Объектно-ориентированное программирование

Часть II

Наше состояние и направление движения

- Что должны и уметь знать к этому моменту
  - Основные понятия объектного подхода
  - Базовую теорию и практику ОО-программирования на примере C++
  - Теоретические знания UML и начала ОО-проектирования
- Что должны знать и уметь после этого курса
  - Углубить понимание объектного подхода так, чтобы «мыслить объектами»
  - Развить практику ООП так, чтобы успешно реализовать курсовой проект
  - Развить навыки проектирования на примере того же КП, попрактиковаться в применении UML для этого
- Что НЕ входит в наши планы
  - Углубленное изучение языка C++ и конкретных библиотек и фреймворков

План курса

- Курсовое проектирование
  - Главное событие сезона ☺
  - Экзамена нет – КП является главной задачей курса
  - Полный цикл: OOA -> OOD -> OOP
- Практические занятия
  - Поддержка работы над КП
- Лекционный курс
  - Первая задача – поддержать вас в работе над проектом, особенно в тех стадиях, которые были меньше покрыты первой частью курса (OOA и OOD)
  - Дополнительно дать вам понимание основных идей и подходов, используемых в проектировании ИС.
  - Отсутствие экзамена – не повод пропускать лекции.
    - Допуск к защите КП никто не отменял.

План лекционного курса

1. Вводная лекция (сегодня). Подробнее о КП.
2. OOA и OOD углубленно – на приближенном к КП примере.
3. Порождающие шаблоны GoF.
4. Структурные шаблоны GoF.
5. Поведенческие шаблоны GoF (начало).
6. Поведенческие шаблоны GoF (продолжение).
7. Архитектурные шаблоны (классика)
8. Прикладные шаблоны (начало)
9. Прикладные шаблоны (продолжение)
10. Игровые шаблоны

Цели и задачи КП

- Цели
  - изучение современного подхода к программированию на основе объектно-ориентированной технологии
  - приобретение навыков написания программ на языке с поддержкой ООП на примере написания программы согласно варианту задания.
- Задачи (обязательная программа)
  - выбор варианта задания и языка реализации, детализация поставки задачи;
  - абстрагирование, разработка классов и их иерархии;
  - написание текста (кодирование) разработанных классов на выбранном языке;
  - разработка тестовых примеров;
  - тестирование и отладка программы;
  - разработка программных документов в соответствии с действующими стандартами.
- Дополнительные задачи (**не для всех и не обязательные, по желанию**)
  - Научиться практической работе с инструментарием управления проектом и исходным кодом.
  - Кто продемонстрирует это в работе– смело присылайте мне резюме. :)

Тематика и варианты заданий

- Общая тематика
  - Моделирование систем со сложным характером взаимодействия значительного количества элементов.
- 4 группы заданий
  - Спроектировать и разработать модель **игры** по заданному плану-варианту;
  - Разработать на базе теории **систем массового обслуживания** модель по заданному плану-варианту;
  - Разработать **систему сбора статистической информации** по заданному плану-варианту;
  - Спроектировать и разработать модель **сложной системы** по заданному плану-варианту
- Возможность индивидуальной формулировки задания
  - Возможно (**и очень желательно**) дополнение или изменение исходных условий постановки задачи в целях более качественной реализации проекта.
  - По сути, задания представляют собой лишь канву тех или иных описаний поведения одиночных объектов или их групп.
  - Конечно, задание **согласовывается с преподавателем**.

