软件设计基础课程作业

|  |  |
| --- | --- |
| **姓名:** | **周凡翔** |
| **学号:** | 201806150231 |

* 特别注意：

1. 所有作业的文档均在此文档中完成，即：只提交一份.docx文档；
2. 请将此文档（Labwork.docx）重新命名为：“姓名-学号.docx”；
3. 所有实验代码和模型等均用单独的文件夹存放，按照作业发布的次序，依次命名为“Lab01”、“Lab02”等，并将它们压缩为一个“code.zip”文件。
4. 所有作业上传到超星系统，你的提交的作业的组织形式应该如下图所示：



# 结构化程序设计 VS. 面向对象程序设计：素数查找和约瑟夫环

1. 素数问题的Java的结构化实现中，iCounter++，这一行代码会导致低效，怎么修改？

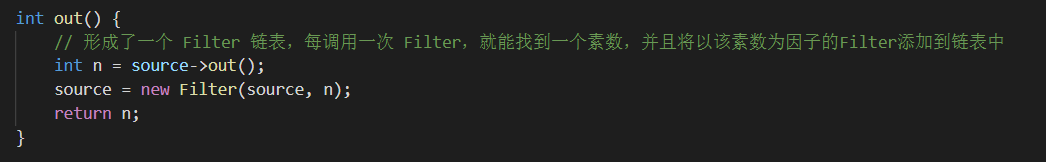
++iCounter;  
while(sieve[iCounter-2]  == 0)  
    ++iCounter;  
iCounter = sieve[iCounter-2];

1. 如何理解素数问题的C++的面向对象实现？包括：Filter链表怎么形成？如何递归调用？如何判断一个数是否可以被所有的Filter中的因子整除？

Filter不是链表，source数组才是链表，链表的初始化是在父类Item中实现的，链表的添加是由筛子从Filter中赋值的。

通过out方法，调用Filter方法，读取Filter类中out方法return的素数，加入source这个链表中。Filter本身不是链表，父类在Sieve中out函数的source数组才是链表，Filter是提供素数的工具类。

递归调用发生在筛子Sieve的out方法。



首先主函数指定counter 的初始值，然后Sieve取到这个数，然后在Sieve中递归调用Filter。当一轮结束以后，counter++开始下一轮调用。





1. 我们给出的素数问题的面向对象实现是基于C++的，如何修改成Java？

public class PrimerNumberOOP {

public static void main(String args[]) {

Counter c(2);

Sieve s(c);

int next, n;

n = 50;

// cin >> n;

while (1) {

next = s.out();

if (next > n) break;

System.out.println(next+" ");

}

return 0;

}

class Item {

public

Item source;

Item(Item src) { source = src; }

int out() { return 0; }

};

class Counter public Item {

int value;

public

int out() {

return value++;

}

Counter(int v) : Item(0) { value = v; }

};

class Filter : public Item {

int factor; // 素数因子

public:

int out() {

while (1) {

int n = source->out(); // 实际上形成了递归调用，递归检查当前 n 是否能被所有的比n小的素数整除

if (n % factor) return n; // 返回不能整除当前素数的n

}

}

Filter(Item \*src, int f) : Item(src) { factor = f; }

};

class Sieve : public Item {

public:

int out() {

// 形成了一个 Filter 链表，每调用一次 Filter，就能找到一个素数，并且将以该素数为因子的Filter添加到链表中

int n = source->out();

source = new Filter(source, n);

return n;

}

Sieve(Item \*src) : Item(src) {}

};

}

1. 约瑟夫环（数据结构中的一个经典问题）：n个人围成一圈，顺序排号，从第1个人开始报数（从1到M报数）， 凡报到 M 的人退出圈子，问最后留下的是原来第几号的哪位。问题：如何用结构化方法实现？如何基于面向对象（ OOP ）实现？

**SOP**

#include<stdio.h>

int main()

{

int i,n,m;

scanf("%d%d",&n,&m);

int r=0;

for(i=2;i<=n;i++)

r=(r+m)%i;

printf("%d\n",r+1);

return 0;

}// 定义小孩类

class Child{

public Child(int position){

this.position = position;

}

public int number; // 小孩当前的报数

public final int position; // 小孩的初始位置,固定不变的

public Child beforeChild; // 小孩前一个孩子

public Child nextChild; // 小孩后一个孩子

}

public class Test {

public static void main(String[] args) {

int n = 500,k = 1,m = 3;

Child[] children = new Child[n];

// 初始化小孩子,将孩子相互关联起来

// 可以理解为,手拉手围成一个圈

// 第一个孩子跟最后一个孩子拉手

for (int i = 0; i <n; i++) {

children[i] = new Child(i+1);

if(i > 0){ // 第二个人开始

together(children[i-1],children[i]);

}

if(i == n-1){ // 最后一个

// 关联到第一个

together(children[i],children[0]);

}

}

OOP

// 取第一个孩子开始报数

Child currentChild = children[k-1];

currentChild.number = 1; // 第一个孩子当然报1

// 循环,一直报数,只剩下1个小孩就停下来

while(true){

// 如果下一个小孩引用的对象是自己说明只剩下一个人了

if(currentChild.nextChild == currentChild){

break; // 停止报数

}

// 如果报数是3,把他的上一个小孩跟他的下一个小孩关联起来

// 这意味着他失去关联,退出

if(currentChild.number == m){

together(currentChild.beforeChild,currentChild.nextChild);

// 下一个小孩重新报数1

currentChild.nextChild.number = 1;

}else if(currentChild.number < m){// 如果不是3,转移到下一个小孩

// 下一个小孩报数+1

currentChild.nextChild.number = (currentChild.number + 1);

}

// 轮到下一个小孩

currentChild = currentChild.nextChild;

}

System.out.println("第"+currentChild.position+"个小孩"); // 第436个小孩

}

// 相互关联,手拉手

private static void together(Child before,Child next){

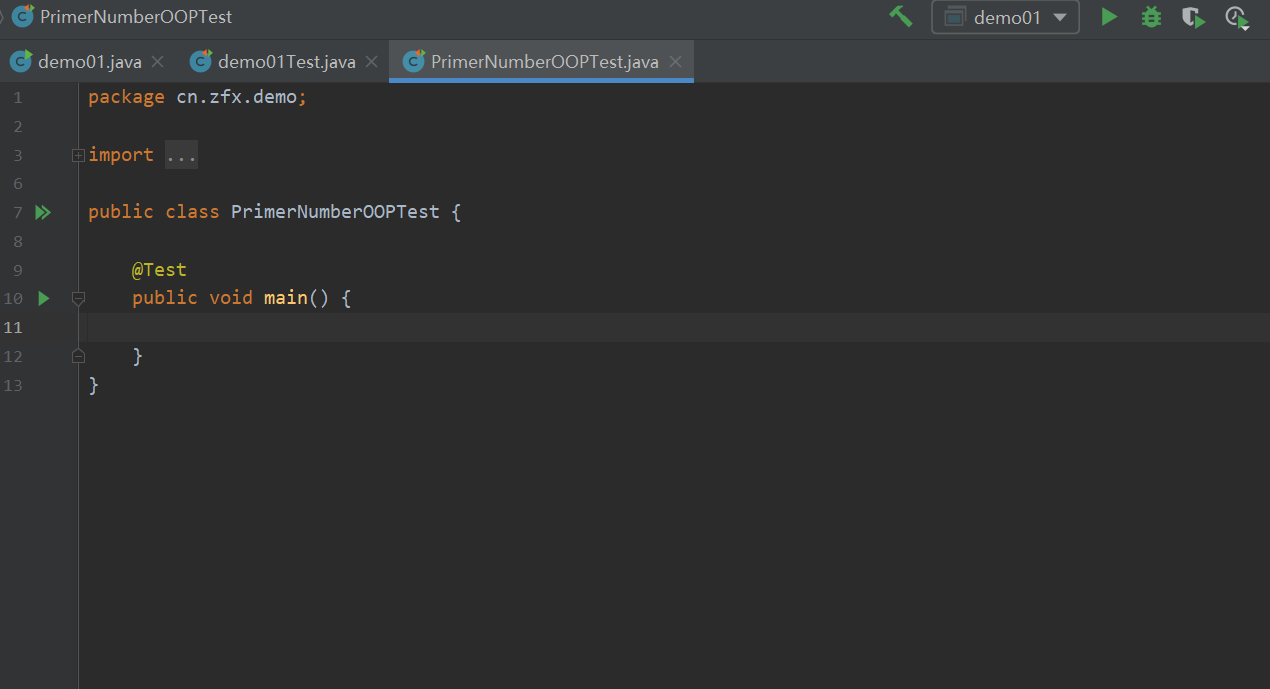
before.nextChild = next;

next.beforeChild = before;

