**2017-2018学年 第1学期**

**2017级《面向对象建模技术》期末考试（A卷）答案及评分标准**

**考试时间：2018年1月18日**

**一、选择题，每题1分，共计20分。每个括号里的数字为该空的题号，从该题号对应的备选答案中选择最佳的一项，每空限填一个选项。请按如下表格在答题册上答此题。**

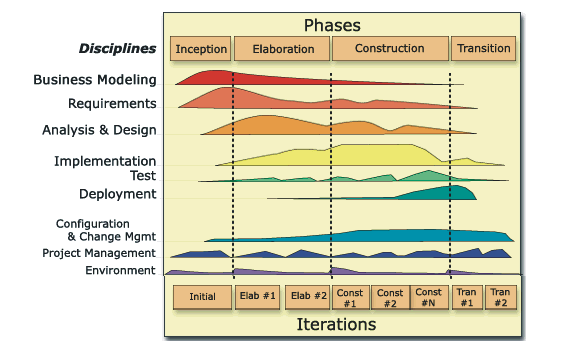
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1~5题** | **B** | **D** | **C** | **C** | **A** |
| **6~10题** | **D** | **B** | **C** | **D** | **D** |
| **11~15题** | **D** | **C** | **B** | **A** | **D** |
| **16~20题** | **C** | **B** | **D** | **A** | **B** |

**二、判断正误，每题1分，共计10分。请按如下表格在答题纸上答此题。**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1~5题** | **F** | **T** | **F** | **F** | **F** |
| **6~10题** | **T** | **F** | **T** | **T** | **F** |

**三、完成下列各题，在答题纸上写出答案，共计25分。**

1. （5分）用一个示意图解释RUP过程模型，要求说明生命周期的四个阶段（phase）、9个工作流（discipline）、迭代的含义以及在RUP中的应用。



Earliest iterations address greatest risks. Each iteration produces an executable release, an additional increment of the system. Each iteration involves activities from all disciplines. The relative amount of work related to the disciplines changes between iterations. For instance, during late Construction, the main work is related to Implementation and Test, and very little work on Requirements is done.

2. （5分）说明UML结构中构造快（Building Block），要求说明4种事物（Things）、4种重要关系（Relationship）以及至少10种图（diagram），并指明那些图是表达系统行为的，哪些图是表达系统结构的。

物件：结构物件、行为物件、分组物件和注解物件

关系：关联、泛化、依赖和实现

图形：第一类是结构图：类图(Class Diagram) 、组件图(Component Diagram) 、对象图(Object Diagram) 、复合结构图(Composite Structure Diagram) 、部署图(Deployment)和包图(Package Diagram)

第二类是行为图：活动图(Activity Diagram)、用例图(Use Case Diagram)、状态图(State Diagram)、顺序图(Sequence Diagram)、通信图(Communication diagrams) 、交互综合图(Interaction Overview Diagram)和时间配置图(Timing Diagram)

3.（3分）举例说明什么是Stereotype（构造型）以及构造型的三种表达方式。

构造型机制是指在已有的模型元素基础上建立一种新的模型元素。它与现有元素要相差不多，只是多一些特别的语义。标签（label）、大图标（Icon）、装饰（decoration）

4. （3分）以下图所示的类图片段为例，说明对象模型与数据模型，以及类图与源程序之间的对应关系。（注：①请根据该类图片段给出数据库表table，并在其上标出主键和外键；②用C++或Java语言给出与该类图片段对应的程序片段）



|  |  |  |
| --- | --- | --- |
| CompanyName | CompanyId(PK) | CompanyAddress |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| CompanyID(FK) | EmployeeName | Salary | EmployeeID(PK) |
|  |  |  |  |
|  |  |  |  |

class Company;

class Employee

{

public:

Employee();

~ Employee();

private:

String EmployeeName;

String EmployeeID;

double Salary;

Company\* theCompany;

};

#include "Employee.h"

class Company

{

public:

Company();

~ Company();

private:

String CompanyName;

String CompanyID;

String CompanyAddress;

Employee\* theEmployees[];

};

5. （3分）简要说明体系结构分析(Architecture Analysis)的目的和主要步骤。

**Purpose**:

* + To define a candidate architecture for the system based on experience gained from similar systems or in similar problem domains.
  + To define the architectural patterns, key mechanisms, and modeling conventions for the system.
  + To define the reuse strategy.
  + To provide input to the planning process.

**Architecture Analysis Steps:**

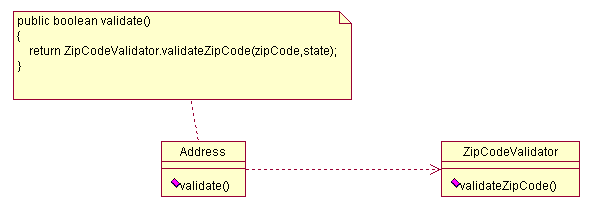
* + Identify Key Concepts
  + Define the High-Level Organization of the model
  + Identify Analysis mechanisms
  + Identify Key Abstractions
  + Create Use-Case Realizations
  + Checkpoints

6. （3分）说明是用例实现（Use-case Realization），用例实现结构中应该包含哪些图。

An Use-Case Realization describes how a particular use case is realized within the Design Model in terms of collaborating objects. A Use-Case Realization in the Design Model can be traced to a use case in the Use-Case Model.