Требования к КП

- КП может выполняться
  - индивидуально
  - группой студентов 2-3 человека, в зависимости от сложности задания – в **этом случае необходимо согласование с преподавателем**.
- Язык программирования – на выбор
  - Ограничивается требованием **абсолютно полной** поддержки возможностей объектного подхода (JS не годится, например).
  - Рекомендуется C++, C#, Java.
- К защите КП подготовить:
  - программу;
  - пояснительную записку (ПЗ);
  - графические материалы (слайды)?
  - *(дополнительно) репозиторий исходного кода и рабочие задачи в таск-трекере*
- Критерии оценки
  - В первую очередь оценивается применение ОО-подхода – модели и код
    - Формальный критерий – иерархия из не менее 5 объектов и не менее 3 уровней.
  - Внешняя красота: графика/музыка (для игр) и прочий UI/UX вторичны

Порядок выполнения КП

- Этапы работы над КП:
  1. Постановка задачи.
  2. OOA. Описание объектов предметной области, классов и их атрибутов. Построение жизненного цикла системы (программы и объектов)
  3. OOD. Разработка иерархии классов в терминах программной среды.
  4. OOP. Реализация системы.
  5. Тех. Документация. Пользовательский интерфейс, описание работы программы.
  6. Анализ качества разработанной системы. Критерии качества и тестовые сценарии.
- Часть этапов итеративна
  - в календарном плане для таких пунктов отражается время первой итерации
- Во время защиты КП студент должен
  - сделать доклад,
  - продемонстрировать работу программы на персональном компьютере,
  - ответить на вопросы, относящиеся как к работе программы, так и к основным теоретическим положениям дисциплины ООП.

Этап постановки задачи

- Определение требований к разрабатываемой системе
  - Функциональные требования (высокоуровневая постановка задачи)
    - Цель разработки, смысл системы.
    - Основные функции системы и ее жизненный цикл.
    - Общее описание моделируемой предметной области.
    - Основные варианты использования системы.
  - Определение требований к платформе и программно-аппаратным ресурсам приложения
    - Выбор языка программирования и сопутствующего инструментария
- Согласование с преподавателем и документирование принятых решений
  - Требования к системе
  - Календарный план реализации

Этап ООА

- Построение модели предметной области
  - Статической
    - Выделение объектов и классов предметной области
    - Выделение связей между ними
  - Динамической
    - Жизненный цикл объектов
    - Взаимодействие между объектами
- Все это в контексте **предметной области**, а не программы!
- Итеративность возможна, когда меняется предметная область, но это происходит не часто и не быстро (не в случае КП).

Этап OOD

- Построение модели проектируемой системы
  - Статической и Динамической
    - Классы и объекты
    - Их структура и связи
    - Жизненный цикл
    - Сценарии взаимодействия
  - Логической и Физической
    - Разделение на системы на логические компоненты и модули
    - Определение необходимых физических компонент
    - Размещение логических компонент на физических
- Все это уже в контексте **разрабатываемой системы**
- Итеративность сильно выражена
  - Спроектировать – реализовать – проверить – обдумать – повторить
  - Артефакты OOD эволюционируют вместе с кодом

ООР. Реализация системы.

- Реализация программного кода разрабатываемой системы
  - Реализация непосредственно кода системы
  - *(дополнительно) Реализация модульных и интеграционных тестов кода*
  - *(дополнительно) Управление исходным кодом команды в репозитории (ветвления, слияния, и т.п.)*
  - *(дополнительно) Автоматизация процесса построения продукта (построение последней версии продукта по актуальным исходникам в репозитории, автозапуск тестов и т.п.)*
  - *(дополнительно) отражение работы команды в таск-трекере*
- Итеративность сильно выражена
  - Спроектировать – реализовать – проверить – обдумать – повторить
  - Реализация и проектирование в современной практике обычно идут вместе

Этап подготовки документации

- ПЗ к курсовому проекту должна содержать (детали в методичке)
  - титульный лист;
  - техническое задание (на специальном бланке);
  - аннотация;
  - содержание;
  - перечень сокращенных и условных обозначений (при необходимости);
  - введение;
  - список исполнителей (при необходимости);
  - постановка задачи;
  - проектное решение;
  - программная реализация;
  - выводы;
  - перечень ссылок;
  - приложения (обязательное приложение — текст программы).
- Итеративность – зависит от вида документации
  - Часть этих документов вы разработаете до начала работ по реализации
  - Часть документов будет постоянно дорабатываться в ходе итераций проектирования и реализации
  - Часть будет оформлена после завершения основных работ

