试卷答题纸

（ 装 订 线 内 不 要 答 题 ）

**复旦大学计算机科学技术学院**

**2021~2022 学年第2学期期末考试答题纸**

**A卷 B卷 C卷**

**课程名称： 软件工程 课程代码： COMP130015**

**开课院系： 计算机科学与技术学院 考试形式：开卷/闭卷/课程论文/其他**

**姓名： 学号： 专业： 计算机科学与技术**

提示：请同学们秉持诚实守信宗旨，谨守考试纪律，摒弃考试作弊。学生如有违反学校考试纪律的行为，学校将按《复旦大学学生纪律处分条例》规定予以严肃处理。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 一 | 二 | 三 | 四 | 总分 |
| 得分 |  |  |  |  |  |

选择题**：**

**CDAAB DBCBA**

**——————————————————————————————————————**

一、

答： 有语句覆盖，判定覆盖，条件覆盖，判定/条件覆盖， 条件组合覆盖，

路径覆盖等覆盖测试标准以及它们对应的覆盖测试方法。它们属于白盒测试。

————————————————————————————————————

二、

答：答：

耦合的七种类型：

1) 内容耦合 ：如果一个模块直接访问另一个模块的内部数据；或者一个模块不通过正常入口转到另一模块内部；或者两个模块有一部分程序代码重迭；或者一个模块有多个入口，则两个模块之间就发生了内容耦合。

2)公共耦合 ：若一组模块都访问同一个公共数据环境，则它们之间的耦合就称为公共耦合。公共的数据环境可以是全局数据结构、共享的通信区、内存的公共覆盖区等。

3)外部耦合 ：指模块间通过软件之外的环境联结（如I/O将模块耦合到特定的设备、格式、通信协议上）时，称为外部耦合。

4) 控制耦合：如果一个模块传送给另一个模块的参数中包含了控制信息，该控制信息用于控制接收模块中的执行逻辑，则称为控制耦合。

5) 标记耦合：两个模块之间通过参数表传递一个 数据结构的一部分（如某一数据结构的子结构），就是标记耦合。

6) 数据耦合：两个模块之间仅通过参数表传递简单数据，则称为数据耦合。

7) 非直接耦合 ：如果两个模块之间没有直接关系，即它们中的任何一个都不依赖于另一个而能独 立工作，这种耦合称为非直接耦合。

内聚的七种类型：

1) 巧合内聚（偶然内聚）：将几个模块中没有明确表现出独立功能的相同程序代码段独立出来建立的模块称为巧合内聚模块。

2) 逻辑内聚 ：指完成一组逻辑相关任务的模块，调用该模块时，由传送给模块的控制型参数来确定该模块应执行哪一种功能。

3) 时间内聚：指一个模块中的所有任务必须在同一时间段内执行。例如初始化模块和终止模块。

4) 过程内聚 ：指一个模块完成多个任务，这些任务必须按指定的过程（procedural）执行。

5) 通信内聚 ：指一个模块内所有处理元素都集中 在某个数据结构的一块区域中。

6) 顺序内聚：指一个模块完成多个功能，这些功能又必须顺序执行。

7) 功能内聚 ：指一个模块中各个部分都是为完成一项具体功能而协同工作，紧密联系，不可分割的。

为什么要高内聚、低耦合？

一个模块内部各个元素之间的联系越紧密，则该模块的内聚性就越高，相对地，该模块与其他模块之间的耦合性就会减低，而功能独立性就越强。因此，功能独立性比较强的模块应是高内聚低耦合的模块。高内聚即指程序间的模块要紧密相关，形成高效的功能单元，如果模块的内聚程度不高，则无法实现高效、可读性好的代码；而低耦合指程序模块之间应该尽可能地不关联，各自实现各自的功能，避免过于复杂的数据传递，实现更好的分工。反之，若程序的耦合度很高，模块之间的连接越紧密，联系越多，耦合性就越高，而其功能独立性就越弱。此外，往往会出现修改某一模块接口，其他相关的模块都要重新测试，即时以前通过的也会受到影响，造成较大的工作量。