}

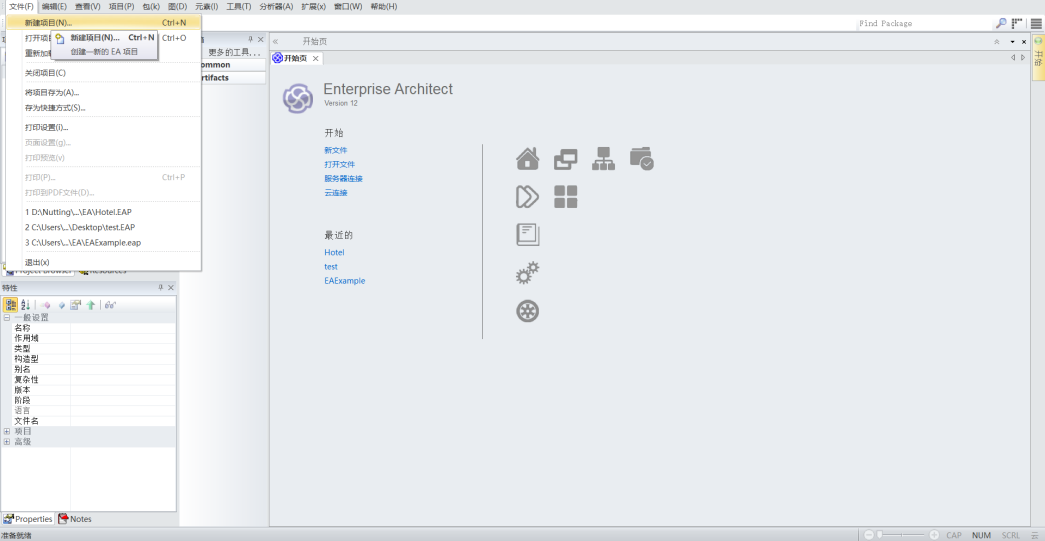
}

1. 基于JUnit对所给出的Java实现进行单元测试。

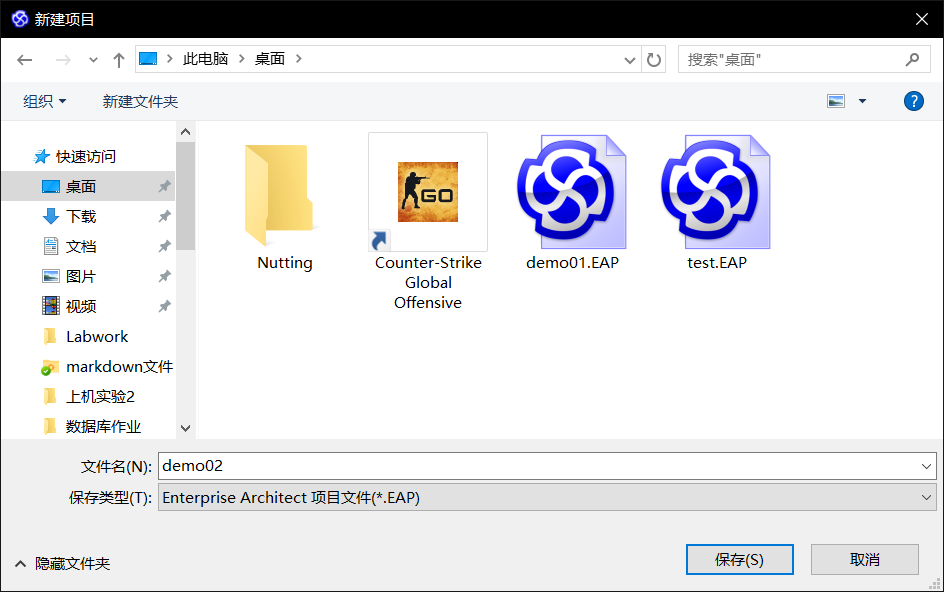


# 业务建模

1. 完成课本3.7节中的所有练习题（自行练习，无须在此做答）。
2. 旅店业务建模：参照3.5.2节中的描述，仿照/ model/Hotel/EA/Hotel.EAP，新建EA工程，构建自己的业务模型。试用文字和截图说明：
   1. 如何构建EA工程？

