Архитектура и шаблоны проектирования

Курс для разработчиков, которые хотят изучить основные паттерны проектирования и научиться применять их, находить им замену в сложных ситуация и научиться мыслить, как архитектор программного обеспечения

Длительность курса: 130 академических часов

1 Проблема сложности и способы ее решения.

1 **Проблема сложности, ее разновидности и их связь с архитектурой ПО.**

Цели занятия:

- объяснить влияние проблем сложности архитектуру, чтобы иметь объективный инструмент оценки архитектурных решений,
- перечислить самые распространенные проблемы сложности, чтобы иметь возможность учитывать их при построении архитектуры приложения.

Краткое содержание:

I. Проблема сложности.
- понятие архитектуры ПО;
- проблема сложности и ее влияние на архитектуру ПО;
- виды сложности: алгоритмическая сложность, скорость разработки ПО, сложность тестирования, информационная сложность, Time To Market и д.р.

---

2 **Архитектура, архитектурные стили и виды**

Цели занятия:

объяснить понятие архитектуры;
рассмотреть архитектурные стили.

Краткое содержание:

раскроем понятие архитектурной системы;
рассмотрим стили:
- монолитное приложение;
- многоуровневая архитектура;
- бессерверный архитектура;
- система, управляемая событиями;
- сервис-ориентированная система;
- микросервисная архитектура.

части архитектуры: системы, компоненты, интерфейсы, потоки данных;
архитектурные виды (модульные, компонентные, информационные, размещения);
понятие архитектурного шаблона.

2 Сложность тестирования

1 Модульные тесты // ДЗ

Цели занятия:

писать модульные тесты.

Краткое содержание:

история возникновения идеи модульных тестов;
определение множества исчерпывающих тестов;
TDD.

Домашние задания

1 Разработать набор модульных тестов

Цель: Научиться писать модульные тесты по TDD, так как модульные тесты является важной частью концепции Time To Market и часто применяется в современных проектах.

1. Создать проект на gitlab/github.
2. Настроить CI, чтобы можно было собирать проект и прогонять тесты.

Необходимо реализовать операцию нахождения квадратного уравнения. Предположим, что эта операция описывается следующей функцией с поправкой на конкретный язык программирования. В ООП языках эта функция реализуется в виде метода класса.

solve(double a, double b, double c): double[]

здесь a, b, c - коэффициенты квадратного уравнения, функция возвращает список корней квадратного уравнения.

3. Написать тест, который проверяет, что для уравнения x^2+1 = 0 корней нет (возвращается пустой массив)

4. Написать минимальную реализацию функции solve, которая удовлетворяет данному тесту.

5. Написать тест, который проверяет, что для уравнения x^2-1 = 0 есть два корня кратности 1

(x1=1, x2=-1)

6. Написать минимальную реализацию функции solve, которая удовлетворяет тесту из п.5.

7. Написать тест, который проверяет, что для уравнения x^2+2x+1 = 0 есть один корень кратности 2 (x1= x2 = -1).

8. Написать минимальную реализацию функции solve, которая удовлетворяет тесту из п.7.

9. Написать тест, который проверяет, что коэффициент a не может быть равен 0. В этом случае solve выбрасывает исключение. Примечание. Учесть, что a имеет тип double и сравнивать с 0 через == нельзя.

10. Написать минимальную реализацию функции solve, которая удовлетворяет тесту из п.9.

11. С учетом того, что дискриминант тоже нельзя сравнивать с 0 через знак равенства, подобрать такие коэффициенты квадратного уравнения для случая одного корня кратности два, чтобы дискриминант был отличный от нуля, но меньше заданного эпсилон. Эти коэффициенты должны заменить коэффициенты в тесте из п. 7.

12. При необходимости поправить реализацию квадратного уравнения.

13. Посмотреть какие еще значения могут принимать числа типа double, кроме числовых и написать тест с их использованием на все коэффициенты. solve должен выбрасывать исключение.

14. Написать минимальную реализацию функции solve, которая удовлетворяет тесту из п.13.

15. Сделать merge request/pull request и ссылку на него указать при сдаче ДЗ.

3 Скорость разработки ПО.

