使用GRASP进行对象设计

创建者

我们遍历了所有由领域模型中衍生而来的类，考虑如何选择这些类的创建者，在实现低表示差异的基础上，我们将考虑的结果以表格的形式展现：

|  |  |  |  |
| --- | --- | --- | --- |
| 类名 | 从属 | 创建者 | 说明 |
| Student | 人员部分 | **Education System** | 教务系统中记录有学生的基本信息 |
| Teacher | 人员部分 | Education System | 教务系统中记录有教师的基本信息 |
| Academic Dean | 人员部分 | Education System | 教务系统中记录有教务员的基本信息 |
| Course | 课程安排部分 | Course Arrangement | 课程安排中聚集了大量的课程信息 |
| Course Arrangement | 课程安排部分 | Education System | 教务系统中记录有课程安排的基本信息 |
| Curriculum | 课程安排部分 | Course Arrangement | 课程安排中具有课表初始化的信息 |
| Member List | 课程安排部分 | **Course Selection Factory** | 选课记录中记录了课程相关的人员信息 |
| Score Record | 成绩部分 | Teacher | 老师具有初始化某一条具体的评分记录的数据信息 |
| Report Cart | 成绩部分 | **Score Data** | 总的成绩数据记录中含有初始化每个人的成绩单的相关数据 |
| Course Selection Operation | 选课部分 | Student | 学生具有初始化某一条具体的选课记录的数据信息 |
| Course Selection Record | 选课部分 | Education System | 由总的教务系统负责选课数据列表的初始化 |
| Exam Arrangement | 考试部分 | Education System | 由总的教务系统负责考试数据列表的初始化 |
| Exam Table | 考试部分 | Exam Arrangement | 考试总安排中包含了考表初始化的信息 |
| Exam | 考试部分 | Academic Dean | 由教务人员进行考试安排，教务人员含有考试初始化的数据的能力 |
| University Major | 学籍部分 | Education System | 教务系统中记录有各专业的基本信息 |
| Education Program | 学籍部分 | Education System | 教务系统中记录有学生学籍的基本信息 |
| Classroom | 其他 | **Map System** | 使用一个新类储存所有与地理位置相关的信息 |
| Lab | 其他 | Map System | 使用一个新类储存所有与地理位置相关的信息 |

通过应用创建者模式进行分析，我们发现了进行领域建模时的一些遗漏

1. 缺乏一个总体的类负责一些独一无二的信息的初始化，例如人员基本信息、整体课程安排等，因此我们需要一个能够代表系统整体的类 “Education System” ，该类主要负责部分类的初始化。
2. 教室与实验室两个类同样也缺少创建者，但是因为这两者并不包含于教务系统中，只是作为基本的信息与教务选课系统中的一些类相关联，因此我们新建了一个类 “Map System” 专门负责这些地理信息相关类的创建与生成。
3. 对象的创建常常具有相当的复杂性，例如为了性能而使用回收的实例，基于某些外部特征值有条件地创建一个或一族对象类的实例。在教务选课系统中选课记录可能需要被批量地创建选课记录，或是选课记录需要被频繁地增加、修改、删除，因此我们考虑使用一个工厂类 ”Course Selection Factory” 去完成课程信息地创建。
4. 关于成绩部分，我们之前在领域模型中建立了一个类 “Score Record” 用于记录具体的一条成信息记录，但是缺乏一个类能够揽括所有的成绩记录，并负责这些成绩记录的生成，因此我们新建了一个类 “Score Data”负责这一职责。