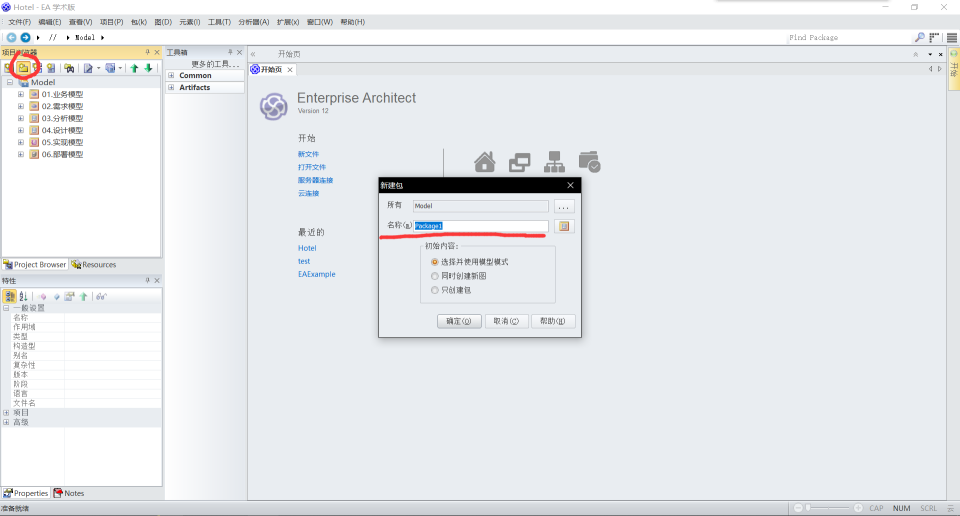


CTRL+N或者菜单导航栏<文件><新建>



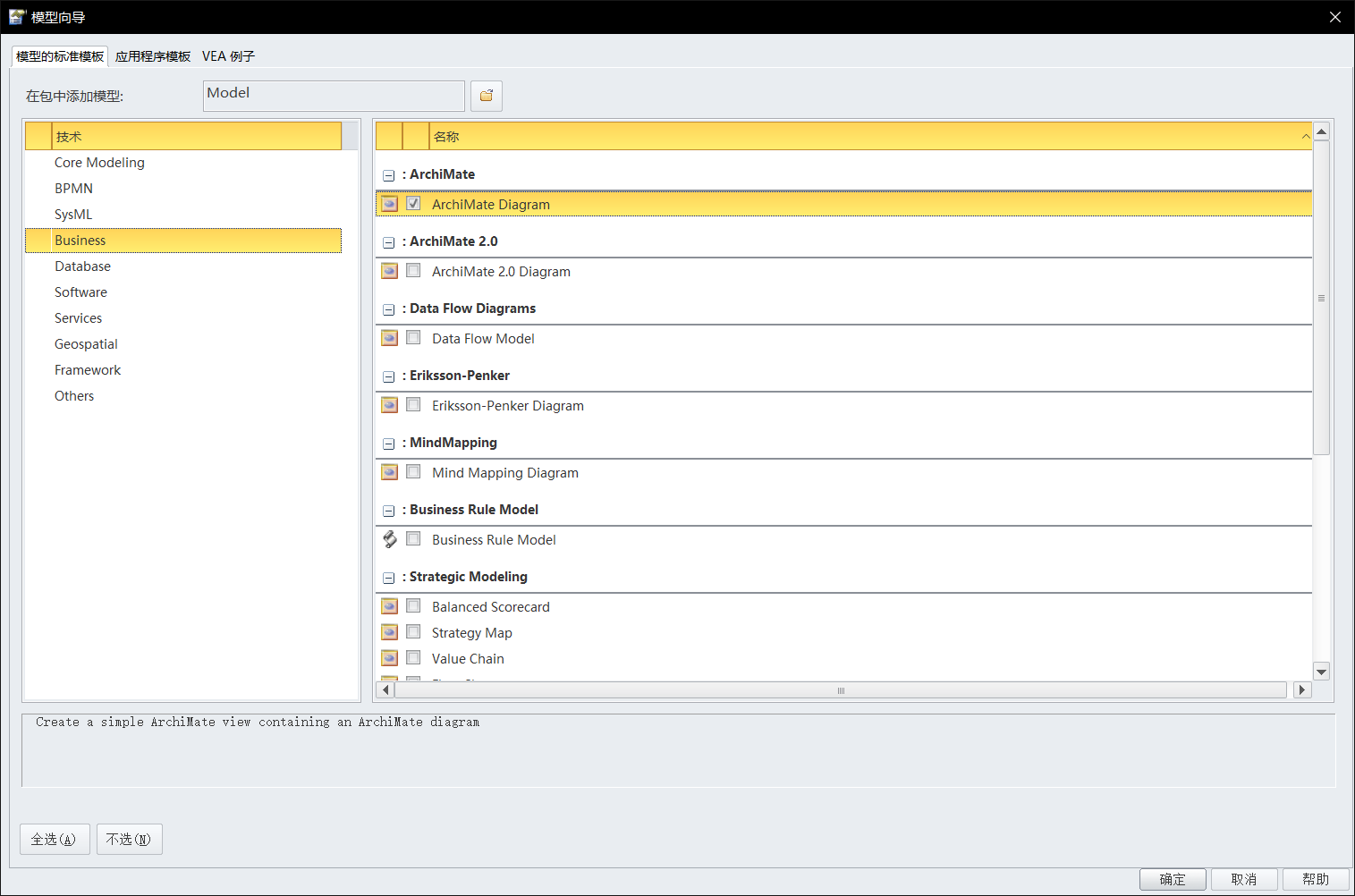
选择目录创建EAP文件

* 1. 如何在EA中添加package？



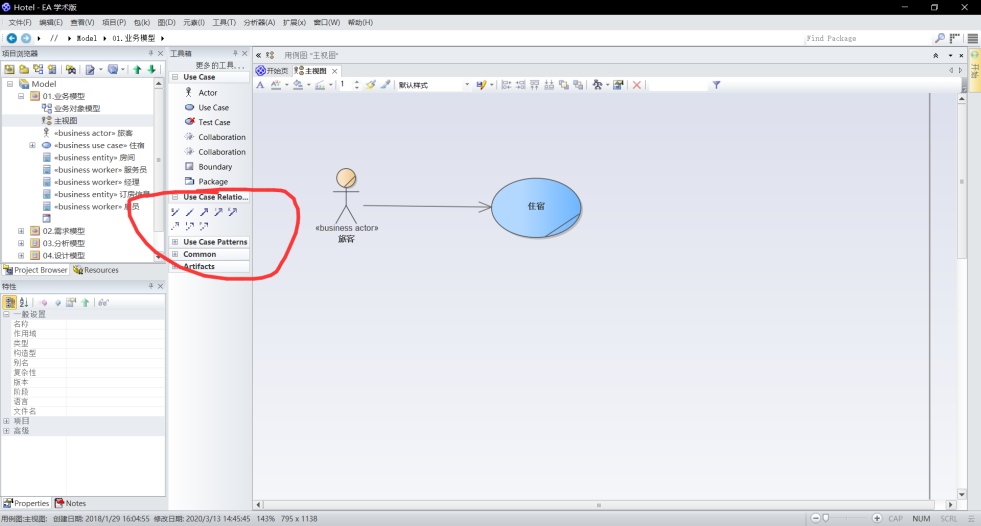
在项目导航栏点击新建包，用红圈圈出

* 1. 如何在EA中添加“业务模型”？



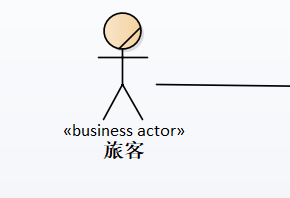
随便点，造出来要改动的

* 1. 如何拖动鼠标在EA中建立2个类之间的关联关系？

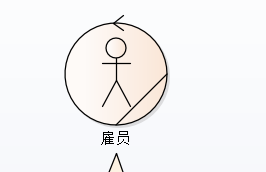


* 1. 业务模型中的Business Actor、Business Worker、Business Use Case都分别用什么图形表示？

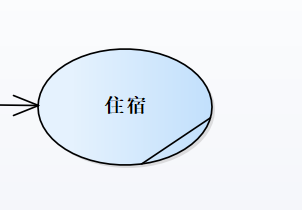
Business Actor



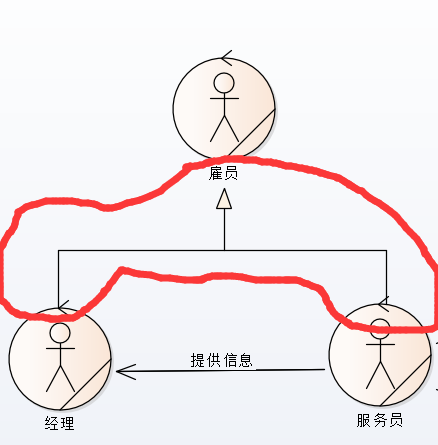
Business Worker



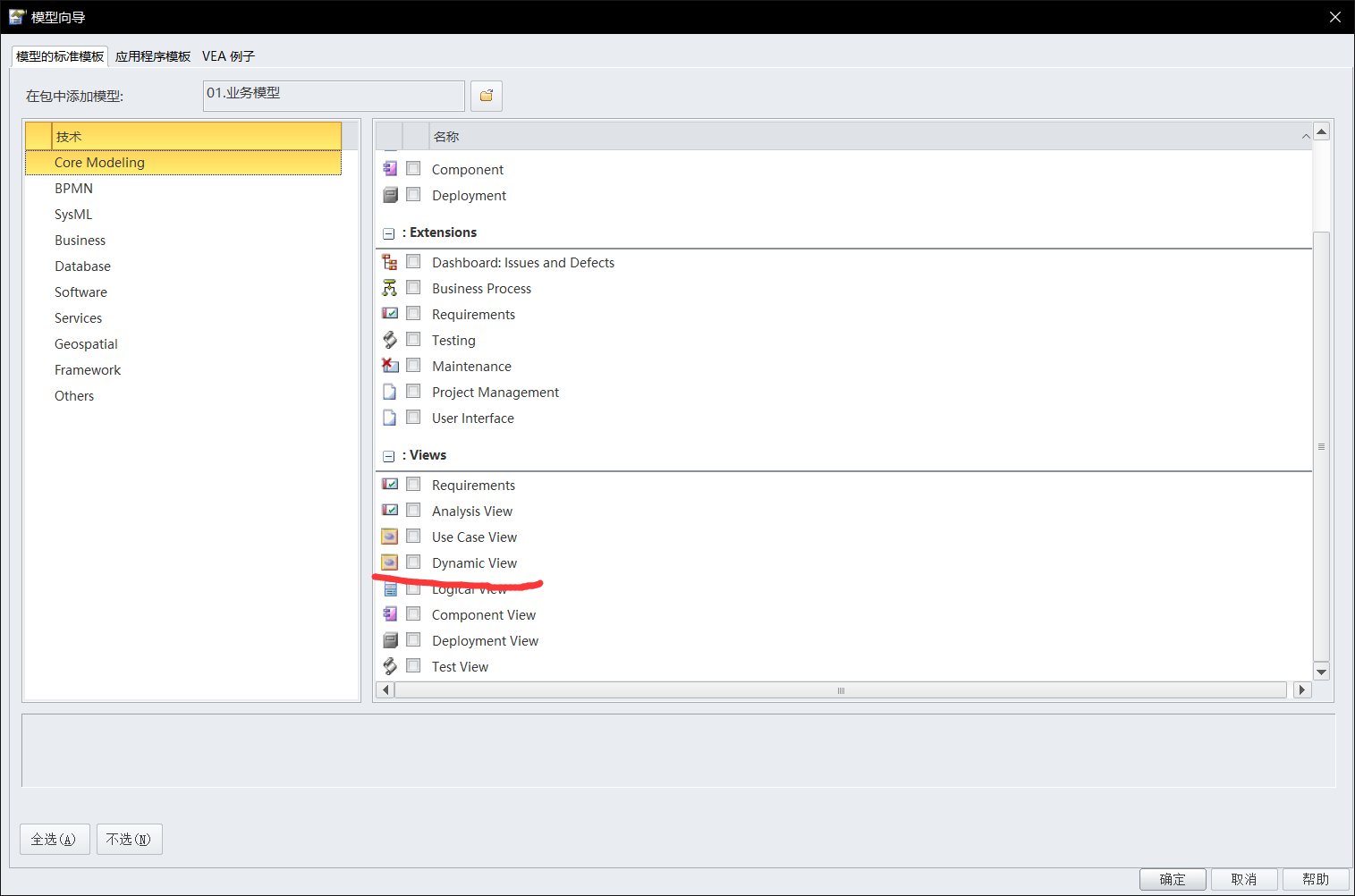
Business Usecase



* 1. 在业务对象模型中，如何操作才能建立起父类与多个子类之间的树形泛化关系？



* 1. 如何建立活动图？

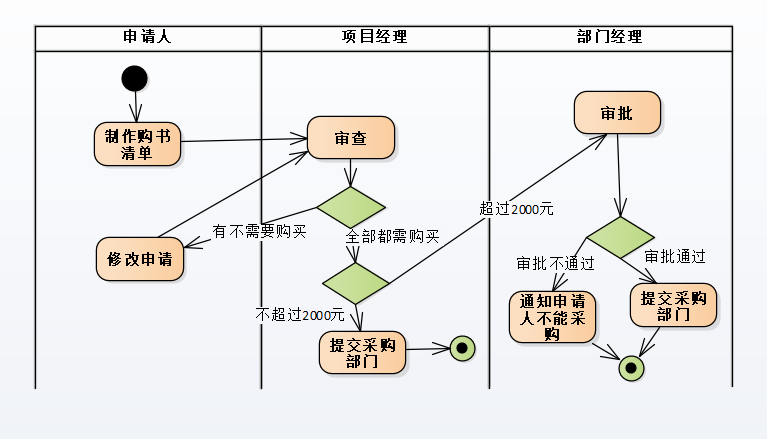


Dynamic view 新建建立

* 1. 如何用键盘轻微移动图形元素？

Shift+↑/↓/←/→