————————————————————————————————————

三、

答：领域工程的步骤如下 ：

1.领域分析。首先要进行领域分析，收集领域中有代表性的应用样本，分析应用中的公共部分或相似部分，抽取该领域的体系结构。

2.建立领域特定的基准体系结构模型 。在领域分析的基础上，构造该领域的基准体系 结构，这个基准体系结构应是可以裁剪和扩充的，以供该领域内的应用复用。

3.标识候选构件。在领域分析和领域基准体系结构模型的基础上标识该领域的候选 构件。

4.泛化 (generalization) 和可变性 (variab 山ty) 分析。由千候选构件可能来自某个特定的应用样本，因此候选构件具有一定的特殊性。为使之能被广泛复用，应将其泛化，提高其 通用性。同时应寻找候选构件在不同应用中可能修改的部分一—变化点 (variation point), 通过设置参数、继承或其他手段，使可变部分局部化。

5.构件重构。在泛化和可变性分析的基础上，对构件进行重构，使它成为可复用构件。

6.构件的测试。对重构的可复用的构件要严格测试，以提高其可 靠性 。 所使用的测试用例可跟随可复用构件一起被复用。

7.构件的包装。应根据构件库的要求，对经测试的构件进行包装，以便构件库对该构 件分类储存和检索。

8.构件入库。包装后的构件即可存入构件库。

————————————————————————————————————

四、

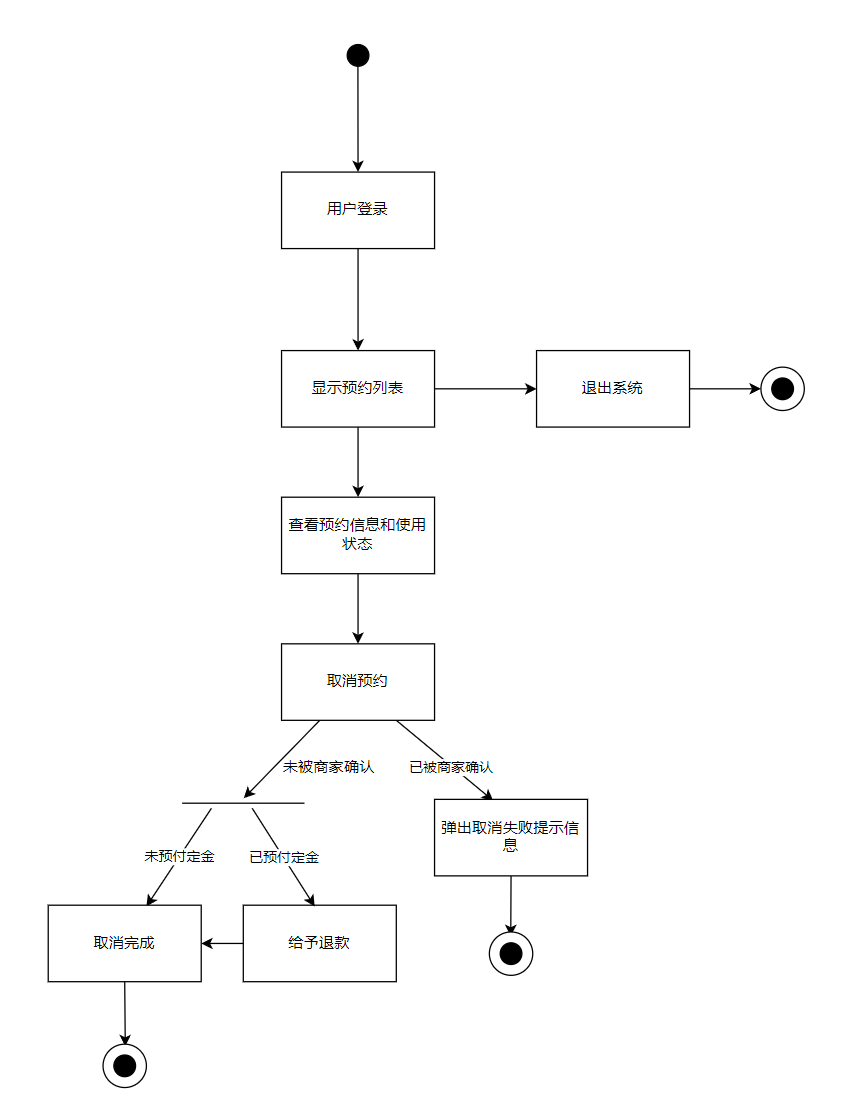
答：

什么是数据字典：数据字典是结构化分析方法导出的分析模型的核心。包括对软件使用和产生的所有数据的描述。数据字典由字典条目组成，每个条目描述DFD中的一个元素，而数据字典条目包括：数据流、文件、数据项(组成数据流和文件的数据)、加工、源或宿。数据字典是关于数据的信息的集合，也就是对数据流图中包含的所有元素的定义的集合。

数据字典和数据流图的关系：数据流图与数据字典是密不可分的，两者结合起来构成软件的逻辑模型(分析模型)，数据流图和数据字典共同构成系统的逻辑模型，没有数据字典数据流图就不严格，然⽽没有数据流图数据字典也难于发挥作⽤。只有数据流图和对数据流字典中每个元素的精确定义放在⼀起，才能共同构成系统的规格说明。

