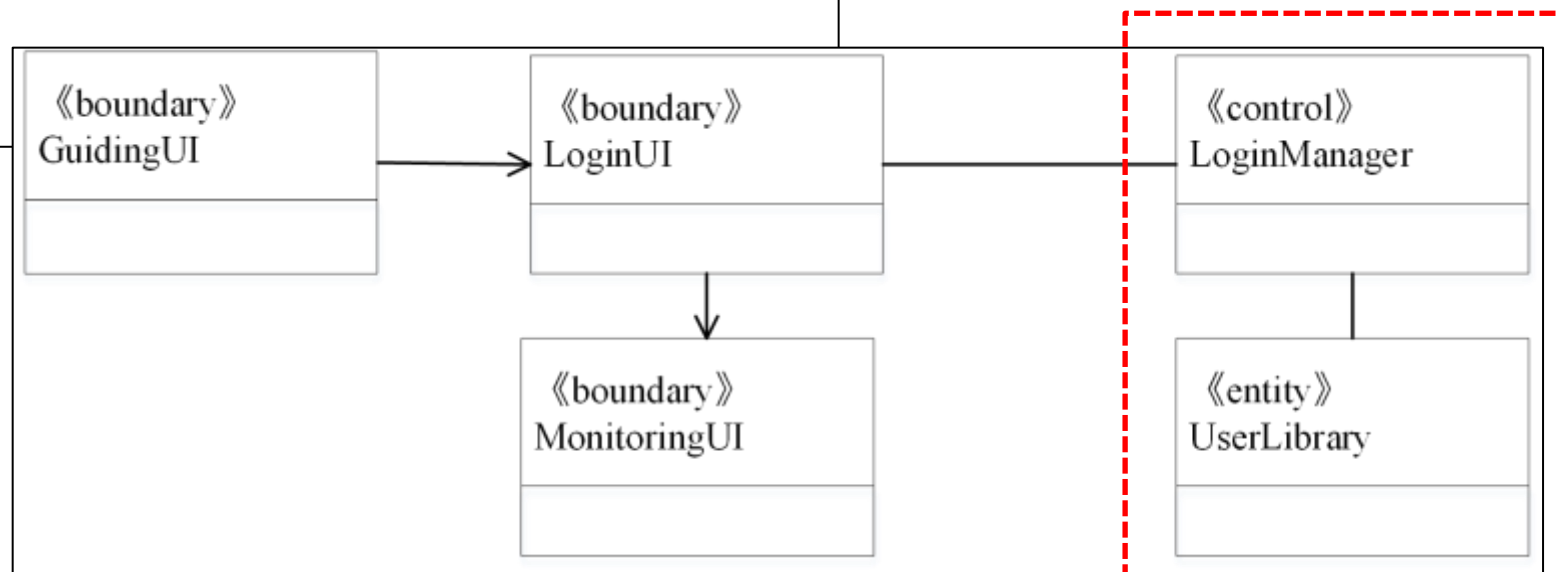



(3) 编写实现类间关联的代码

□ 将类间关联关系的**语义信息**具体落实到相应类的**程序代码**中，即综合考虑关联关系的**方向性**、**多重性**、**角色名**和**约束特性**等信息来编写相关的类程序代码

```
public class LoginManager {  
    private UserLibrary userLib; // UserLibrary 的对象  
    ...  
}
```

如果A与B存在单向关联，意味着A中存在一项属性记录了B的对象或者指针和应用





(4) 编写实现设计类间聚合和组合关系的代码

- 聚合和组合关系是一种特殊的关联关系，可以采用类似于实现**关联关系的方法**来编写实现聚合和组合关系的代码
- 根据多重性来设计相应类属性的数据结构



(5) 编写实现接口关系的代码

- 类设计模型可能包含有表征类与接口之间实现关系的语义信息
- 诸多面向对象程序设计语言（如Java、C++等）提供了专门针对接口实现的语言机制，因而可以直接将接口设计信息转换为相应的程序代码
 - ✓ 如 “Implement” 机制