1. 绘制活动图：在EA中绘制3.7节习题三（应用题）中第2题的活动图。



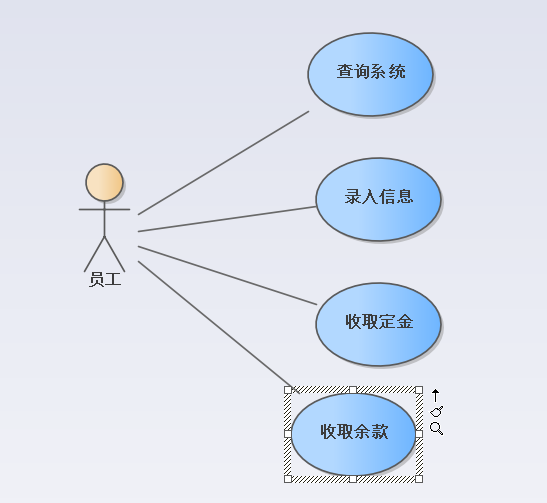
# 用例建模

## 简答题（自行练习，无须在此做答）

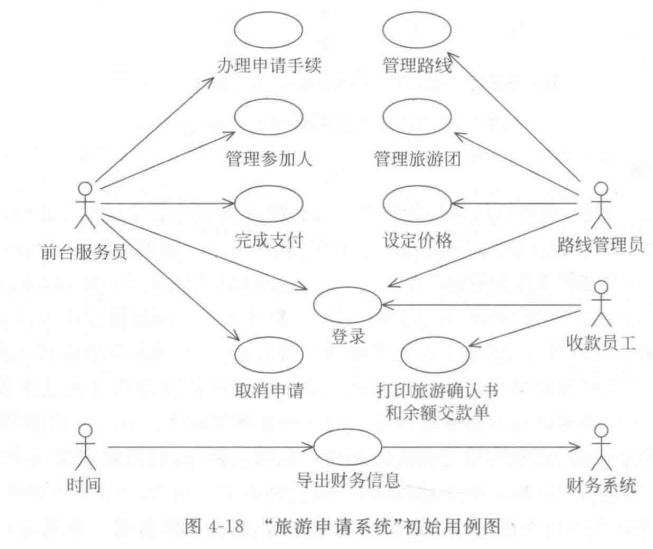
1. 参考课本4.1~4.2节及视频“Ch04-用例建模-需求-用例建模流程-从业务模型获取需求.mp4”，回答下列问题：
   1. 什么是需求，有哪些类型的需求?
   2. 在业务建模后，如何寻找业务改进点?哪些业务改进点可能会作为需求?
2. 参考课本4.3节及视频“4.3-用例建模讲解.mp4”，回答下列问题：
   1. 从初始需求出发（与从业务用例模型出发相对比），进行用例建模，基本步骤有哪些？
   2. 什么是系统参与者，识别参与者的要点包括哪些?
   3. 什么是系统用例，获取用例的要点包括哪些?
   4. 系统参与者之间存在什么关系？
   5. 系统用例之间存在什么关系？
3. 阅读课本4.4节“编写用例文档”，回答下列问题：
   1. 什么是涉众，涉众和参与者有什么区别和联系?
   2. 什么是用例的前置条件和后置条件，它们有什么作用，定义时需要注意什么?
   3. 什么是用例的事件流，描述事件流时需要注意什么?有哪几种事件流，它们之间有什么区别和联系?
   4. 用例的补充约束主要包括哪些内容?如何描述补充约束?
4. 参考课本4.5节“重构用例模型”，回答下列问题：
   1. 在用例模型中，可以定义哪几种用例关系，它们有何不同?
   2. 什么是扩展点，扩展点有什么作用?
   3. 有哪些用例的分包策略，一般如何进行用例分包?
   4. 如何对用例进行分级，高优先级的用例有什么特征?