1 **Абстрагирование. Схема применения SOLID принципов.**

Цели занятия:

- сформулировать математическое определение абстрагирования, чтобы иметь возможность применять абстрагирование при разработке ПО;
- объяснить роль обобщения при построении абстрагирований в 4D, чтобы уметь строить повторно используемые абстракции;
- описывать абстракции, устойчивые к изменениям, чтобы писать код, соответствующий OCP принципу.

Краткое содержание:

I. Абстрагирование.
- недостаток определения Буча;
- математическое определение абстрагирования;
- примеры применения абстрагирования;
- 4D моделирование;
- обобщение и частично-определенное абстрагирование;
- повторное использование и неизменяемые абстракции.

II. Использование принципов SOLID для написания кода устойчивого к изменениям;
the Open-Closed Principle;
the Single Repsonsibility Principle;
the Dependency Inversion Principle.

---

2 **Определение абстракций, устойчивых к изменениям требований. // ДЗ**

Цели занятия:

вносить исправления в абстракции, чтобы повысить их устойчивость к изменениям.

Краткое содержание:

использование принципов SOLID для написания кода устойчивого к изменениям;
the Liskov Substitution Principle;
the Interface Segregation Principle.

Домашние задания

1 Движение игровых объектов по полю.

Цель: Выработка навыка применения SOLID принципов на примере игры "Космическая битва".

В результате выполнения ДЗ будет получен код, отвечающий за движение объектов по игровому полю, устойчивый к появлению новых игровых объектов и дополнительных ограничений, накладываемых на это движение.

В далекой звездной системе встретились две флотилии космических кораблей. Корабли могут передвигаться по всему пространству звездной системы по прямой, поворачиваться против и по часовой стрелке, стрелять фотонными торпедами. Попадание фотонной торпеды в корабль выводит его из строя.
От каждой флотилии в сражении принимают участие по три космических корабля.
Победу в битве одерживает та флотилия, которая первой выведет из строя все корабли соперника.

Управление флотилиями осуществляется игрокам компьютерными программами (то есть не с клавиатуры).

Концептуально игра состоит из трех подсистем:
1. Игровой сервер, где реализуется вся игровая логика.
2. Player - консольное приложение, на котором отображается конкретная битва.
3. Агент - приложение, которое запускает программу управления танками от имени игрока и отправляет управляющие команды на игровой сервер.

Реализовать движение объектов на игровом поле в рамках подсистемы Игровой сервер.

3 **Общие шаблоны распределения ответственностей**

Цели занятия:

проанализировать функциональное разделение функционала;
рассмотреть 9 шаблонов GRASP (они понадобятся для пояснения более широких понятий, таких как микросервисная архитектура).

Краткое содержание:

информационный эксперт;
создатель;
контроллер;
слабое зацепление;
высокая связность;
полиморфизм;
чистая выдумка;
посредник;
устойчивость к изменениям.

---

4 **SOLID и исключения // ДЗ**

Цели занятия:

особенности обработки исключений с точки зрения SOLID принципов.

Краткое содержание:

какие исключения и где их стоит обрабатывать;
стратегии обработки исключений.

Домашние задания

1 Механизм обработки исключений в игре "Космическая битва"

Цель: Научится писать различные стратегии обработки исключений так, чтобы соответствующий блок try-catch не приходилось модифицировать каждый раз, когда возникает потребность в обработке исключительной ситуации по-новому.

Предположим, что все команды находятся в некоторой очереди. Обработка очереди заключается в чтении очередной команды и головы очереди и вызова метода Execute извлеченной команды. Метод Execute() может выбросить любое произвольное исключение.

1. Обернуть вызов Команды в блок try-catch.
2. Обработчик catch должен перехватывать только самое базовое исключение.
3. Есть множество различных обработчиков исключений. Выбор подходящего обработчика исключения делается на основе экземпляра перехваченного исключения и команды, которая выбросила исключение.
4. Реализовать Команду, которая записывает информацию о выброшенном исключении в лог.
5. Реализовать обработчик исключения, который ставит Команду, пишущую в лог в очередь Команд.
6. Реализовать Команду, которая повторяет Команду, выбросившую исключение.
7. Реализовать обработчик исключения, который ставит в очередь Команду - повторитель команды, выбросившей исключение.
8. С помощью Команд из пункта 4 и пункта 6 реализовать следующую обработку исключений: при первом выбросе исключения повторить команду, при повторном выбросе исключения записать информацию в лог.
9. Реализовать стратегию обработки исключения - повторить два раза, потом записать в лог.
Указание: создать новую команду, точно такую же как в пункте 6. Тип этой команды будет показывать, что Команду не удалось выполнить два раза.