(6) 编写实现继承关系的程序代码

□面向对象程序设计语言（如Java、C++）提供了继承机制以及相应的语言设施

- ✓Java支持单重继承，C++支持多重继承
- ✓如“extends”机制

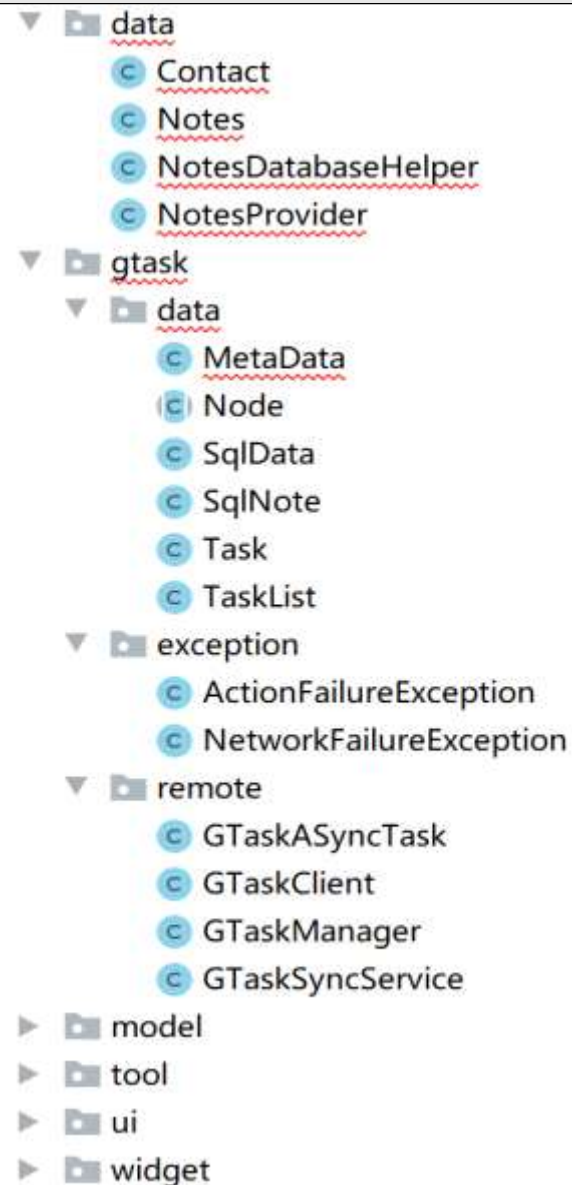
□将设计模型中的类间继承关系用程序设计语言提供的语言机制来表示

```
public class LoginUI extends Activity {  
    //成员方法说明  
    public void login();  
    public void cancel();  
    public boolean isInputAccountValid();  
    public boolean isInputPswValid();  
}
```



(7) 编写实现包的代码

- 用包 (package) 来**组织和管理**软件系统中的类
- 包是对软件系统中模块的**逻辑划分**，也可以将包视为是一种**子系统**
- 面向对象程序设计语言（如Java）提供了对包进行编程的语言机制，每个包对应于代码目录结构中的**某个目录**





1.1.2 编写用户界面代码

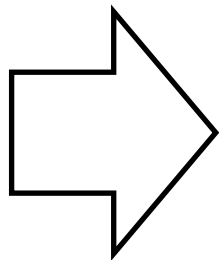
□ 用户界面设计模型

- ✓ 描述了构成用户界面的各个界面设计元素（包括静态元素、动态元素、用户输入元素、用户命令元素等）
- ✓ 用户界面之间的跳转关系

□ 编码实现

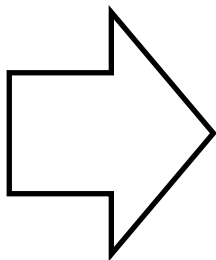
- ✓ 编写界面类属性的代码以定义界面设计元素
- ✓ 编写界面类的方法以对界面操作或者对界面事件进行响应处理

示例：“LoginUI” 的编码实现



```
public class LoginUI extends Activity {  
    private EditText mAccount;  
    private EditText mPsw;  
    private Button mCancelButton;  
    private Button mLoginButton;  
    private UserLibrary mUserLibrary;  
  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
    }  
    //用户登录方法  
    public void login() {  
        ...  
    }  
    //取消登录方法  
    public void cancel() {  
        ...  
    }  
}
```

示例：“LoginUI” 的编码实现



```
OnClickListener mListener = new OnClickListener() {  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.login_btn_cancel: //取消  
                onPause();  
                break;  
            case R.id.login_btn_login: //确认  
                login();  
                break;  
        }  
    }  
};
```




1.1.3 编写数据设计代码

□数据设计

- ✓定义了软件系统中需要持久保存数据及其组织（如数据库的表、字段）和存储（如数据库中的记录）方式
- ✓设计了相应的类及其方法来读取、保存、更新和查询持久数据

□编码实现

- ✓创建相应的数据库关系表格及其内部的各个字段选项等，确保它们满足设计的要求和约束
- ✓编写相应的程序代码来操作数据库，如增加、删除、更改、查询数据记录等

示例: “T_User” 表的创建

```
private static final String TABLE_NAME = "T_User";  
public static final String USER_ACCOUNT = "user_account";  
public static final String USER_NAME = "user_name";  
public static final String USER_PSW = "user_password";  
public static final String USER_MOBILE = "user_mobile";  
public static final int USER_TYPE = "user_type";  
  
//创建数据库表 “T_User” 的 SQL 语句  
String DB_CREATE = "CREATE TABLE " + TABLE_NAME + " (" +  
    + USER_ACCOUNT + " varchar primary key," + USER_NAME + " varchar," +  
    + USER_PSW + " varchar," + USER_MOBILE + " varchar," +  
    + USER_TYPE + " integer" + ");";  
  
db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME + ";"); //执行 SQL 语句  
db.execSQL(DB_CREATE);
```