Этап анализа качества

- Подготовка тестовых сценариев
  - Анализ вариантов использования системы
  - Описание стандартных сценариев исполнения вариантов использования
  - *(дополнительно) Описание нестандартных сценариев исполнения вариантов использования*
- Исполнение тестовых сценариев
  - Описание либо демонстрация результатов исполнения тестового сценария
  - Документирование результатов – в случае КП, для финальной проверки
- Итеративность
  - Подготовка сценариев обычно не итеративна (при изменении требований)
  - Исполнение сценариев чаще всего итеративно (Спроектировать – реализовать – **проверить** – обдумать – повторить)

Варианты заданий

- Игры
  - Стратегия
  - Шутер
  - Головоломка
- Системы массового обслуживания
  - Почтовые сервера
  - Маршрутное такси
- Системы сбора статистической информации
  - Мониторинг погоды
  - Компьютерные сети
- Модели сложных систем
  - Океан
  - Улей
  - Муравейник

Игры. Стратегия. Пример описания.

- В системе моделируется противоборство двух колоний. Колония состоит из строений и юнитов (живых существ).
  - Юнит обладает набором характеристик: на сколько клеток игрового поля он «видит», уровень «здоровья», сколько «здоровья» он теряет во время боя за один удар и др.
  - Юниты разделяются на два типа: строящие и воюющие. Строители наносят врагу незначительный урон и обладают низкой жизнестойкостью. Воины наносят более существенный урон и более защищены. Урон, наносимый воином, меняется случайно в незначительных пределах. Юнит обладает характерным поведением: строитель – если видит врага, – убегает, воин вступает в бой и т.д.
  - У строений есть свои характеристики: размер, количество ударов, после которых строение рухнет и т.д. Строения могут быть двух видов: дома юнитов и защитные укрепления. Каждый дом обеспечивает жизнедеятельность N юнитов. Если вместимости домов не хватает, новый юнит не может быть порожден. Если дом разрушен противоборствующей стороной, то юниты пострадавшей стороны начинают голодать.
  - Дома можно строить без ограничений, на строительство дома нужно достаточно много времени. Юнит создается быстрее, количество ограничено вместительностью строений.
  - Игра происходит на игровом поле. Юнит занимает одно место, здание – несколько. Юнит может перейти на соседнее место, если оно не занято зданием или другим юнитом. Игра может проходить в двух режимах: из некоторого начального положения без участия человека (на начало игры уже созданы все здания и юниты) и с участием человека (он управляет юнитами).

Игры. Шутер. Пример описания.

- Моделируется противостояние игрока и компьютера. Суть: главный герой (ГГ) противостоит множеству вражеских юнитов.
  - Характеристики ГГ: количество жизней, величина урона, защита, скорость и др. ГГ может собирать некоторые бонусы, выпадающие случайным образом или из побежденных противников. Возможна смена оружия, способностей или характеристик героя со временем или после побед над врагами.
  - Юниты имеют характеристики, сходные с ГГ. Юниты делятся на несколько типов, каждый из которых отличается особыми параметрами и способностями.
  - Цель игрока – сохранить ГГ от столкновений с юнитами или уничтожить наибольшее их количество. Цель юнитов уничтожение ГГ.
  - Игра завершается победой ГГ над всеми юнитами, управляемыми компьютером, либо достижением максимального количества оценочных баллов игры (опыт, очки, время и др), либо смертью ГГ.

Игры. Головоломка. Пример описания.

- Существует множество объектов различного вида и типа,которые взаимодействуют между собой.
- Задача: перемещая и комбинируя определенным образом объекты, необходимо добиться заданного графически ожидаемого результата. Достижение результата представляется возможным в случае единственно правильного расположения и взаимодействия используемого множества объектов.

СМО. Почтовые сервера. Пример описания.