---

5 **Команда // ДЗ**

Цели занятия:

применять шаблон;
построить процесс выполнения задачи с использованием шаблона "Команда".

Краткое содержание:

хороший результат командной работы зависит от слаженности работы каждого;

команда выполняет некоторые действия и, в то же время, команда есть часть бизнес процесса, определяющего последовательность выполнения действий;
снова нужен интерфейс, какие операции у него будут?
как здесь может участвовать "Состояние"?

Домашние задания

1 Макрокоманды

Цель: Цель: Научиться обрабатывать ситуации с точки зрения SOLID, когда требуется уточнить существующее поведение без модификации существующего кода.

Предположим, что у нас уже написаны команды MoveCommand и RotateCommand. Теперь возникло новое требование: пользователи в игре могут устанавливать правило - во время движение расходуется топливо, двигаться можно только при наличии топлива.

Реализовать новую возможность можно введя две новые команды.
CheckFuelCommand и BurnFuelCommand.
CheckFuelCommand проверяет, что топлива достаточно, если нет, то выбрасывает исключение CommandException.
BurnFuelCommand уменьшает количество топлива на скорость расхода топлива.

После этого мы можем три команды выстроить в цепочку.
CheckFuelCommand MoveCommand BurnFuelCommand

Чтобы это было прозрачно для пользователя реализуем Макрокоманду - специальную разновидность команды, которая в конструкторе принимает массив команда,
а методе execute их все последовательно выполняет.

При повороте движущегося объекта меняется вектор мгновенной скорости. Напишите команду, которая модифицирует вектор мгновенной

скорости, в случае поворота.
Постройте цепочку команд для поворота.

1. Реализовать класс CheckFuelComamnd и тесты к нему.
2. Реализовать класс BurnFuelCommand и тесты к нему.
3. Реализовать простейшую макрокоманду и тесты к ней. Здесь простейшая - это значит, что при выбросе исключения вся последовательность команд приостанавливает свое выполнение, а макрокоманда выбрасывает CommandException.
4. Реализовать команду движения по прямой с расходом топлива, используя команды с предыдущих шагов.
5. Реализовать команду для модификации вектора мгновенной скорости при повороте. Необходимо учесть, что не каждый разворачивающийся объект движется.
6. Реализовать команду поворота, которая еще и меняет вектор мгновенной скорости, если есть.

---

6 **Расширяемая фабрика и IoC // ДЗ**

Цели занятия:

узнать и научиться применять шаблон.

Краткое содержание:

проблемы сильносвязанных структур;
от конкретной реализации к интерфейсам;
инкапсуляция создания объектов;
шаблон "Фабричный метод";
от Фабрики к абстракциям. "Абстрактная фабрика";
разбор примера.

Домашние задания

1 Реализация выбора подходящего метода сортировки (выбором, вставки, слиянием) набора данных с использованием абстрактной фабрики и описание применения шаблона в проекте

Цель: Получите навык работы с абстрактной фабрикой.

Данные задаются в файле. Результат также помещается в файл.
1. Выбрать массив размером 50 элементов.
2. Создать программу, которая в качестве входного параметра получает вариант сортировки (выбором, вставки, слиянием), имя файла со входным набором данных и имя файла с выходными данными.
3. Реализовать в программе абстрактную фабрику и конкретные фабрики, отвечающие за каждый вариант сортировки как продукты.
4. Программа записывает результаты в выходной файл данных. В содержании в пишется тип сортировки и результаты.
5. Если потребуется использовать абстрактную фабрику или фабричный метод в проектной работе, предоставить описание в текстовом файле в GitHub репозитории где конкретно и в какой роли используется этот шаблон.
6. нарисовать диаграмму классов.
Срок 2 недели от занятия
ДЗ сдается в виде ссылки на GitHub репозиторий с проектом.
По вопросам обращаться в Slack к студентам, преподавателям и наставникам в канал группы

---

7 **Чистый код и рефакторинг**

Цели занятия:

видеть недостатки кода, смогут корректировать его, превращая в корректный удобный код.