示例：连接和关闭数据库的代码

```
// 打开数据库↵  
public void openDataBase() throws SQLException {↵  
    mDatabaseHelper = new DataBaseManagementHelper(mContext);↵  
    mSQLiteDatabase = mDatabaseHelper.getWritableDatabase();↵  
}↵  
// 关闭数据库↵  
public void closeDataBase() throws SQLException {↵  
    mDatabaseHelper.close();↵  
}↵
```



示例：操作数据库的程序代码

//向数据库表中插入用户的数据↵

```
public boolean insertUser(User user) {↵
```

```
    String UserAccount = user.getUserAccount();↵
```

```
    String UserName=user.getUserName();↵
```

```
    String UserPsw=user.getUserPsw();↵
```

```
    String UserMobile=user.getUserMobile();↵
```

```
    int UserType=user.getUserType();↵
```

```
    ContentValues values = new ContentValues();↵
```

```
    values.put(USER_NAME, UserName);↵
```

```
    values.put(USER_PSW, UserPsw);↵
```

```
    values.put(USER_MOBILE, UserMobile);↵
```

```
    values.put(USER_TYPE, UserType);↵
```

```
    return mSQLiteDatabase.insert(TABLE_NAME, ID + "=" + UserAccount, values);
```

```
}↵
```

示例：判断用户账号和密码合法性的代码

```
// 基于账号和密码来判断用户身份的合法性↵  
public boolean verifyUserValidity(String account,String psw){↵  
    boolean result= FALSE;↵  
    Cursor mCursor=mSQLiteDatabase.query(TABLE_NAME, null,  
    USER_ID+"="+account+" and "+USER_PSW+"="+psw, null, null, ↵  
    null, null);↵  
    if(mCursor!=null){↵  
        result=TRUE;↵  
        mCursor.close();↵  
    }↵  
    return result;↵  
}↵
```

1.2 代码片段的重用

□ 何为代码片段

- ✓ 对应于类代码中所包含的一组语句序列
- ✓ 实现了类中的一个具体、细粒度的功能

□ 代码片段示例

- ✓ 与远端数据库服务器建立连接
- ✓ 向远端的Socket程序发送一段数据

```
mDatabaseHelper = new DataBaseManagementHelper(mContext);  
mSQLiteDatabase = mDatabaseHelper.getWritableDatabase();
```

开源社区中的代码片段

- 开源技术问答社区（如Stack Overflow、CSDN）中，大量的程序员在其中分享了许多形式多样、极有价值的代码片段
- 通常这些代码片段都经过实践检验，因而表现出较高的代码质量
- 在编写代码的工程中，程序员可以针对其代码编写要求，到开源技术问答社区中寻找相关的代码片段，然后通过
对代码片段的理解，选定和重用所需的代码片段，进而完成相应的编程任务



示例：重用开源社区中的代码片段

□完成与MySQL数据库服务器连接的代码片段

```
12         try{
13             Class.forName("com.mysql.jdbc.Driver");
14         } catch (ClassNotFoundException e){
15             System.out.println("未能成功加载驱动程序,请检查是否导入驱动程序!");
16             e.printStackTrace();
17         }
18         Connection conn = null;
19         try{
20             conn = DriverManager.getConnection(URL, NAME, PASSWORD);
21             System.out.println("获取数据库链接成功");
22         } catch (SQLException e){
23             System.out.println("获取数据库连接失败");
24             e.printStackTrace();
25         }
```




内容

1. 编写代码

- ✓ 任务、过程和方法
- ✓ 代码片段的重用

2. 软件缺陷和调试

- ✓ 软件缺陷、错误和失效
- ✓ 代码缺陷的应对方法及调试

3. 解决编程和调试问题

- ✓ 开源技术问答社区
- ✓ 群智知识的利用





2.1 何为软件缺陷

□ **软件缺陷是指软件制品中存在不正确的软件描述和实现**

- ✓ 存在缺陷的软件制品不仅包括**程序代码**，而且还包括需求和设计的**模型和文档**
- ✓ 软件缺陷**产生于软件开发全过程**，只要有人介入的地方就有可能产生软件缺陷
- ✓ **任何人**都有可能在软件开发过程中犯错误，进而引入软件缺陷
- ✓ 无论是高层的需求分析和软件架构缺陷还是底层的详细设计缺陷，它们**最终都会反映在程序代码**之中，导致程序代码存在缺陷