1. 设计题

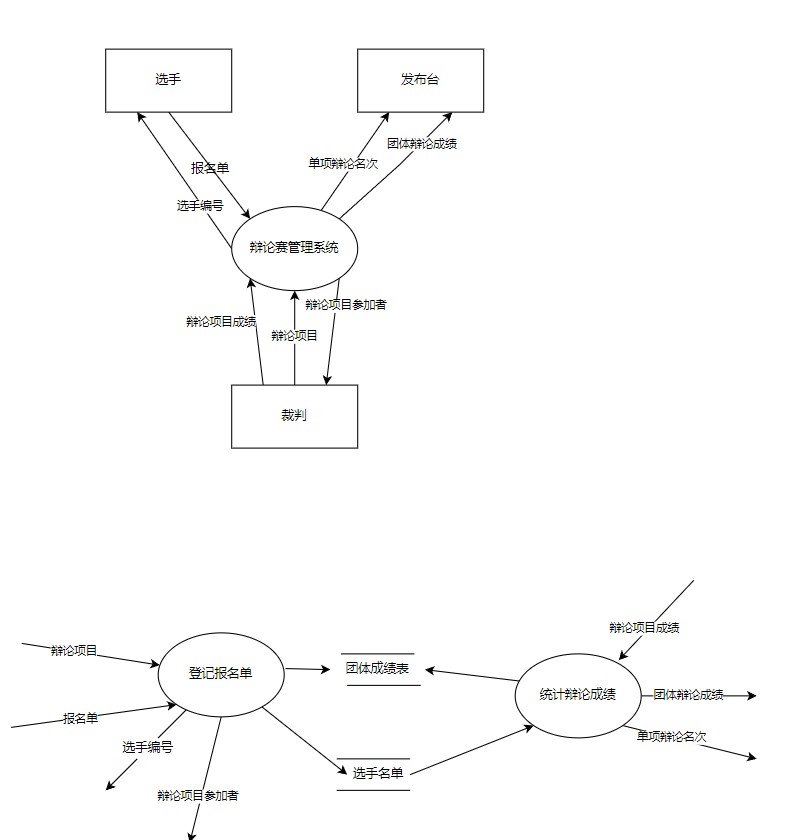
一、



—————————————————————————————————————

二、

**画图题注：由于题干和小标题（1）（2）中对于数据流的名字不同，本作答以更为细节的（1）（2）中的为准**。



1. 应用题

一、

原因（方式，重量，货物类别）：

1. 运输方式为公路

2. 运输方式为铁路

3. 货物重量小于等于2kg

4. 货物重量大于2kg小于等于20kg

5. 货物重量大于20kg

6. 货物重量小于等于20kg

7. 货物为快件

8. 货物为慢件

结果（收费）：

9. 收费为1元/kg

10. 收费为2元/kg

11. 收费为3元/kg

12. 收费为5元/kg

13. 收费为6元/kg

14. 一律收费10元

判定表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 条件 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 |  | 0 | 0 |
| 3 | 1 | 0 | 0 |  | 0 | 0 |
| 4 | 0 | 1 | 0 |  | 0 | 0 |
| 5 | 0 | 0 | 1 |  | 0 | 1 |
| 6 | 0 | 0 | 0 |  | 1 | 0 |
| 7 |  |  |  | 0 | 1 | 1 |
| 8 |  |  |  | 1 | 0 | 0 |
| 结果 | 9 | 0 | 0 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 1 |
| 12 | 0 | 0 | 1 | 0 | 0 | 0 |
| 13 | 0 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 | 0 |

—————————————————————————————————————

二、

最早最迟开始时间如下：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J |
| VE | 0 | 3 | 6 | 10 | 18 | 18 | 21 | 38 | 22 | 48 |
| VL | 0 | 24 | 14 | 10 | 39 | 18 | 41 | 38 | 36 | 48 |

关键路径：

A→D→F→H→J

—————————————————————————————————————

三、

为了方便维护，重构代码如下：

public class Create {

public void create(User sender){

sender.constructCreate();

}

public Decimal total;

private void setTotal(Decimal total){

this.total=total;

}

}

public class CreateProcessor {

public Decimal process(Create create,User sender){

create.create(sender);

return create.total;

}

}

protected void studentCreateClick(Student sender,EventArgs e){

CreateProcessor createProcessor=new CreateProcessor();

createProcessor.studentCreateClick(sender,e);

}

public abstract class User {