## 练习题

1. 观看教学视频“4.3-用例建模讲解.mp4”，阅读“旅游业务申请系统”的案例描述（详见：model/Tour/CASE02-LY01-案例描述文档.doc，或者4.3.5节的问题陈述），进行如下练习：
   1. 识别其中的系统参与者与系统用例，自行绘制系统用例图；



* 1. 与课本中的图4-18（“旅游申请系统”初始用例图）进行比较：将自己绘制的和课本中给出的用例图放在一起进行比较，总结自己设计中的不足和问题；



问题：

1.没有时间这个参与者

2.把员工三合一了

3.员工露了很多功能

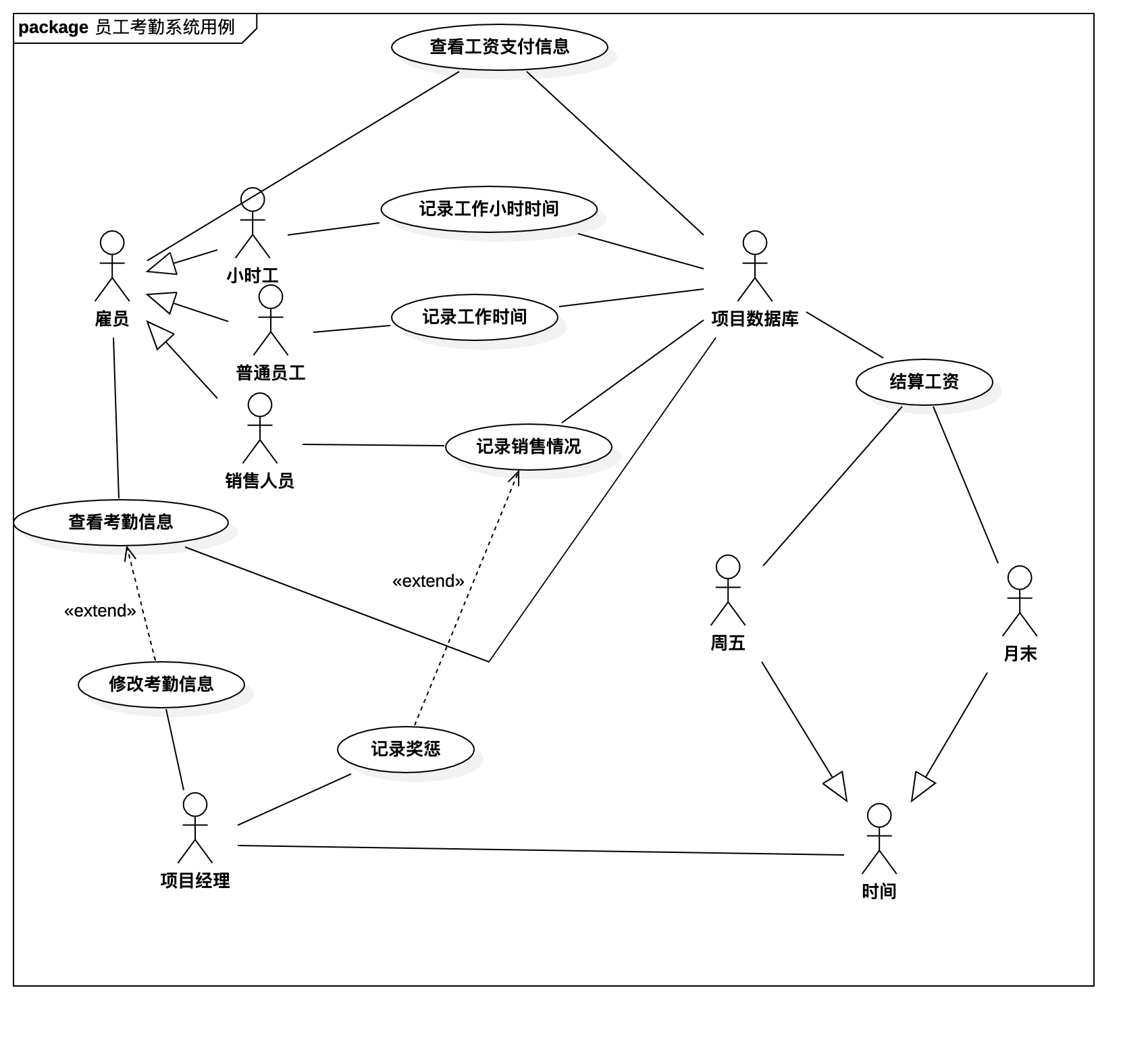
* 1. 阅读4.3.5节中的“2. 识别参与者”和“3. 识别用例”，比较自己的识别方法与课本中的异同，并进行总结。

**比较这个东西有没有参与系统，参与了就不是参与者。**

**用例尽量多**

1. 综合案例题：课后练习题三（应用题）之5和6任选一题。

答：第五题



# 用例分析

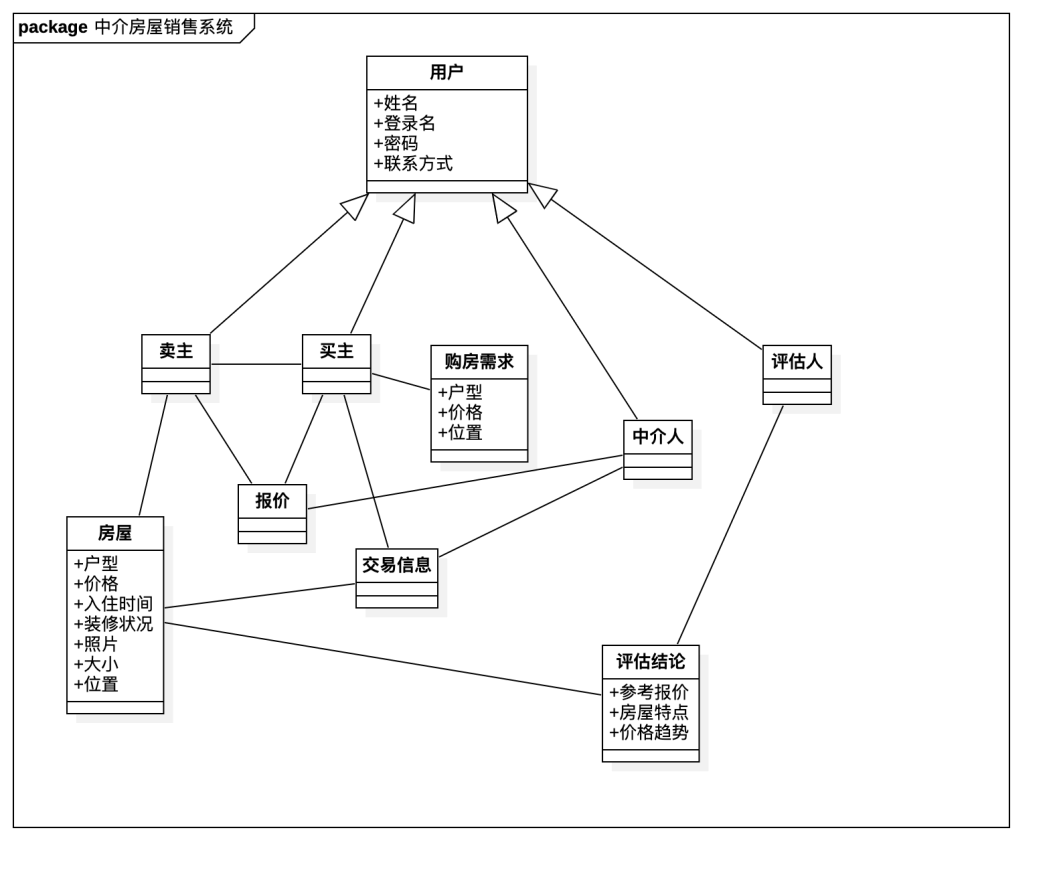
1. 定义分析类最终期望达成的最主要的目标是形成实体类类图。试根据课后练习题三.1中所陈述的系统业务背景，抽取系统中的实体类，以及类的属性和关系，并构造实体类类图。