软件缺陷带来的问题：错误

□存在缺陷的程序代码在运行过程中会产生不正确或者不期望的**运行状态**，将这种情况称程序出现了错误

- ✓经过计算后某个变量的取值不正确
- ✓接收到的消息内容不正确
- ✓打开一个非法的文件

□引发程序报错



错误带来的问题：失效

- 运行错误的程序无法为用户提供**所需的功能和行为**，在此情况下我们称程序出现了**失效**
 - ✓ 如用户无法正常登录到系统中
 - ✓ 无法正确地分析出老人是否处于摔倒的状态等等。
- 程序**错误的根源**在于程序中存在**缺陷**，程序的错误运行必然导致软件失效
- 错误和失效是程序缺陷在程序运行时的**内部展示和外在工作表现**



2.2 软件缺陷的描述 (1/2)

□标识符

- ✓每个软件缺陷都被给予一个唯一的标识符。

□类型

- ✓说明软件缺陷的类型，如**需求缺陷、设计缺陷、代码缺陷**
- ✓代码缺陷还可以进一步区分为是逻辑缺陷、计算缺陷、判断缺陷

□严重程度

- ✓**危急程度**是指缺陷会影响软件的正常运行甚至危及用户安全
- ✓**严重程度**的缺陷是指会导致软件丧失某些重要功能，或出现错误
- ✓**一般程度**的缺陷是指缺陷会使得软件丧失某些次要的功能
- ✓**轻微程度**是指缺陷会导致软件出现小毛病，但不影响正常运行

软件缺陷的描述 (2/2)

- **症状**：软件缺陷所引发的程序错误是什么，有何运行表现
- **修复优先级**：缺陷应该被修复的优先程度，包括：非常紧迫、紧迫、一般和不紧迫等几种
- **状态**：描述缺陷处理的进展状态，如已经安排人员来处理、正在修复、修复已经完成等。
- **发现者**：谁发现了软件缺陷。
- **发现时机**：什么状况下发现的软件缺陷
- **源头**：软件缺陷源头在哪里，如软件文档的哪一个部分
- **原因**：说明导致软件缺陷的原因是什么



2.3 软件缺陷的应对方法 (1/2)

□预防缺陷

- ✓通过运用各种软件工程技术、方法和管理手段，在软件开发过程中**预防和避免软件缺陷，减少软件缺陷的数量，降低软件缺陷的严重程度**
- ✓采用结对编程、严格的过程管理、必要的技术培训、CASE工具的使用等手段，起到预防缺陷的目的

□容忍缺陷

- ✓增强软件的缺陷容忍度，借助于**软件容错机制和技术**，允许软件出现错误，但是在**出现错误时软件仍然能够正常的运行**
- ✓在高可靠软件系统的开发过程中，软件工程师通常需要提供**容错模块和代码**。显然这会增加软件开发的**复杂度和冗余度**



软件缺陷的应对方法 (2/2)

□发现缺陷

- ✓通过有效的**技术和管理手段来发现这些软件缺陷**
- ✓例如，制定和实施软件质量保证计划、开展软件文档和模型的评审、程序代码的走查、软件测试等工作。它们都可以帮助软件工程师找到潜藏在文档、模型和代码中的软件缺陷

□修复缺陷

- ✓通过一系列的手段来修复缺陷
- ✓采用程序调试等手段来**找到缺陷的原因、定位缺陷的位置，进而修改存在缺陷的程序代码**，将软件缺陷从软件制品中**移除出去**



2.4 软件缺陷的状态 (1/2)

□尚未确认 (Unconfirmed)

- ✓有人汇报了软件缺陷，但是尚未确认该软件缺陷是否真实存在

□有效 (New)

- ✓经过确认，所汇报的软件缺陷真实存在，被正式视为是新缺陷，并等待进一步处理

□无效 (Invalid)

- ✓经过确认，所汇报的软件缺陷并不存在，是一个无效的软件缺陷汇报

□重复 (Duplicate)

- ✓该软件缺陷之前已经有人汇报过，属于重复性的软件缺陷

软件缺陷的状态 (2/2)

□ 已分配 (Assigned)

- ✓ 以安排人员负责修复缺陷

□ 已修复 (Fixed)

- ✓ 缺陷已经修复

□ 信息不完整 (Incomplete)

- ✓ 缺陷的描述信息不完整，导致无法准确和清晰地理解缺陷的内容

□ 已解决 (Resolved)

- ✓ 针对该缺陷的处理已经完成

□ 已关闭 (Closed)