Краткое содержание:

понятия "чистого кода";
корректное представление;
обработка ошибок, тестирование;
"воняющий" код;
реализация принципа единственной ответственности;
модели выполнения (производители-потребители, читатели-писатели, обедающие философы);
корректное завершение;
понятие рефакторинга, принципы;
тестирование.

---

8 **Стратегии разрешения зависимостей IoC**

Цели занятия:

научиться писать различные стратегии разрешения зависимостей

Краткое содержание:

смысл «Одиночки», когда он применяется с точки зрения архитектуры, когда есть один канал (например, если если есть сокетное соединение, и по нему передаётся информация) и его проще контролировать, глобальные настройки и т.д.;
плюсы, минусы;
что делать с синглтоном в многопоточных приложениях?
как создать синглтон?
как проверить, что он действительно один?

---

9 **Адаптер и мост // ДЗ**

Цели занятия:

узнать и научиться применять шаблоны "адаптер" и "мост"

Краткое содержание:

две команды: одна проектирует приложение со своим интерфейсом (например, передача данных по сокетному соединению), другая проектирует своё приложение, которое должно отправлять сообщение в другую систему.

Домашние задания

1 Адаптер для работы двух независимых программ. Описание применения шаблона в проекте

Цель: 1. Вы напишете адаптер, чтобы связать функционал двух отдельных программ в единый процесс. Разберётесь с тем, как адаптер работает в случае вызова отдельных программ. Получите навыки работы с формальными и фактическими параметрами передачи данных.

2. Получите навык анализа системы - использовать или нет этот шаблон в проектной работе.

Написать простую консольную программу П1, с интерфейсом вызова И1, которая читает данные о двух матрицах A и B из файла F0, складывает матрицы и сохраняет результат A+B в другой файл F1.
Написать вторую консольную программу П2, которая может генерить данные матриц A и B и писать их в файл с именем F2.
Чтобы она могла их просуммировать, следует сделать адаптер для программы П1, который позволит программе П2 вызвать П1.
1. Написать программу П1
2. Написать программу П2, включив туда адаптер вызова и использования программы П1
3. Написать автотест для проверки функционирования
4. Если потребуется использовать адаптер в проектной работе, предоставить описание в текстовом файле в GitHub репозитории где конкретно и в какой роли используется этот шаблон.
5. Нарисовать диаграмму классов.
Срок - 2 недели от занятия.
ДЗ сдается в виде ссылки на GitHub репозиторий с проектом.
По вопросам обращаться в Slack к преподавателям и наставникам в канал группы.

---

10 **QA-сессия**

4 Масштабирование

1 **Проблема вертикального масштабирования и синхронизация. //ДЗ**

Цели занятия:

показать основные архитектурные решения для приложений, использующих несколько потоков.

Краткое содержание:

"Бесплатного супа больше не будет"
Ограничение "традиционной" модели вычислений.
Акторная модель
Без инверсии зависимостей.

Домашние задания

1 Многопоточное выполнение команд

Цель: Разработка многопоточной системы выполнения команд на примере игры "Танки".

В результате выполнения ДЗ будет получен код, отвечающий за выполнение множества команд в несколько потоков, устойчивый к появлению новых видов команд и дополнительных ограничений, накладываемых на них.

Предположим, что у нас есть набор команд, которые необходимо выполнить. Выполнение команд организуем в несколько потоков.
Для этого будем считать, что у каждого потока есть своя потокобезопасная очередь.
Для того, чтобы выполнить команду, ее необходимо добавить в очередь. Поток читает очередную команду из очереди и выполняет ее.
Если выполнение команды прерывается выброшенным исключением, то поток должен отловить его и продолжить работу.
Если сообщений нет в очереди, то поток засыпает до тех пор, пока очередь пуста.