答：

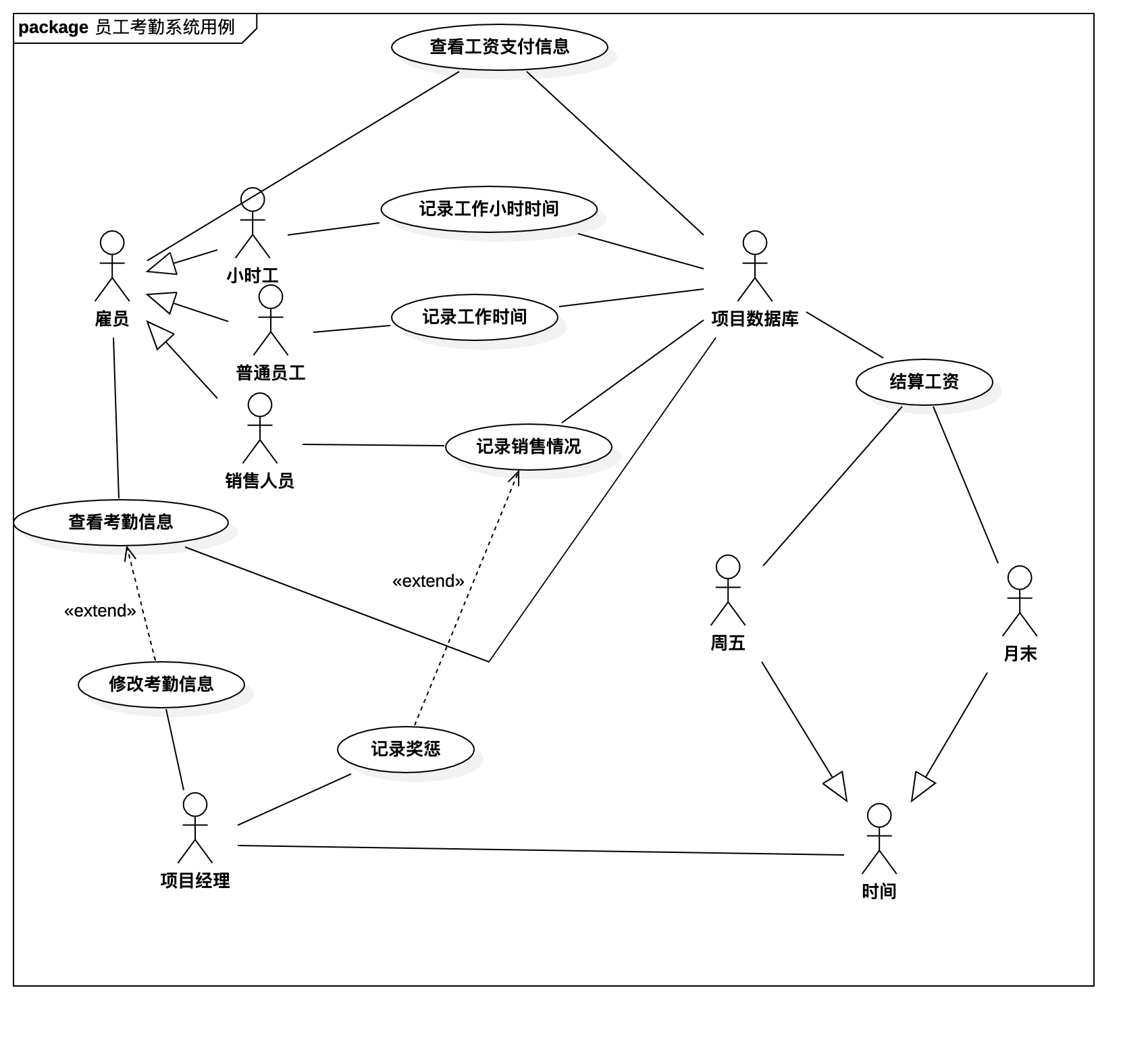
核心的类：房屋、评估结论、买主、卖主、购房需求、交易

核心类的属性：房屋类、用户类的关键属性

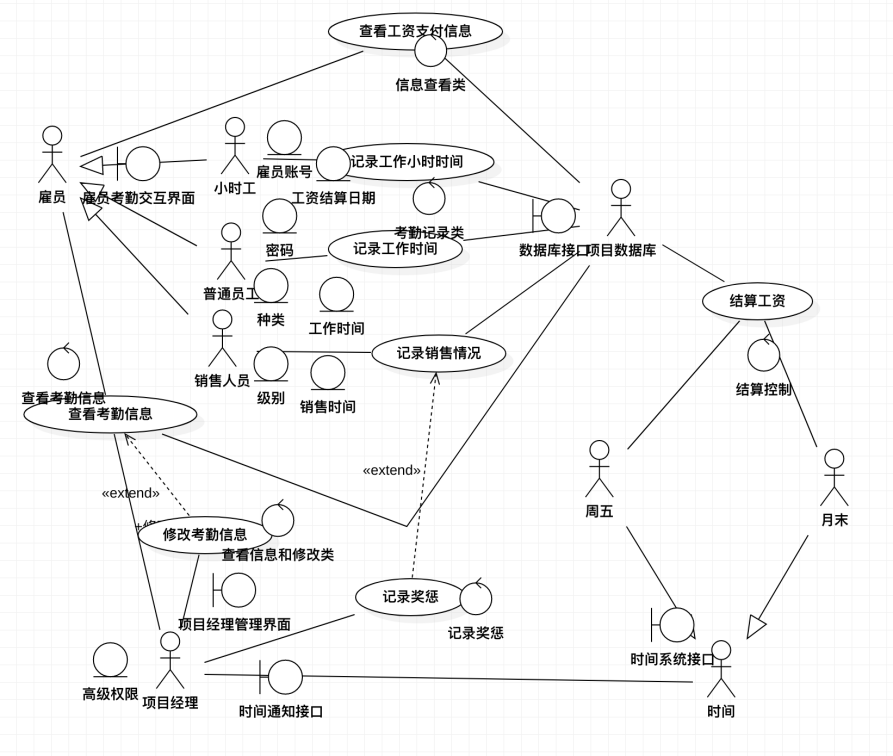
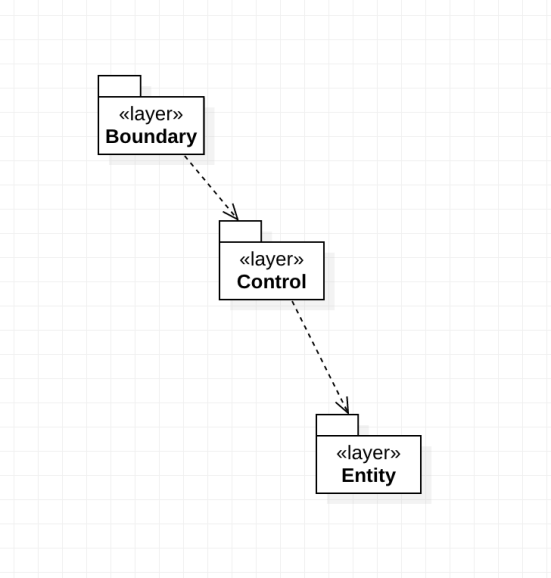
关系的使用：典型的关联关系（如交易和房屋、买主之间；评估结论与房屋之间），另注意泛化关系的使用。