信息专家

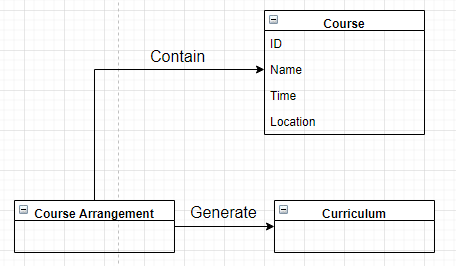
该模式主要需要解决地问题是“谁应该了解相关信息”

|  |  |  |  |
| --- | --- | --- | --- |
| 信息 | 类别 | 相关对象 | 说明 |
| 学生可选课程 | 课程部分 | Course Arrangement | 整体课程安排中含有课程所属类别，以此可以筛选出学生可选课程 |
| 学生已选课程 | 课程部分 | Curriculum | 学生已选的课程会更新入课表中，通过课表可以查询到学生已选课程 |
| 教师任教课程 | 课程部分 | Curriculum | 教师任教的课程会更新入课表中，通过课表可以查询到教师任教课程 |
| 学生课表 | 课程部分 | Curriculum | 类基本职能 |
| 教师课表 | 课程部分 | Curriculum | 类基本职能 |
| 选课学生名单 | 课程部分 | Member List | 类基本职能 |
| 课程基本信息 | 课程部分 | Course | 类基本职能 |
| 教师开课申请 | 课程部分 | **Course Application** | 区别于课程内容等描述内容，具体到某一学期的课程信息 |
| 教师考表 | 考试部分 | Exam Table | 类基本职能 |
| 学生考表 | 考试部分 | Exam Table | 类基本职能 |
| 考试基本信息 | 考试部分 | Exam | 类基本职能 |
| 考试安排 | 考试部分 | Exam Arrangement | 类基本职能 |
| 参与考试学生名单 | 考试部分 | **Exam Member List** | 因为考试还分为补考、免修考试，因此考试名单于课程名单并不对等 |
| 考试形式申请 | 考试部分 | Course Application | 考试形式申请应包含在课程相关申请中，在申请课程时一并进行 |
| 免考、缓考与免修考试的申请 | 考试部分 | **Special Case Application** | 关于免考与缓考行为的申请在之前涉及的类中并未包含 |
| 学生成绩 | 成绩部分 | Score Record | 类基本职能 |
| 学生成绩单 | 成绩部分 | Report Card | 类基本职能 |
| 学生选课记录 | 选课部分 | Course Selection Record | 类基本职能 |
| 学生学籍信息 | 学籍部分 | Student | 类基本制能 |
| 学生培养方案 | 学籍部分 | Education Program | 类基本制能 |

通过信息专家模式，我们发现我们在领域模型中设计的对象无法将全部的信息表示，因此，在了解缺失信息的基础上，我们新增了一些类，以弥补信息表示不全的缺点。

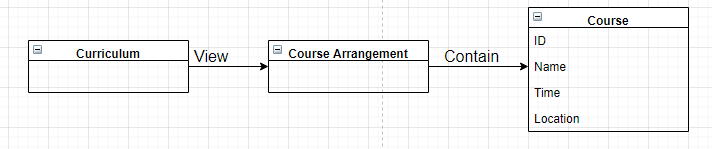
1. 在用例上，教师先应提交开课申请，开课申请中应包含了课程名字、主要内容等基本信息，同时也应包含课时安排、考核安排这样与具体学期相关的信息，这样的信息在之前所设计的类中并不能完整地表达。因此我们新设计了 “Course Application” 这各类充当这部分信息恶信息专家。
2. 在之前的领域模型上，我们为课程设计了与课学生名单这一对象，我们发现，对于考试，并不能完全将这一名单沿用，因为考试除正常期末考试外，还有补考与免修考试，这一类考试的参与者名单于选课学生名单并不完全对等，因此，我们为考试专门设计了一个参与考试人员名单类，以充当这一部分信息的信息专家。
3. 同样是在考试上，我们发现免考、缓考、免修考试等各类考试申请应保存在教务选课系统中，但之前所设计的类并不能认知到这一部分信息，因此我们专门设计了一个 “Special Case Application” 这一申请类，充当这一部分内容的信息专家。