The use-case realization structure includes the following elements: Use-case realization packages, Traceabilities diagrams, Interaction diagrams and Class diagrams.

7. （3分）类之间的依赖关系在程序中有多种不同的实现，任意写一段程序代码（Java或C++），用以指明至少一种实现方式。



**四、综合应用题（45分）**

**1.** 阅读下列说明，回答问题 1问题 3，将解答填入答题纸的对应栏内。

【问题1】（5分）根据说明中的描述，给出图1中C1~C9所对应的类名。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C1: System | C2: User | C3: Student | C4: Faculty | C5: Staff |
| C6: Publication | C7: ConfPaper | C8: Journal Article | C9: TechReport | 注：C3、C4、C5可互换  C6、C7、C8可互换 |

【问题2】（2分）根据说明中的描述，给出图1中类C6~C9的属性。

C6: Publication的属性有：题目、作者以及出版年份

C7: ConfPaper的属性有：会议名称、召开时间以及召开地点

C8: Journal的属性有：期刊名称、出版月份、期号以及主办单位

C9: TechReport的属性有：由学校分配的唯一ID

【问题3】（3分）第五项需求中提到“用户也可以在系统中为某篇出版物注册引用通知，若有新的出版物引用了该出版物，系统将发送电子邮件通知该用户”，在图1添加适当的类和关系，并适当添加类的操作和对操作的注释，使之成为观察者模式（Observer）的一个应用实例，并解释哪个类（或哪几个类）扮演了Observer角色，哪个类（或哪几个类）扮演了Subject角色，改进后的设计有什么优点。

改进后的设计定义了一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都会得到通知并自动更新。



2. [问题 1]（2 分）举例说明该设计有何不足之处；

该设计的明显缺点是可扩展性差。例如，要添加一种新的具体的鱼类，就需要在该类的属性部分增加一个新的鱼类的属性。导致可扩展性差的原因是：该设计实际上具有自顶向下的依赖。需要按照依赖倒转原则，引进接口类，改善可扩展性。

[问题 2]（3 分）按照自己的想法，重新设计，并且画出新的设计类图。（注：新设计至少应该包括两个接口类，一个是鱼类，一个是贝类）



3. （5分）

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Logon System | 2. Receive Web Form | | 3. Get Student Record |
| 4. Display Error Message | 5. Display Current Student Data | | 6. Enter Changes |
| 7. Validate Changes | 8. Display Error Message | | 9. Update Student Record |
| 10. Create Change Student Journal Record | | 11. Display Confirmation | |

注：9和10可互换

4. (18分)

（1）为Browse Music编写一个用例规约，其事件流根据上述描述进行定义。

|  |
| --- |
| 用例名称：Browse Music |
| 用例编号：001 |
| 简短描述：该用例用于用户选择歌曲列表 |
| 主要参与者：user |
| 次要参与者：无 |
| 前置条件：用户在界面上点击“浏览音乐” |
| 主事件流：  （1）用户从下拉列表中选择音乐类型；  （2）系统获得一系列与该类型匹配的演奏家、音乐家或团体，并用下拉列表进行格式化显示  （3）用户从演奏家下拉列表中选择一个演奏者；  （4）系统取回并显示该演奏家的所有CD和音乐作品  （5）用户选择某个CD  （6）系统取回并显示该CD内的所有歌曲  （7）用户多重选择歌曲  （8）用户单击“Add To Shop Cart（添加到购物车）”图片，导致添加到购物车用例的执行 |
| 后置条件：用户选择了歌曲列表 |
| 备选流：无 |

（2）使用边界对象、控制对象和实体对象绘制一个顺序图。



（3）编写一个列表，包含所有被传递到这些类的实例的消息，包括名称、参数和数据类型以及返回值的数据类型，可以作出有关数据的任何假设。

|  |  |  |
| --- | --- | --- |
| void 浏览音乐（） | void选择歌曲（） | String 取回音乐类型（） |
| String 选择音乐类型（） | String 取回演奏者（string 音乐类型） | void 显示演奏者（String 演奏者） |
| String 选择演奏者（） | String 取回演CD（String 演奏者） | void 显示CD（String CD） |
| String 选择CD（） | String 取回歌曲（String CD） | void 显示歌曲（String 歌曲） |
| String 选择歌曲（） | void 添加到购物车（） |  |

（4）根据你画的顺序图创建相应的协作图。



（5）创建一个初始的VOPC。



5. 阅读下列说明和程序代码，将应填入（n）处的字句填写完整。**（问题1和问题2任选一题完成）**

**【问题1】（7分）Java代码**

1. FlyBehavior flybehavior
2. TakeOffBehavior takeOffBehavior
3. flyBehavior.fly()
4. takeOffBehavior.takeOff()
5. extends
6. SubsonicFly()
7. VerticalTakeOff() （1、2可互换）

**【问题2】（7分）C++代码**

1. FlyBehavior \*flyBehavior
2. TakeOffBehavior \*takeOffBehavior
3. flyBehavior->fly()
4. takeOffBehavior->takeOff()
5. SubSonicFly()
6. VerticalTakeOff()
7. ~Helicopter() (1、2可互换)