Последовательность шагов решения:

1. Реализовать код, который запускается в отдельном потоке и делает следующее
В цикле получает из потокобезопасной очереди

		команду и запускает ее. Выброс исключения из команды не должен прерывать выполнение потока. 2. Написать команду, которая стартует код, написанный в пункте 1 в отдельном потоке. 3. Написать команду, которая останавливает цикл выполнения команд из пункта 1, не дожидаясь их полного завершения (hard stop). 4. Написать команду, которая останавливает цикл выполнения команд из пункта 1, только после того, как все команды завершат свою работу (soft stop). 5. Написать тесты на команду запуска и остановки потока.
2	**Интеграция программного обеспечения**	**Цели занятия:** объяснить, как работать с интеграцией программного обеспечения и какие существуют подходы. **Краткое содержание:** интеграция данных, интеграция приложений; факторы, влияющие на принятие решений по интеграции.
3	**Системы обмена сообщениями // ДЗ**	**Цели занятия:** рассмотреть архитектурные концепции построения систем обмена сообщений. **Краткое содержание:** стили интеграции (File Transfer, RPI, Shared Database, Messaging); основные концепции обмена сообщениями; каналы, сообщение, маршрутизация, трансляция, конечная точка. **Домашние задания** 1 Написать Endpoint для приема входящих сообщений

Цель: Написать Endpoint для приема входящих сообщений Игровым сервером.

Термины:

Игровой сервер - это приложение, на котором вычисляется состояние танковых боев, то есть выполняются все те команды, которые рассматривались на предыдущих занятиях.

Агент - это специальное приложение, на котором игрок запускает свой алгоритм управления своей командой танков.

Для полноценной реализации игры необходимо обеспечить двусторонний обмен данными между Игровым сервером и Агентами. Частью такого обмена будет Endpoint для приема входящих сообщений Игорвым сервером.

В результате выполнения ДЗ будет получен код, основанный на паттернах системы обмена сообщениями для приема входящих сообщений от Агентов.

Цель: применить навыки построения архитектуры, основанной на системе обмена сообщениями.

Задача Endpoint принять входящее сообщение от Агента, десериализовать его и передать его на обработку внутрь игрового сервера.
Прием входящих сообщений и их десериализацию скорее всего возьмут на себя те библиотеки, которыми Вы будете пользоваться. А вот определить какому из танковых боев это сообщение было адресовано (маршрутизация) и его уведомление о входящем сообщении (передача сообщения в обработку) - это то, что придется делать в рамках этого ДЗ.

Организовать Endpoint можно реализовать разными способами. Здесь выбирайте тот, который считаете полезным для своих прикладных задач на работе, а не для данной игры.
Ну или тот, с которым сами бы хотели разобраться. Ниже приведены разные варианты, что можно было бы сделать:
- Endpoint поверх Tcp/IP или UDP протокола.

- Controller в web-фреймворке.
- Endpoint на основе веб-сокетов.
- Rest
- Подписка на получение сообщений из какого-нибудь Message Broker'a.

Выбор варианта не сильно отразится на остальной архитектуре, то есть концептуально будет одно и тоже, а вот у каждого варианта могут быть свои ньюансы реализации, поэтому степень полезности того или иного способа реализации прямо зависит от того, в какую сторону лично Вы хотите развиваться.

Шаги выполнения ДЗ:
1. Определить формат сообщений, которые отправляет Агент игровому серверу.
Указание: правила во всех играх могут сильно отличаться друг от друга, поэтому нужно подумать о таком формате, который бы не зависел от текущей реализации конкретных команд, то есть чтобы endpoint не приходилось модифицировать каждый раз, когда мы разработаем новое правило игры. Как идею для решения поставленной задачи можно рассмотреть следующий набор данных, который стоит передавать в сообщении:
- id игры - для идентификации игры, в рамках которой это сообщение обработано. С помощью этого id можно будет определить получателя сообщения при маршрутизации сообщения внутри игрового сервера.
- id игрового объекта, которому адресовано сообщение. С помощью этого id можно будет определить игровой объект внутри игры, для которого адресовано это сообщение.
- id операции - по этому id в IoC можно будет определить команду, которая будет обрабатывать данное сообщение.
- args - вложенный json объект с параметрами операции. Содержимое этого объекта полностью зависит от команды, которая будет применяться к игровому объекту.