1. 综合案例题：课后练习题三（应用题）之2和3任选一题。

答：

1. 备选架构：



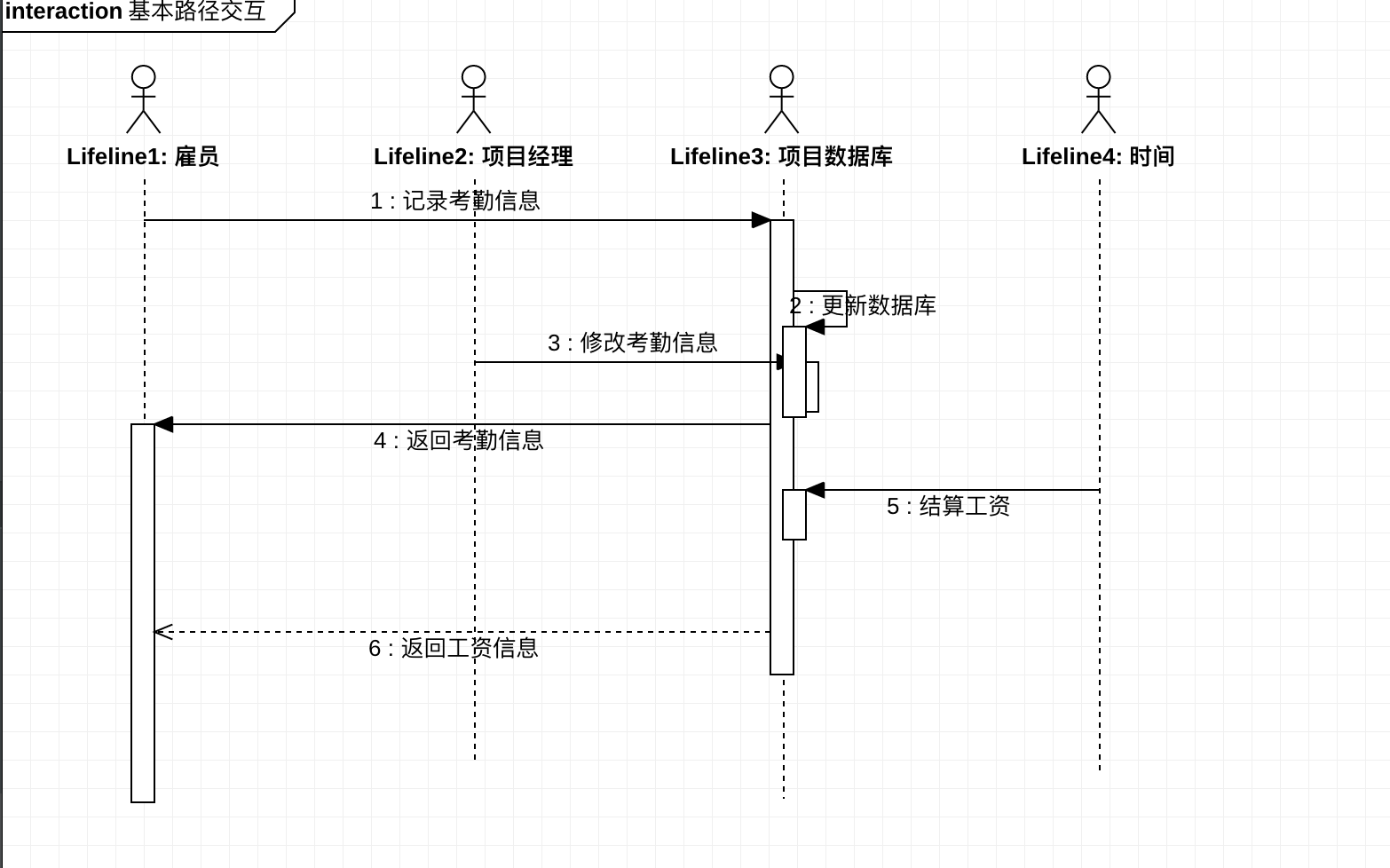
B:雇员考勤交互界面、项目经理管理界面、时间通知接口、时间系统接口、数据库接口

C：工资信息查看类、查看考勤信息类、考勤记录类、结算控制类、支付类、查看信息和修改类、记录奖惩类。

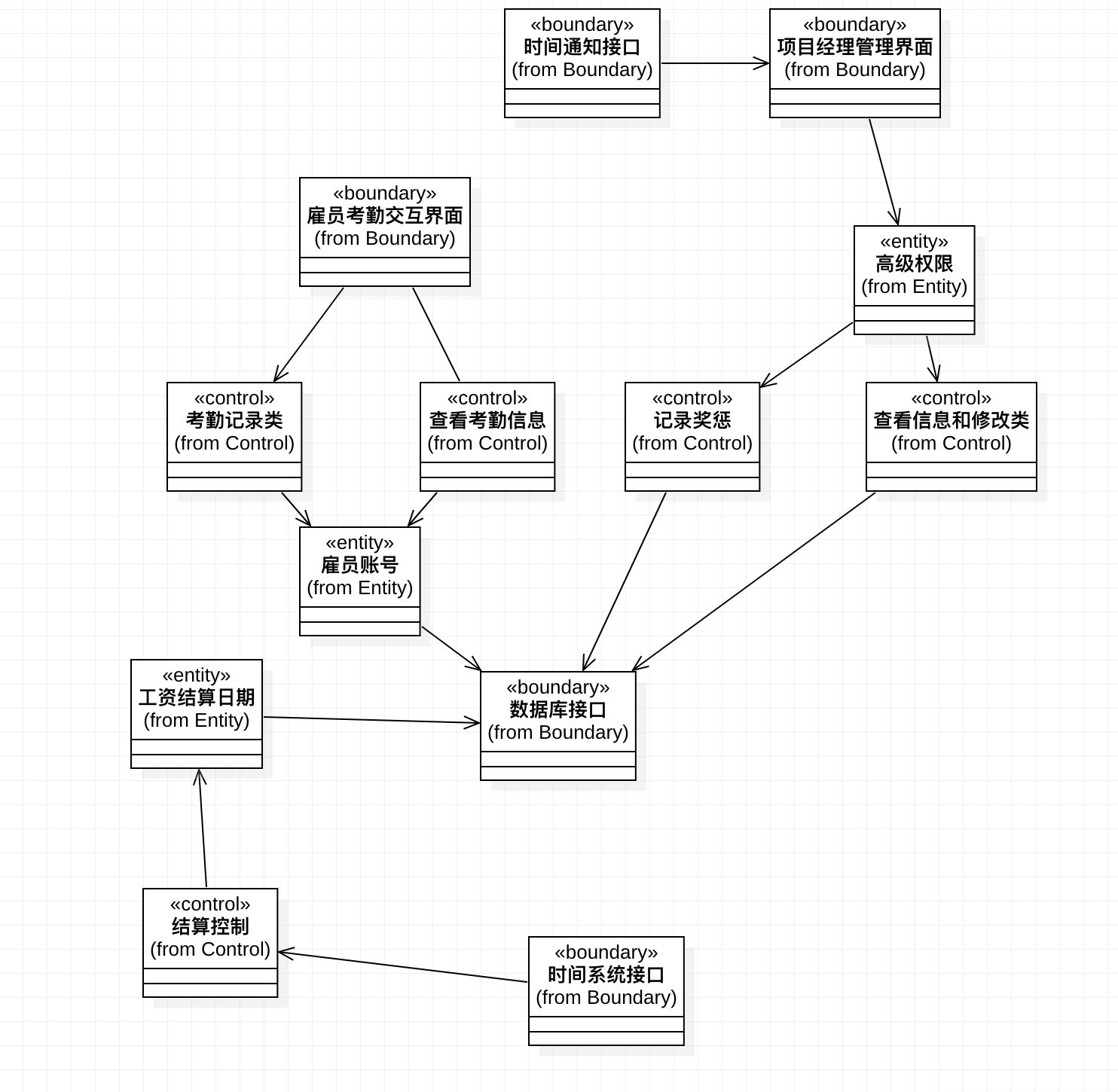
E：……

（2）

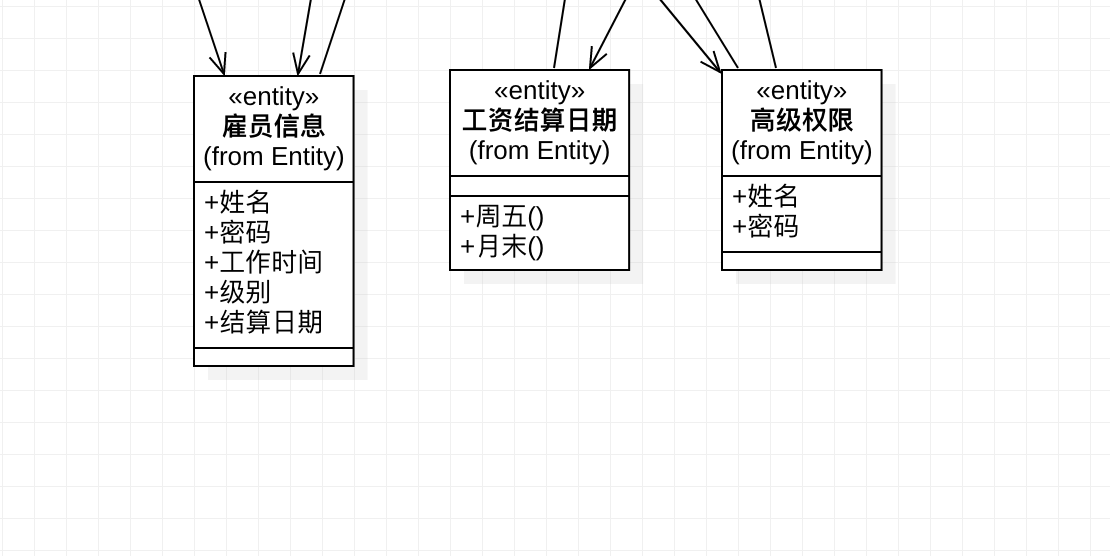
（2.1）基本路径的交互



（2.2）用例实现的参与类类图



（3）构造实体的完整类类图



# 面向对象的设计模式

1. Java提供了该观察者模式的实现框架（在java.util包中提供了Observable类和Observer接口），请给出使用该观察者模式进行程序设计的范例。

答：见code文件夹中的lab05

1. 基于策略模式的重构调整。“影片出租店管理程序”的问题陈述及初步实现代码详见教案的7.5.1~7.5.2节，试根据7.5.3~7.5.5节描述（问题分析、类图和序列图）给出你的改进程序。

答：见code文件夹中的lab05

# **架构设计（以小组为单位，五人/小组）**

1. **设计/模型图绘制：**
   1. **参考“model\Tour\EA\TourModel\_Design.eap”，绘制8.4.2节所描述的持久性机制的类图和活动图（图8-20~图8-23），并详细说明在EA中进行相关操作的具体流程，尤其是添加基础设施层(Infrastructure)的过程。**

答：

* 1. **绘制“图8-24 医院预约挂号系统的进程视图”**

答：

* 1. **绘制“图8-26 为进程分配设计元素”**

答：

* 1. **绘制“图8-27医院预约挂号系统的部署模型”，熟悉设备、执行环境、通信路径等概念；**

答：

* 1. **绘制“图8-28医院预约挂号系统的部分工件图”，熟悉工件、承载、逻辑组件等概念；**

答：

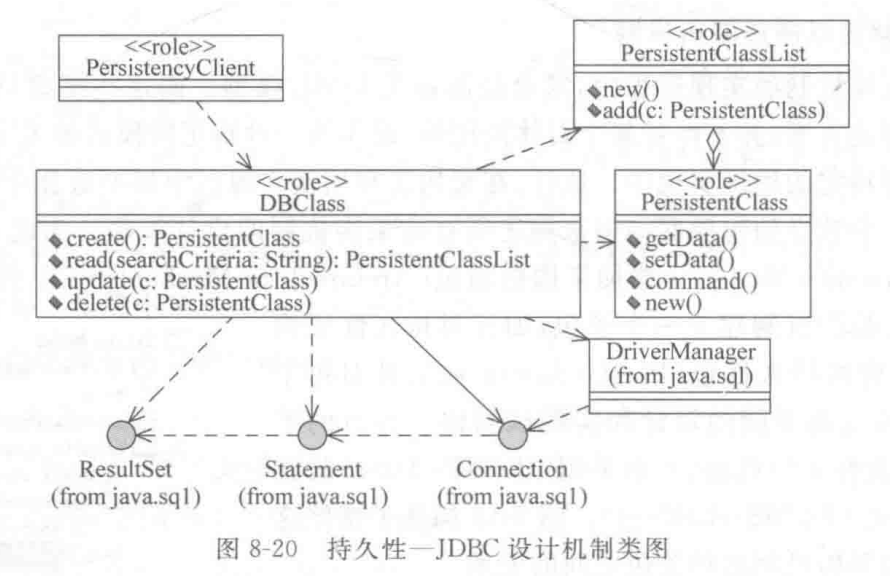
* 1. **绘制“图8-29 将工件部署到节点的示例”，熟悉“部署规范”；**

答：

* 1. **绘制“图8-31 分布-RMI机制设计类图”，了解RMI的原理。**

答：

1. **编程实践**
   1. **编写基于Java的数据库访问程序，结合“图8-20 持久性：JDBC设计机制类图”，理解8.4.2节所描述的持久性机制。DriverManager类的作用？Connection类、Statement类、ResultSet类之间为何存在依赖关系？ResultSet和PersistentClassList在存储数据上有何区别？**

答：

DriverManangr是JDBC驱动类，使得数据库可以响应JAVA虚拟机，从而提供JDBC的服务。

