Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №4.

«Проектирование объектной модели»

Выполнил: студент группы К4113с Тимошкина Анна Викторовна

Проверил: к. т. н., доцент ФИКТ   
Осипов Никита Алексеевич

Санкт-Петербург

2020 г.

Цель:

Проектирование и разработка объектной модели.

Задачи:

1. Изучить основы разработки объектных моделей с использованием шаблонов GRASP для распределения обязанностей между классами;
2. Построить UML-диаграммы последовательностей и классов.

Ход работы:

Краткое описание первичных классов разрабатываемой системы:

1. ***DataReadingSystem*** – система считывания данных, которая располагается в терминале и производит считывание данных со смарт-карты пользователя *readCard()*, путем его идентификации в базе данных предприятия. Полученная информация передается *системе учета времени*.
2. ***TimeTrackingSystem*** – система учета времени, которая производит учет всех находящихся на работе сотрудников в поле класса *List< Employee > employees*, фиксирует их время прихода *fixComing()* и ухода *fixLeaving()*, в последствии отправляя данные базе данных предприятия *saveWorking()*.
3. ***Employee*** – сотрудник предприятия. В данном классе хранится информация о самом сотруднике *Card card*, его должности *typePosition*, а также подробного отчета о рабочем сеансе *Work work*. В этом классе определен конструктор, заполняющий поля *card*: *Employee(Card card).*
4. ***ProductionEmployee*** – сотрудник производственного цеха, дочерний класс класса *Employee*. Дополняет базовый конструктор, заполняя поле *work*, где задает правила работы *specialWorkRule* согласно трудовому договору сотрудников производственного цеха.
5. ***OfficeEmployee*** – сотрудник офисного отдела, дочерний класс класса *Employee*. Дополняет базовый конструктор, заполняя поле *work*, где задает правила работы *specialWorkRule* согласно трудовому договору сотрудников офисного отдела.
6. ***Card*** – смарт-карта, хранящая основную информацию о пользователе: его идентификатор *id*, фамилию *surname* и имя *name*.
7. ***Work*** – рабочий сеанс, содержит информацию о времени прихода на работу *timeComing* и времени ухода *timeLeaving*, а также рабочее время сотрудника по категориям: нормированное время работы *rationedTime*, сверхурочное время работы *overtime*, время перерыва *breakTime*, дневное время *dayTime*, вечернее время *eveningTime*, ночное время *nightTime* и правила расчета указанных параметров *specialWorkRule*.
8. ***SpecialWorkRule*** – правила работы сотрудника, они определяют с какой регулярностью должен проводиться перерыв *breakRegularity*, его длительность *breakDuration*, и дневную норму работы *workRate*. Данные показатели зависят от занимаемой должности сотрудника (для данной работы – производственная и офисная) и получаются из БД предприятия.
9. ***WorkRule*** – общие правила работы. Это базовый класс для *SpecialWorkRule*. Он содержит общие правила работы, регламентируемые трудовым кодексом РФ. Они затрагивают расчет дневного, вечернего и ночного времени работы, что в последствии отразится на расчете заработной платы, следовательно, должно быть рассчитано системой.

На вопрос «Где будет происходить расчет отработанного времени?» поможет ответить ***Информационный эксперт***, обязывающий выполнять действие тот класс, который обладает требуемой информацией. В данном случае для расчета общего отработанного времени требуется знание времени прихода и ухода сотрудника. Эта информация есть у класса *Work*, поэтому обязанность расчета общего рабочего времени возлагается на него – *calculateGeneralTime()*.

Для расчета времени работы по разным категориям, помимо времени начала и окончания работы, требуется знание правил расчета, которые гранятся в поле *workRule* класса *Work*, поэтому обязанность расчета данных величин также возлагается на представленный класс - *calculateRationedTime()*. Помимо этого, от класса *SpecialWorkRule* требуется знание правил расчета по категориям – *getWorkRate()*.

На вопрос «Где будут создаваться экземпляры класса Employee?» помогает ответить шаблон ***Создатель***. Поскольку *Система учета времени* содержит список объектов *Employee*, то предлагается выполнять данную задачу ему: *createEmployee(int id)*.

Также, согласно данному шаблону, экземпляры класса *SpecialWorkRule* могут быть созданы как в классе *Employee* (т. к. он содержит требуемую для инициализации информацию), так и в классе *Work* (т. к. он содержит поле типа *SpecialWorkRule*). Чтобы решить, каким из двух вариантов воспользоваться, обратимся к принципам ***Слабой связанности*** и ***Сильной связность***. Согласно этим шаблонам, требуется обеспечить слабую связанность функций разных классов и сильную связность функций внутри класса. Таким образов создание экземпляров *SpecialWorkRule* делегировано объекту *Work*, так как благодаря этому поддерживается более высокая степень зацепления объекта *Employee* (у него нет знания о неиспользуемом в его работе классе).

Следуя шаблону ***Контроллер***, который является обработчиком системных событий, создаем класс *Terminal*, который играет роль физического устройства считывания данных. Данный класс будет промежуточным звеном между интерфейсом пользователя и бизнес-логикой, и будет отрабатывать действия пользователя: прикладывание карты на входе *fixComing(int id)* и прикладывание карты на выходе *fixLeaving(int id)*, порождая требуемые системные действия. При этом класс ***DataReadingSystem*** не выполняет никаких обязанностей (они были перенесены в контроллер), поэтому в дальнейшем будет играть роль интерфейса и переименуется – *Form*.

Диаграмма классов согласно представленным шаблонам расположена в файле *«Диаграмма классов.png»*. Диаграммы последовательности прихода и ухода сотрудников на работу расположены в файлах *«Фиксирование прихода на работы.png»* и *«Фиксирование ухода с работы.png»*.

Вывод:

В ходе выполнения данной лабораторной работы были изучены шаблоны проектирования GRASP, на основании которых происходило проектирование разрабатываемой системы и были построены UML-диаграммы классов и последовательности. Следование изученным шаблонам позволило улучшить систему и оптимизировать ее работу. Например, после применения шаблонов проектирования была убрана связь класса Card с системой считывания, а обязанность создания экземпляров этого класса делегирована агрегирующему его классу Employee.