2. Определить endpoint, который принимает входящее сообщение и конвертирует в команду InterpretCommand.
enpoint должен определить игру, которой адресовано сообщение, создать команду InterpretCommand и поместить эту команду в

очередь команд этой игры.
Команда InterpretCommand получает всю информацию об операции, которую необходимо выполнить, параметрах и объекте, над которым эта операция будет выполняться.

3. Задача команды InetrpretCommand на основе IoC контейнера создать команду для нужного объекта, которая которая соответствует приказу, содержащемуся в сообщении и постановки этой команды в очередь команд игры.
Например, если сообщение указано, что объект с id 548 должен начать двигаться, то результат InterpretCommand заключается можно описать следующим псевдокодом

```
var obj = IoC.Resolve<UObject>("Игровые объекты", "548"); // "548" получено из входящего сообщения
IoC.Resolve("Установить начальное значение скорости", obj, 2); // значение 2 получено из args переданного в сообщении
var cmd = IoC.Resolve<Command>("Движение по прямой", obj); // Решение, что нужно выполнить движение по прямой получено из сообщения
// обратите внимание само значение "Движение по прямой" нельзя читать на прямую из сообщения,
// чтобы избежать инъекции, когда пользователь попытается выполнить действие, которое ему выполнять не позволено
IoC.Resolve<Command>("Очередь команд", cmd).Execute(); // Выполняем команду, которая поместит команду cmd в очередь команд игры.
```

Дополнительно:
Можно реализовать двусторонний обмен данными между Игровым сервером и Агентом. Так как для принятия решения о той или иной операции на Агенте необходимо знать текущее состояние танковой битвы. Для этого можно использовать websocket или tcp/ip или любой другой способ. С точки зрения игры это выглядит так - есть команда, при выполнении которой она сериализует состояние игровых объектов и инициирует процесс отправки этих данных на Агент. Далее рассылка сохраненного состояния производится через реализованный двусторонний обмен данными.

4 **Построение архитектуры приложения, построенного на системе обмена сообщениями**

Цели занятия:

будет знать как строить сервер приложений, построенный на системе обмена сообщениями.

Краткое содержание:

основные проблемы;
способы синхронизации.

5 Time To Market

1 **Методологии разработки ПО**

Цели занятия:

объяснить различные подходы к разработке ПО, их особенности и ограничения.

Краткое содержание:

модели жизненного цикла и методологии разработки ПО.

---

2 **Создание микросервиса // ДЗ**

Цели занятия:

познакомиться с паттернами декомпозиции системы на микросервисы.

Краткое содержание:

технический подход и бизнес-подход к декомпозиции.

Домашние задания

1 Создать микросервис Авторизация пользователя

Цель: В предыдущем ДЗ необходимо было реализовать Endpoint на Игровом сервере для приема входящих сообщений от Агента.

Термины:

Игровой сервер - это приложение, на котором вычисляется состояние танковых боев, то есть выполняются все те команды, которые рассматривались на предыдущих занятиях.

Агент - это специальное приложение, на котором игрок запускает свой алгоритм управления своей командой танков.

При реализации этого Endpoint возникает задача авторизации пользователя на отправку сообщений для управления командой танков в конкретной игре, чтобы не допустить вмешательства в процесс игры сторонними пользователями.

В результате выполнения этого ДЗ будет разработан микросервис, который выдает jwt токен из участников танкового сражения, для того, чтобы игровой сервер мог принять решение о возможности выполнения входящего сообщения от имени пользователя.

Цель: применить навыки разработки микросервиса.

Предполагается реализация микросервиса авторизации с помощью jwt токена.