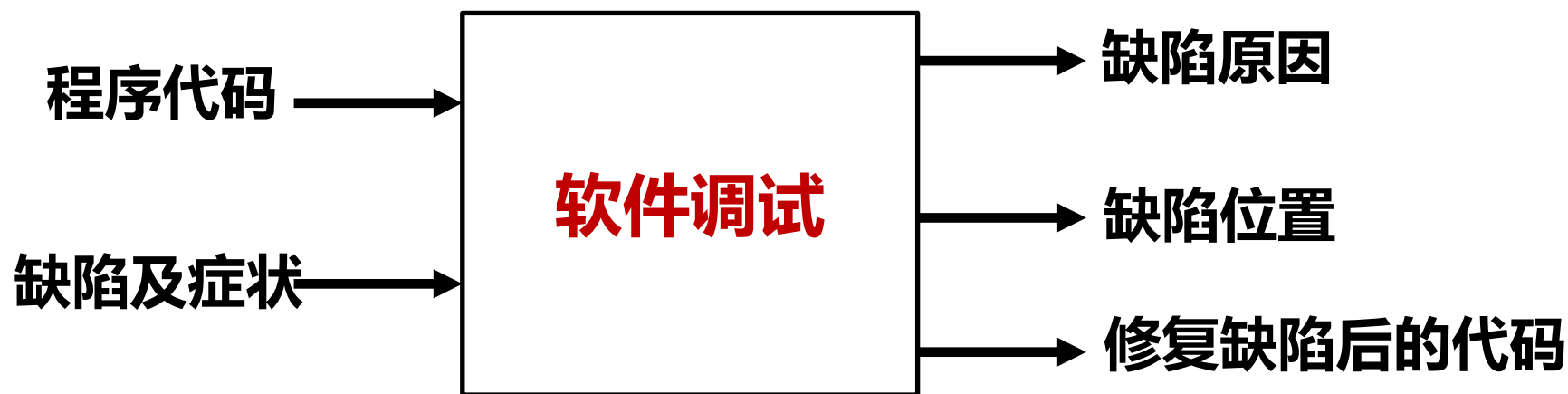
- ✓ 关闭该缺陷，后续将不再针对该缺陷采用任何措施



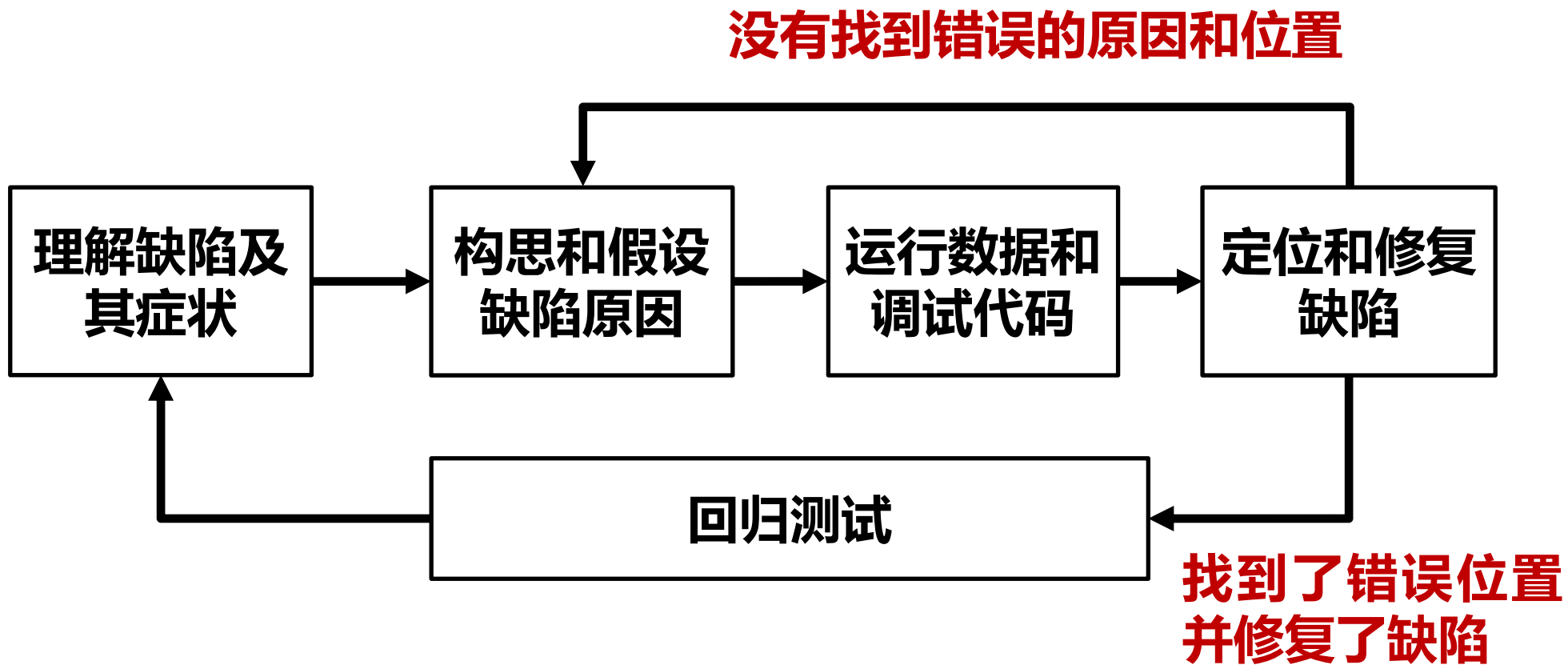
2.5 程序调试

□ 软件调试就是要基于程序代码，确定软件缺陷的原因、定位缺陷的位置，从而知道哪里错了、如何修复缺陷

- ✓ 程序员需要花费大量的时间和精力用于软件调试
- ✓ 软件调试通过运行目标软件系统的程序代码，找到缺陷的代码位置、明确软件错误的具体原因，从而开展缺陷修复工作