- Рассматривается работа системы, состоящей из почтового сервера и его клиентов.
- Сервер связан с клиентами каналом связи ограниченной пропускной способностью, и не может обслуживать одновременно больше (x<n) клиентов. Пропускная способность канала X бод.
- В исходном состоянии сервер свободен, ни одно соединение с клиентом не установлено. После того, как клиент запросит соединение, сервер проверяет, может ли он предоставить соединение. Если соединение устанавливается, клиент передает серверу общее количество писем, их заголовки и размер. Получив заголовок очередного письма, сервер проверяет правильность адреса (на это уходит T секунд), если адрес правильный, начинается передача тела письма. В случае, если адрес неправильный – клиент получает сообщение об ошибке и соединение с ним закрывается. Время, необходимое для передачи тела письма зависит от пропускной способности канала и размера письма. Если к серверу подключены несколько клиентов, то пропускная способность делится между ними поровну. Если соединение с сервером не может быть установлено, клиент ждет некоторое время, после чего пытается повторить передачу.
- Письма классифицируются следующим образом: 1 вид –малые (до A Кб), 2 вид – средние (до B Кб), 3 вид – большие (все остальные). У каждого клиента есть уникальное распределение генерирующих писем, т.е. сколько процентов от общего количества генерируется первого вида, сколько второго, и т.д. Общий поток писем характеризуется интенсивностью - количеством писем в единицу времени. Каждый клиент обладает своей интенсивностью.
- Необходимо промоделировать работу системы в течение задаваемого количества времени, определить следующие характеристики:
  - процент занятости сервера,
  - количество отказов каждого клиента,
  - среднюю загруженность канала,
  - среднее время передачи письма.

СМО. Маршрутное такси. Пример описания.

- Смоделировать работу сети маршрутных такси.
  - По маршруту движется не более N маршрутных такси (МТ). Каждое МТ имеет 18 посадочных мест. Водители МТ не нарушают ПДД и не берут дополнительных пассажиров.
  - На маршруте имеется L остановок, на каждой из которых случайным образом генерируется некоторое количество пассажиров. Количество пассажиров, скопившихся на остановке, пропорционально времени ожидания. Для каждого пассажира случайным образом генерируется число остановок P, которое он хочет проехать P<=2*L/3.
  - Время посадки/высадки 1 пассажира занимает 7 секунд. Время стоянки на остановке, если есть свободные места, 10 секунд, в противном случае МТ не останавливается. МТ движется по маршруту со средней скоростью 40 км/ч (скорость с учетом светофоров, пробок и т.п.). Расстояние между всеми остановками заранее известно.
- Необходимо промоделировать работу системы в течение задаваемого количества времени, определить следующие характеристики:
  - процент занятости всех МТ на маршруте,
  - среднее число пропускаемых МТ остановок за 1 прохождение маршрута,
  - среднюю загруженность МТ,
  - среднее время проезда пассажира.

Мониторинг погоды. Пример описания.

- Разработать систему сбора данных для мониторинга погоды.
  - Система состоит из датчиков скорости и направления ветра, температуры, давления, влажности воздуха и др. Значения каждой величины в момент снятия показаний генерируется случайным образом из указанного диапазона с учетом общих закономерностей (например, изменение температуры в течение суток).
  - Система должна снимать показания каждые 5 сек., усреднять по каждому часу, вычислять относительные изменения измеряемых величин, выводить результаты на экран, выводить данные за последние 24 часа, сохранять результаты работы в файле.
  - Программа должна иметь: графический интерфейс и возможность указания начальных условий моделируемого процесса.
  - Количество и тип каждого из датчиков, диапазон значений в которых изменяются измеряемые величины, интервал времени моделирования в часах и коэффициент соотношения реального времени со временем системы разрабатываются студентом и согласовываются с преподавателем.

Компьютерные сети. Пример описания.