Алгоритм взаимодействия сервиса авторизации и Игрового сервера следующий:

1. Один из пользователей организует танковый бой и определяет список участников (их может быть больше 2-х).
На сервер авторизации уходит запрос о том, что организуется танковый бой и присылается список его участников. Сервер в ответ возвращает id танкового боя.
2. Аутентифицированный пользователь посылает запрос на выдачу jwt токена, который авторизует право этого пользователя на участие в танковом бое.
Для этого он должен указать в запросе id танкового боя.
Если пользователь был указан в списке участников танкового боя, то он выдает пользователю jwt токен, в котором указан id игры.
3. Пользователь при отправке сообщений в Игровой сервер прикрепляет к сообщениям выданный jwt токен, а сервер при получении сообщения удостоверяется, что токен был выдан сервером авторизации (проверят хэш jwt токена) и проверяет, что пользователь запросил выполнение операции в игре, в которой он эту операцию может выполнять.

3 **DevOps**

Цели занятия:

рассмотреть процесс создания и поставки ПО в рамках devops.

Краткое содержание:

проблемы доставки ПО;
4 фактора devops;
версионирование, тестирование, CI, CD, мониторинг;
инструментарий.

---

4 **Микросервисная архитектура // ДЗ**

Цели занятия:

рассмотреть описание, характерные свойства и характеристики;
оценить порядок перехода от "монолита".

Краткое содержание:

понятие микросервисной архитектуры;
свойства и характеристики;

Домашние задания

1 Разработать микросервисную архитектуру для игры Танки

Цель: Представить в виде одной или нескольких диаграмм архитектуру приложения для игры в танки.

Цель: разбить Игру Танки на набор взаимодействующих между собой микросервисов и приложений.

Танковые бои могут проводить только зарегистрированные пользователи. Танковые бои проводятся как в рамках турниров, либо между любыми пользователями по договоренности. Можно посмотреть список будущих турниров, подать заявку на участие в турнире, посмотреть результаты проходящих турниров, уже прошедших.

Участники боев получают уведомлении о приглашении на турнир, решение по заявке на участие в турнире, о завершении танкового боя, о скором начале стартового боя.
Участник танкового боя может посмотреть прошедший бой. За места в турнире участники получают рейтинговые очки.
Каждый пользователь может организовать свой турнир. Турниры получают рейтинг, который рассчитывается на основе рейтингов ее участников. Турнир может быть регулярным, тогда рейтинг его накапливается.
Игрок участвует в танковом бое посредством программы, которая загружается в специальное приложение Агент.

Для успешного решения задачи необходимо:
1. Определить набор микросервисов и веб-приложений и отобразить их на диграмме/наборе диаграмм.
2. Определить направления обмена сообщениями и Endpoints для каждого микросервиса и приложения.
3. Определить узкие места и потенциальные проблемы масштабирования приложения и способы их решения.
4. Определить компоненты, к которым чаще всего будут меняться требования и способы выполнения OCP для них.
5. Сделать текстовые пояснения к созданному решению.

Все документы, составляющие решение выложить в git.

6 Шаблоны проектирования

1 Итератор

Цели занятия:

объяснить, как инкапсулировать этот процесс;
проанализировать один общий интерфейс для перебора, множество реализаций;
объяснить как работает итератор;
рассмотреть на примере.

Краткое содержание:

коллекция как сложно упорядоченный набор данных. Это и списки, и деревья;
связи могут быть сложными.

---

2 Состояние // ДЗ

Цели занятия:

объяснить:
состояние не так однозначно, как кажется;
диаграмма состояний и переходов;
обзор конечных автоматов;
простая прямая реализация;
расширение функциональности;
от простой реализации объектов к интерфейсам.

Краткое содержание:

локализация поведения каждого состояния в отдельных классах;
инкапсуляция состояний;
определение шаблона "Состояние" как результат анализа примера.

Домашние задания

1 Смена режимов обработки команд

Цель: Применит паттерн Состояние для изменения поведения обработчиков Команд.

В домашнем задании №2 была реализована многопоточная обработка очереди команд.

Предлагалось два режима остановки этой очереди - hard и soft.
Однако вариантов завершения и режимов обработки может быть гораздо больше. В данном домашнем задании необходимо реализовать возможность смены режима обработки Команд в потоке, начиная со следующей Команды.