调试的步骤





内容

1. 编写代码

- ✓ 任务、过程和方法
- ✓ 代码片段的重用

2. 软件缺陷和调试

- ✓ 软件缺陷、错误和失效
- ✓ 代码缺陷的应对方法及调试

3. 解决编程和调试问题

- ✓ 开源技术问答社区
- ✓ 群智知识的利用





3.1 编码和调试面临的挑战

□ 编码和调试需要开放的知识

- ✓ 包括软件设计的文档和模型、程序设计语言、程序调试技术等等

□ 编码和调试要求程序员有丰富的软件编程经验、扎实的编码和调试的技能、熟练的软件开发工具使用技巧等

□ 编程和调试中仍然会遇到各种各样的棘手问题

- ✓ 明明知道程序出现了错误，但是找不到错误的原因
- ✓ 程序中的错误有时会出现，有时候不会出现
- ✓ 程序代码和他人的程序代码一模一样，但是运行结果就是不正确



3.2 解决编程和调试问题的方法

- **独立自主解决问题**，但是有时候会出现无法解决的情况，用几个小时甚至几天的时间都未能解决问题
- **寻找团队成员的帮忙**，让有经验的编程高手帮助程序员解决问题
- **借助于开源技术问答社区中的互联网大众来解决问题**

□ 常见社区

✓ CSDN、Stack Overflow

□ 交流形式

✓ 提出问题

✓ 回答问题

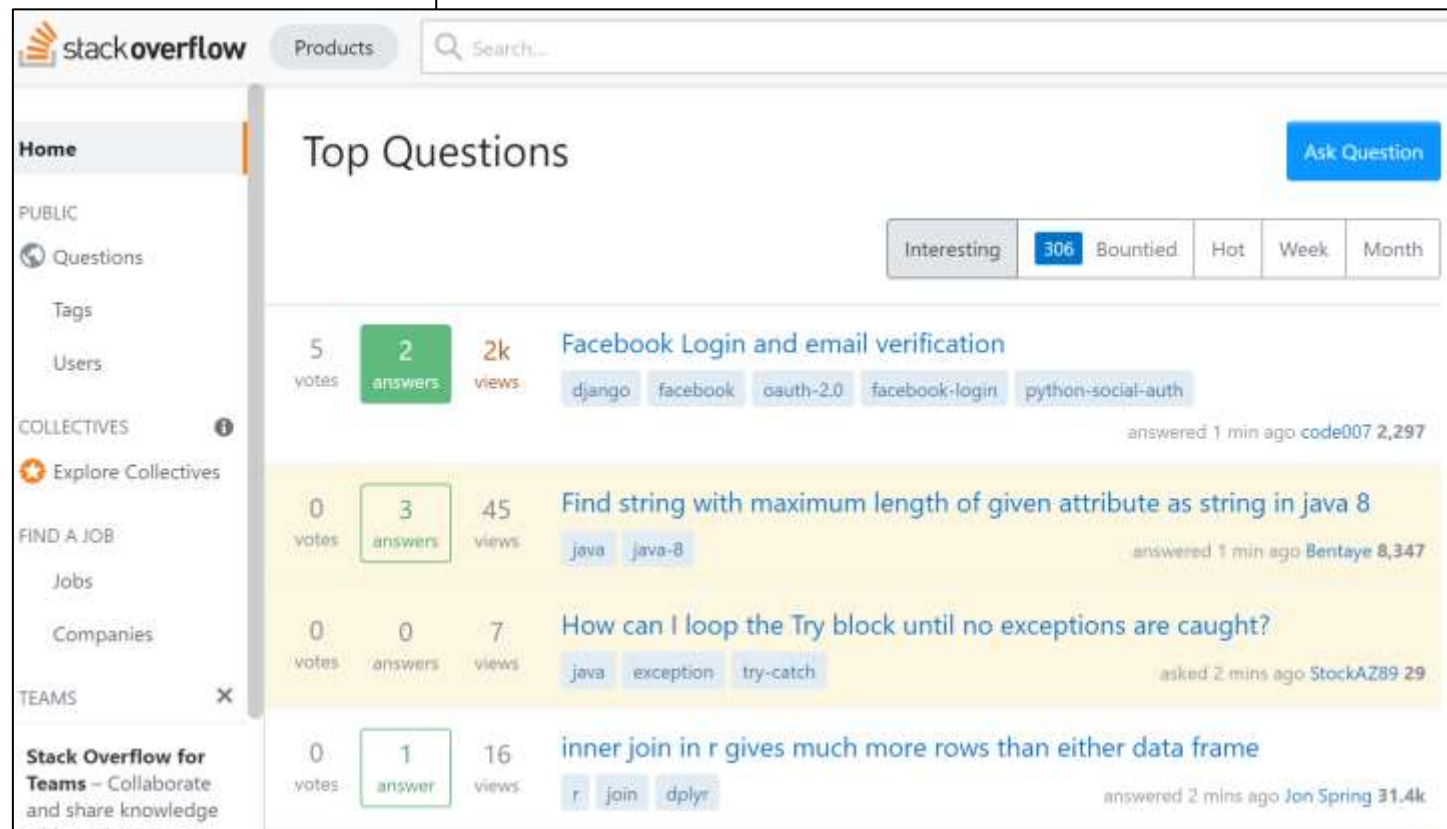
✓ 参加评论

最新 最热 悬赏 待回答 筛选

0
回答

2
浏览

matlab iris bp 看了你的，有些没懂
能加你个联系方式，问你一些问题嘛。是否可以请教一下您。有偿...
有问必答 matlab weixin_44192691 2021-09-02 15:59



The screenshot shows the Stack Overflow homepage with the 'Top Questions' section. The left sidebar contains navigation links: Home, PUBLIC (Questions, Tags, Users), COLLECTIVES (Explore Collectives), FIND A JOB (Jobs, Companies), and TEAMS. The main content area displays a list of questions with their respective statistics (votes, answers, views) and tags. The questions are:

- Facebook Login and email verification**: 5 votes, 2 answers, 2k views. Tags: django, facebook, oauth-2.0, facebook-login, python-social-auth. Answered 1 min ago by code007 (2,297).
- Find string with maximum length of given attribute as string in java 8**: 0 votes, 3 answers, 45 views. Tags: java, java-8. Answered 1 min ago by Bentaye (8,347).
- How can I loop the Try block until no exceptions are caught?**: 0 votes, 0 answers, 7 views. Tags: java, exception, try-catch. Asked 2 mins ago by StockAZ89 (29).
- inner join in r gives much more rows than either data frame**: 0 votes, 1 answer, 16 views. Tags: r, join, dplyr. Answered 2 mins ago by Jon Spring (31.4k).

站上输入用户...