Connection类检查JDBC连接状态。Statement类是对传入的数据库内容进行操作。操作之前必须保证JDBC连接是成功的，即需要先检查一遍Connection Status，必须先使用Connection类。因此产生依赖。

ResultSet类保存操作结果值。操作结果值来自于Statement类。因此有依赖。

ResultSet类保存的数据是一组暂时保存的数据库关系结果值，内容是随时可以被替换的一段值。PersistentClassList类保存的数据是一个可增量的线性表，内容被保存在线性结构内，只会增量，不会被迭代。

* 1. **编写基于Java的多线程并发程序，理解8.5节中描述的运行时构架。**

答：代码见coding

运行结果如下：

window5线程在运行10

window5线程在运行9

window5线程在运行8

window4线程在运行10

window1线程在运行10

window3线程在运行10

window3线程在运行9

window6线程在运行10

window2线程在运行10

window2线程在运行9

window2线程在运行8

window9线程在运行10

window2线程在运行7

window2线程在运行6

window2线程在运行5

window2线程在运行4

window2线程在运行3

window2线程在运行2

window2线程在运行1

window1线程在运行9

window4线程在运行9

window5线程在运行7

window8线程在运行10

window8线程在运行9

window4线程在运行8

window1线程在运行8

window7线程在运行10

window9线程在运行9

window7线程在运行9

window6线程在运行9

window7线程在运行8

window3线程在运行8

程序运行就成为进程。进程processs，占用硬件资源并实现对应功能。thread是process的轻量级控制流，许多thread分解了process。thread并发控制，可以同时利用CPU和其他硬件资源，提高资源利用率和实现速度。

* 1. **编写基于Java的RMI程序，并结合图8-30和图8-31，了解RMI的工作原理。请特别说明UnicastRemoteObject类的作用。**

答：代码见coding

UniCastRemoteObject类，存在于java.rmi.server包中，使用public限制符，把该接口公开，以表示是用于远程调用的接口。这是RMI规则里必须实现的一步。相当于是开了一扇门，规定了信息流传递的信道和方向。

1. **应用题（二选一）**
   1. **[综合案例:员工考勤系统]结合已经完成的需求和分析模型，以本章介绍的架构设计方法，设计该系统架构。**

答：

* 1. **[综合案例:医院预约挂号系统]结合已经完成的需求和分析模型，以本章介绍的架构设计方法，设计该系统架构。**

答：