低耦合



类设计中关于课程的几个关键部分疑似存在耦合性问题，但实际上这只是 “Curriculum” 与 “Course Assangement” 之间的之间的关系表达不恰当造成的。

“Curriculum”实际上相当于“Course Assangement”类的一个视图，对其中的信息进行筛选，然后选择性地呈现，因此实际上所有的类都耦合于“Course”类，故可以将几个类之间的排列关系进行调整，正确的表示方法如下所示：



控制器

控制器是UI层之上的第一个对象，主要负责接收和处理系统操作信息。

在之前的领域模型设计中，主要是针对用例场景中出现的实体进行类的设计，而那些用户操作的处理，几乎没有设计，因此我们需要引入一些“控制器”类，负责对用户的操作进行处理。

一般而言，控制器有两种选择，要么是表示整个“系统”、“根对象”的类，要么就是专门的系统事件的接收者或处理者。在教务选课系统中，教务系统本身承担着大量类的创建职责，因此我们选择为事件新建专门的事件处理类。

|  |  |  |
| --- | --- | --- |
| 控制器 | 类名 | 说明 |
| 学生选课操作 | **Course Selection Handler** | 包括学生进行选课、退课、补选操作 |
| 老师提交课程相关申请操作 | **Course Application Handler** | 包含对课程内容的首次提交以及之后的补交 |
| 学生提交考试、成绩相关申请操作 | **Special Case Handler** | 包含学生缓考、免考、免修、成绩复查等申请 |
| 教务员管理课程操作 | **Course Arrangement Handler** | 包含对于课程内容的增删改查 |
| 教务员管理考试操作 | **Exam Arrangement Handler** | 包含对于考试内容的增删改查 |
| 教务员管理学籍操作 | **Student Info Handler** | 包含对于学生个人信息的增删改查 |

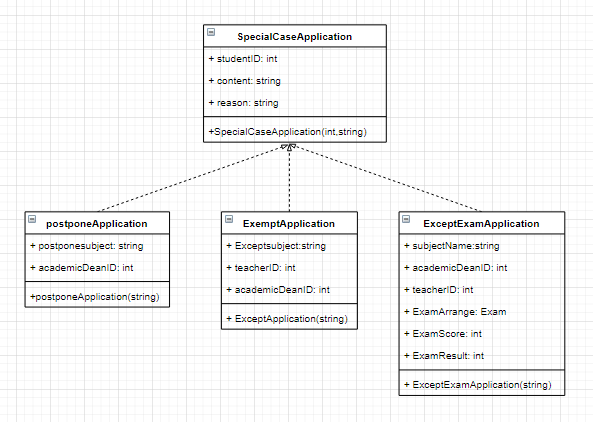
而对于一些“显示”任务，如学生查询课表或成绩单，则应直接由GUI对象负责，因此我们在类的设计中忽略关于这一类任务的描述。

通过这些新增的类，我们可以更加完备地覆盖用例中的场景，使得用例的设计与现实情况更加符合，用户需求得到满足。

多态

多态是一种基本设计原则，用来设计系统如何组织以处理类似真实世界中的变化。基于多态分配职责的设计能够被简便地拓展以处理新的变化。

在之前的类设计中，有一些类实际上具有较广的职责范围，我们考虑可以通过继承与多态的思想将这些类进一步具体化，以实现软件设计的实际要求



|  |
| --- |
| 基类：Special Case Application  说明：该类主要负责储存学生特殊情况的申请，包括缓考、免修、进行免修考试等等  子类：  Postpone Application：主要负责储存学生缓考申请信息  Exempt Application：主要负责储存学生免修申请信息  Exempt Exam Application：主要负责学生免修考试申请相关信息 |

而对于一些其他也可以继续细分的类，例如课表我们可以分成学生课表与老师课表，考试我们可以分成一般考试、补考、免修考试，因为这样细分形成的子类之间并没有太大的差别，因此我们不再额外对这些类进行子类的设计。