2021-09-02 15:58

位置

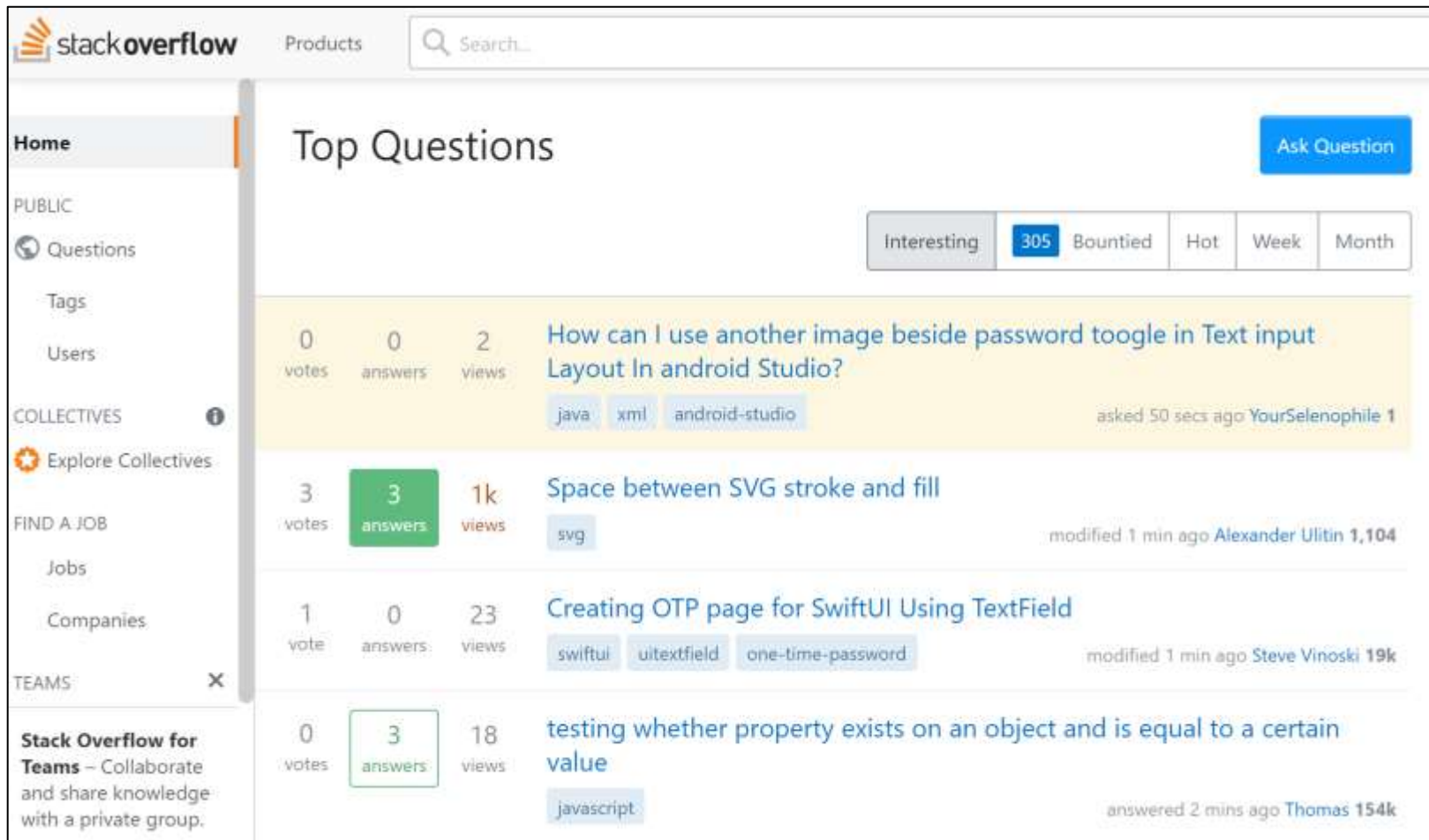
2021-09-02 15:58

7s)

2021-09-02 15:56

海量的群体和群智知识

- 用户有1500万
- 问题有2200万
- 回答有3200万
- 评论有8200万



stackoverflow Products Search...

Home

PUBLIC

- Questions
- Tags
- Users

COLLECTIVES

- Explore Collectives

FIND A JOB

- Jobs
- Companies

TEAMS

Stack Overflow for Teams – Collaborate and share knowledge with a private group.

Top Questions

Ask Question

Interesting 305 Bountied Hot Week Month

0 votes 0 answers 2 views How can I use another image beside password toggle in Text input Layout In android Studio? asked 50 secs ago YourSelenophile 1

java xml android-studio

3 votes 3 answers 1k views Space between SVG stroke and fill modified 1 min ago Alexander Ulitin 1,104

svg

1 vote 0 answers 23 views Creating OTP page for SwiftUI Using TextField modified 1 min ago Steve Vinoski 19k

swiftui uitableview one-time-password

0 votes 3 answers 18 views testing whether property exists on an object and is equal to a certain value answered 2 mins ago Thomas 154k

javascript



获取群智知识的方法

- 访问Stack Overflow、CSDN等社区
- 描述和输入自己遇到的问题
- 寻找针对该问题的有效解答



示例：查找的问题 “socket connection”

13,080 results

RelevanceNewestMore ▾

1824
votes

A: How do SO_REUSEADDR and SO_REUSEPORT differ?
A TCP/UDP **connection** is identified by a tuple of five values: {<protocol>, <src addr>, <src port>, <dest addr>, <dest port>} Any unique combination of these values identifies a **connection**. ... Setting SO_REUSEADDR on a **socket** in Windows behaves like setting SO_REUSEPORT and SO_REUSEADDR on a **socket** in BSD, with one **exception**: Prior to Windows 2003, a **socket** with SO_REUSEADDR could always been ...