- Сеть состоит из серверов (видео, аудио, файлообменников и др.), машин пользователей, оргтехники (принтер, сканер, копир, факс и др). Сеть имеет некую топологию (звезда, кольцо, дерево).
- В системе существует несколько типов сообщений: запросы на обслуживание, ответы от обслуживающей техники, системные сообщения, сообщения включения/выключения серверов, запросы на получение данных и пакеты передаваемых данных и др. Машины пользователей генерируют запросы к серверам и оргтехнике. Запросы генерируются случайно с некоторой вероятностью их появления (вероятность появления запроса, а также распределение вероятностей появления по типам запросов разрабатываются студентом). Оргтехника способна обрабатывать только один запрос в единицу времени, на обработку запроса тратится N единиц времени.
- Существует некоторая вероятность отказа техники и некоторая вероятность отказа в обслуживании запроса. Результаты работы оргтехники сообщаются в ответном сообщении. Сервера могут обслуживать одновременно несколько запросов от пользователей. Запрос к серверу может быть «на получение информации», «на отправку информации», «на поиск информации» и др. Запросы «на информацию» инициализируют серию сообщений между сервером и клиентом. Если техника либо соединение вышли из строя, то их восстановление проводится ремонтной бригадой с заданной производительностью (то есть определенное время). Время ремонта определяется коэффициентом сложности оборудования.
- Программа должна отображать топологию сети, выводить статистические данные по запросам в сети и работе оборудования каждый час моделируемого времени системы, выводить во внешний файл средние значения за сутки.

Модели систем. Океан. Пример описания.

- В первобытном океане существует две формы жизни: хищники и существа, питающиеся растительной пищей.
- Океан вырабатывает растительную пищу следующим образом: к количеству вещества на предыдущем шаге (x) добавляется некоторая часть (например, 0.5x), если итоговая масса не превышает некоторого предела – порога (Y), т.к. океан не может вырастить больше некоторого количества растительной пищи на единицу поверхности.
- Потребление растительной пищи происходит следующим образом: если клетка поля занята «травоядным» существом, то оно потребляет 0.1x за один ход, 0.2x за второй, 0.3x за третий и т.д., т.к. в благоприятных условиях популяция травоядных возрастает. Размеры колонии травоядных увеличиваются на 1 ряд с каждой стороны за ход, если колония не голодает.
- Хищники, в отличие от травоядных, индивидуальны. Один хищник занимает целое поле и за 1 ход уничтожает половину его содержимого. Хищник находится в постоянном движении. В случае, если хищник в течение 5 ходов не «голодал» (получал не менее 50% от Y пищи), он раздваивается (делится). Два хищника не могут занимать одну клетку. В случае, если хищник не может перейти на клетку, не занятую другим хищником, один из них погибает (более «сытый»).
- Хищник живет 30 ходов, «голодающий» хищник в течение трёх ходов погибает.

Модели систем. Улей. Пример описания.