Для этого предлагается использовать паттерн Состояние. Каждое состояние будет иметь свой режим обработки команд. Метод handle возвращает ссылку на следующее состояние.
Необходимо реализовать следующие состояния автомата:
1. "Обычное"
В этом состоянии очередная команда извлекается из очереди и выполняется. По умолчанию возвращается ссылка на этот же экземпляр состояния.
Обработка команды HardStop приводит к тому, что будет возвращена "нулевая ссылка" на следующее состояние, что соответствует завершению работы потока.
Обработка команды MoveToCommand приводит к тому, что будет возвращена ссылка на состояние MoveTo
2. MoveTo - состояние, в котором команды извлекаются из очереди и перенаправляются в другую очередь. Такое состояние может быть полезно, если хотите разгрузить сервер перед предстоящим его выключением.
Обработка команды HardStop приводит к тому, что будет возвращена "нулевая ссылка" на следующее состояние, что соответствует завершению работы потока.
Обработка команды RunCommand приводит к тому, что будет возвращена ссылка на "обычное" состояние.

3 **Цепочка обязанностей // ДЗ**

Цели занятия:

изучить шаблон и научиться его применять.

Краткое содержание:

как увязать множество типов сообщений и множество обработчиков этих сообщений?
превращения элементов поведения в объекты;
рассмотрим на примере - формируем диаграмму классов.

Домашние задания

1 Реализация проверки коллизий игровых объектов

Цель: Научится применять цепочку обязанности

---

4 **Заместитель**

Цели занятия:

научиться делегировать;
научиться давать команду, чтобы "Заместитель" (Proxy) выполнил задачу.

Краткое содержание:

работа двух объектов: клиента и сервера;
обсуждение причин необходимости перехвата доступа и взятия его под контроль;
различные виды заместителя (удалённый, виртуальный, замещающий, кеширующий, синхронизирующий...);
рассмотрение на конкретном примере преимущества и недостатки.

---

5 **Декоратор**

Цели занятия:

применять шаблон.

Краткое содержание:

основной принцип - добавление функциональности к существующему объекту, возможно, с учётом
очередности действий;
реализации;
некоторая система - интерфейс;
"декоратор", "адаптер" и "прокси", сходства и различия;

задание: одна команда реализует прокси, а другая адаптер к некой системе, обсуждение различий.

---

6 **Шаблонный метод**

Цели занятия:

объяснить, что цели можно достичь разными путями, порой важно оперировать не объектами, а процессами;

проанализировать:
от бизнес процесса к абстрактным общим шагам;
декомпозиция;
временная диаграмма;
один процесс, разные шаги;
необходимость контроля процесса;
методы-перехватчики.

Краткое содержание:

примеры использования;
применение на практическом примере.

---

7 **Интерпретатор // ДЗ**

Цели занятия:

изучить шаблон и научиться его применять.

Краткое содержание:

назначение, структура;
формы Бекуса-Науэра;
диаграммы Вирта;
лексический, синтаксический и семантический анализы.

Домашние задания

1 Разработать систему команд для танков

Цель: Научиться применять интерпретатор

7 Проектная работа

1 Вводное занятие по проектной работе

Цели занятия:

определиться с финальной архитектурой проекта;
выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом.

Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Обсуждаем вопросы проектирования, корректируем принятые решения

Цель: выбрать темы;
закрепить тему в чат с преподавателем.

1. выбрать стиль архитектуры;
2. описать бизнес-процессы;
3. отрисовать функциональные процессы;
4. копонентная схема;
5. информационная схема;
6. используемые шаблоны;
7. описание интеграций.

проект должен содержать: виртуальную машину, где развёрнут проект (при необходимости);
документация: описание бизнес-процессов, отрисованные, функциональные процессы, копонентная схема, информационная схема, описание интеграций;
доклад (15 минут).

2	**Консультация по проектам и домашним заданиям**	**Цели занятия:** получить ответы на вопросы по проекту, ДЗ и по курсу. **Краткое содержание:** вопросы по улучшению и оптимизации работы над проектом; затруднения при выполнении ДЗ; вопросы по программе.
3	**Защита проектных работ**	**Цели занятия:** защитить проект и получить рекомендации экспертов. **Краткое содержание:** презентация проектов перед комиссией; вопросы и комментарии по проектам.