linuxwindowssocketsunixportability

answered Jan 17 '13 by Mecki

17
votes

Q: Python socket connection exception
I have a **socket-connection** going on and I wanna improve the **exception** handling and Im stuck. ... Also, is it okay to exit programs using sys.exit when an unwanted **exception** occurs? Thank you ...

2
answers

pythonsocketsexceptionexception-handling

asked Aug 22 '14 by erling

302
votes

Q: No connection could be made because the target machine actively refused it?
System.Net.WebException: Unable to connect to the remote server ---> System.Net.Sockets.SocketException: No **connection** could be made because the target machine actively refused it 127.0.0.1:80 at System.Net.Sockets.Socket.DoConnect ... s4, **Socket** s6, **Socket& socket**, IPAddress& address, ConnectSocketState state, IAsyncResult asyncResult, Int32 timeout, **Exception& exception**) --- End of inner **exception** stack trace --- at System.Net.HttpWebRequest.GetRequestStream ...

30
answers

c# .net asp.net-web-api2 socketexception system.net.webexception

asked Jun 4 '10 by hsnkvk



编写代码的输出

□源程序代码

□程序单元测试报告



小结

□编写代码

- ✓任务是要产生高质量程序代码，完成单元测试、程序调试等活动
- ✓基于软件设计模型和文档来编写代码
- ✓可以通过重用技术问答社区中的代码片段来编写程序

□软件缺陷、错误和失效

- ✓缺陷是指软件制品中不正确的描述和实现，缺陷的内在表现是程序运行产生不正确或者不期望的运行状态，导致程序无法为用户提供所需的功能和行为
- ✓调试目的是要发现缺陷原因、定位缺陷位置，促进缺陷的修复

□借助于技术问答社区来解决编码和和调试中遇到的问题