public void constructCreate(){}

}

public class Teacher extends User{

@Override

public void constructCreate() {

//construct create code

}

}

public class Student extends User{

@Override

public void constructCreate() {

//construct create code

}

}

public class Administrator extends User{

@Override

public void constructCreate() {

//construct create code

}

}

试题回忆：

选择：都是书上的。由于今年可以利用电子书pdf，可以全局搜索，所以找的比较方便。

简答1：软件测试中的覆盖标准和测试方法，属于黑or白盒？

简答2：耦合和内聚

简答3：领域工程步骤

简答4：什么是数据字典，数据字典与数据流图的关系

设计题1：一个餐厅的预约活动图

设计题2：结构化分析方法，改错和画图

应用题1：某个铁路运输收费不同，画表。书上有类似的题目

应用题2：最后一章的最早最迟开始时间

应用题3：代码重构

public class Create

{

public void createTeacher()

/construct create code

public void createStudent(

f

//construct create code

public decimal total get;private set;}

public class CreateProcessor

{

public decimal processTeacher(Create create)

{

create.createTeacher(;

return create.total;

public decimal processStudent(Create create)

create.createStudent();

return create.total:

//调用方法

protected void teacherCreateClick(object sender,EventArgs e)

{

CreateProcessor createProcessor new CreateProcessor();

createProcessor .processTeacher(new Create()):

}

protected void studentCreateClick(object sender,EventArgs e)

{

CreateProcessor createProcessor new CreateProcessor();

createProcessor .processstudent(new Create()):

}

上面的伪代码实现了创建Teacher和Student的主要类和主要实现方法及调用事件。

大家试想，如果现在要添加一个创建Administrator用户的事件，需要修改以下几处：

(1)在类Create中增加一个createAdministrator(的方法；

(2)在类CreateProcessor中增加一个处理创建Administrator用户的方法

processAdministrator(Create create);

(3)在调用方法的地方增加一个新的administratorCreateClick方法(createAdministrator的Button

事件)。那么，为了便于今后的维护，你可以怎样重构这些代码？（考虑抽象、接口等方式）

请在下方写出重构后的代码，可辅以自然语言注释。