- Разработать программную систему, моделирующую жизнь улья.
- 3 вида пчел — собиратели, строители и матка. Каждый из этих видов отвечает за выполнение определенных действий. У каждого вида пчел есть уровень жизни. Каждая пчела употребляет определенное количество меда в единицу времени, в зависимости от состояния в котором находится.
- Собиратели – отвечают за облет окружающей территории и сбор нектара. Когда Собиратель покидает улей, он перестает есть мед и начинает терять уровень жизненных сил. Интенсивность потери им жизненных сил зависит от того, какое количество нектара он несет с собой. Если он ничего не несет – 1 единица жизни за 1 ход. Каждая единица веса нектара увеличивает расход жизненных сил на 0.2 за 1 ход. Таким образом, если собиратель несет с собой 5 единиц нектара – он теряет 1+0.2*5=2 единицы жизненных сил за 1 ход. После возвращения в улей он начинает есть с интенсивностью в 4 раза больше, чем строитель в спокойном состоянии. После пополнения сил Собиратель может вновь отправиться за нектаром.
- Строители — отвечают за строительство улья и преобразование нектара в мед. Например, 1 строитель за 1 ход может преобразовать 10 единиц нектара в 7 единиц меда. Строительство дополнительных сот улья занимает определенное длительное время. Строитель может находиться в одном из 3-х состояний. Спокойное — малое потребление меда. Преобразование нектара в мед — удвоенное потребление меда. Строительство — утроенное потребление меда.
- Матка — отвечает за порождение новых членов колонии. При создании нового улья Матка порождается из Строителя (достаточно длительный процесс). В одном улье не может быть более одной Матки. В спокойном состоянии Матка потребляет столько же меда, сколько Строитель в спокойном состоянии. При создании новой пчелы она ест в 5 раз больше.
- Цветы, дающие нектар, появляются случайным образом на игровом поле. Длительность жизни цветка ограничена и не зависит от того, собирают ли с него нектар или нет.
- На обеспечение жизнедеятельности каждого насекомого колонии должно быть определенное количество сот в улье. Перед порождением нового члена колонии должна осуществляться проверка на возможность его жизнеобеспечения в улье. Если такой возможности нет, то улей должен быть достроен определенным количеством сот. Улей не может быть больше определенного размера. Если колония не голодает, может расти, но улей переполнен, то 30 процентов колонии покидают улей и образуют новый в другой части игрового поля. При этом они должны взять с собой определенное (достаточно большое) количество меда.
- Если в улье не хватает меда, то жители начинают голодать. Во время голодания скорость потери жизненных сил увеличивается. Работающие пчелы теряют уровень жизненных сил быстрее. Как только уровень жизненных сил пчелы становится <=0, пчела погибает.

Модели систем. Муравейник. Пример описания.

- Разработать программную систему, моделирующую жизнь муравейника, основываясь на принципах жизни насекомых и иерархии особей: личинка, матка, рабочий муравей, воин.
- Матка порождает личинки, которые в свою очередь пре- вращаются в рабочих муравьев. Матка может породить новую личинку только при достаточном количестве пищи и места в муравейнике.
- Воины охотятся, добывая еду муравейнику, и защищают муравейник от опасностей (атак других насекомых).
- Муравейник строится в виде центральной шахты и боковых ответвлений. Если длина бокового ответвления превышает определенные пределы, то должна быть построена очередная шахта. Строительство боковых тоннелей процесс, требующий не очень много еды. Данный процесс занимает много времени.
- На создание нового муравья требуется много времени и еды. Еда - это насекомые других видов (несколько разновидностей) отличающихся силой, размерами и количеством пищи, в которое они могут превратиться, если будут убиты муравьями. Соответственно и количество муравьев, необходимых для убийства насекомого и доставку его в муравейник разнится в зависимости от вида насекомого. Еда появляется случайным образом в произвольных точках игрового поля и может самостоятельно перемещаться по нему, в течение произвольного времени или пока не будет убита.
- Поглощение муравьями еды внутри муравейника и вне его аналогично потреблению пчелами меда в задании про улей

Итого по смыслу заданий

- Необходимо выбрать такую задачу, которая бы продемонстрировала
  - множество разнородных объектов, взаимодействующих друг с другом
  - как композиционные, так и иерархические связи между классами объектов
- Идеально подходят различного рода модели сложных систем
  - Игры – как модель с интерактивным участником.
- Что НЕ надо делать (из опыта студентов прошлых лет)
  - Не надо делать скучные бизнес-приложения с банальными CRUD-сценариями – всевозможные БД, реестры, и прочий «кровавый энтерпрайз» вы еще успеете возненавидеть за свою карьеру разработчика ☺
  - Не надо делать простые решения, не моделирующие множество объектов (из игр в т.ч., текстовые квесты, игры в стиле «угадай картинку» и прочее непотребство, где ООП не проявляет себя в полной мере)
- Начинайте продумывать задачу своей мечты уже сегодня, чтобы на следующее занятие прийти уже с пониманием этой задачи.