1) ООП – парадигма программирования, в которой программная система рассматривается как совокупность объектов и их взаимодействий. ООП строится на таких принципах как:

Инкапсуляция – объединение полей с данными и методов в одной сущности с варьирующимися правами доступа к ним.

Наследование – возможность производного класса унаследовать некоторые поля с данными или методы класса-предка.

Полиморфизм – возможность классов имеющих одного и того же предка иметь различное поведение (различные функции).

Абстракция – возможность не принимать во внимание все детали реализации функционала объекта при работе с ним.

### 2) Объектно-ориентированная декомпозиция – процесс разбиения программной системы на модули, когда в качестве модулей выступают объекты (классы). Например, можно использовать паттерны проектирования.

### 3) Дизайн ПО – процесс определения (проектирования) структуры (архитектуры) компонентов ПО для удовлетворения заданных требований и достижения определенных целей. Примеры:

### Прототипирование

### Создание моделей данных и диаграмм

### 4) Магическое число 7 Миллера в контексте IT :

### Число 7 (+-2) – число объектов, которые человек может одновременно держать в памяти (во внимании) согласно Джорджу Миллеру. Примеры:

### При дизайне UI/UX принято не создавать в меню больше 7 пунктов чтобы человеку было легче ориентироваться.

### Обычно при вложении циклов или условий не стоит допускать вложенности больше 7.

### Число параметров в функции не следует делать слишком большим при возможности (например, при работе с winapi я пока что не видел функции с числом параметров больше 7)

### 

При создании коммита по возможности не следует менять много файлов в репозитории (больше 7) или при создании пулл реквеста.

Не следует делить сервис на более чем 7 микросервисов (тут имеется в виду для работы одной команды с этими 7 микросервисами)

При работе над большой задачей лучше делить ее не больше чем на 7 подзадач (а их в свою очередь опять не больше чем на 7 и так с небольшой вложенностью).

5) 5 признаков сложных систем по Гради Бучу:

Иерархическая структура

Относительная элементарность подсистем

Разделение на компоненты (которые обычно взаимодействуют меньше между собой чем внутри себя)

Общие механизмы (компоненты)

Превращение сложных компонентов в элементарные (постоянное совершенствование)

Пример 1: Операционная система

Иерархия: самое верхнее – ядро (управляет ресурсами, устройствами, правами доступа), затем идет графическая оболочка и другие интерфейсы для работы с ядром.

Относительная элементарность подсистем: для разработчика ядра элементарными являются подсистема выделения памяти, драйвера, система выделения процессорного времени потокам, а для пользователя элементарными являются, например, подсистемы создания файлов и подключения к проводным и беспроводным сетям (на высоком уровне), создание ярлыка или виджета.

Разделение на компоненты: например, система работы с сетевыми протоколами обычно слабо взаимодействует с графической оболочкой, но внутри себя содержит очень много сильно связанных между собой компонентов.

Общие механизмы: можно сказать, что, все подсистемы операционной системы используют выделение блоков в оперативной памяти или также многие подсистемы используют запись и чтение с диска.

Превращение сложных компонентов в элементарные: со временем разработчики операционных систем всё меньше возвращаются к реализации корневых систем (например, опять же выделение памяти или bootloader) и они со временем становятся элементарными и на их базе строятся сложные системы.

Пример 2: Сервис, предоставляющий облачные вычислительные мощности (например, Google Cloud)

Иерархия: самое верхнее – операционная система, которая и выделяет процессорное время на сервере для вычислений или хостинга какого-то приложения (развертываемой серверной системы того, кто использует платформу), далее логика (система) отдельных микросервисов, из которых состоит сам сервис (то есть, например сервис аутентификации, сервис БД или сервис REST API), чуть ниже идут различные системы мониторинга, фронтенд составляющие, различные инструменты для управления. (как я понял)

Относительная элементарность подсистем: для команд разработчиков, которые занимаются одним микросервисом, его системы будут выглядеть как сложные системы, построенные на чем-то более элементарном, а для разработчиков, которые занимаются другим микросервисом, системы первого будут восприняты как элементарные, потому что им не нужно глубоко погружаться в то, как они работают.

Разделение на компоненты: Микросервисы и есть такие компоненты.

Общие механизмы: обычно предусмотрено, что все микросервисы можно воспроизводить в нескольких экземплярах или расширять (обычно это происходит автоматически). То есть при реализации микросервисов используется эта общая концепция. Также можно как пример к этому пункту привести взаимодействие микросервисов между собой (все они имеют такие механизмы).

Превращение сложных компонентов в элементарные: можно сказать, что инструменты, которые использовались для реализации репликации и расширения микросервисов со временем автоматизировались и стали более простыми в использовании (элементарными).

Пример 3: многопользовательская игра (например CS 2)

Иерархия: первое и самое верхнее – физический движок и графический движок, затем логика игры, реализованная на их основе. Также для многопользовательской игры должен существовать сервер, так что к одной из самых верхних систем в иерархии относится серверная составляющая (то есть, например менеджмент множественных подключений). Также чуть ниже идут различные сервисы (БД, развертывание, мониторинг и управление)

Относительная элементарность подсистем: то же самое, что и в предыдущих примерах – для разных разработчиков разные системы будут являться элементарными.

Разделение на компоненты: опять же микросервисы, матчмейкер, система игрового чата.

Общие механизмы: взаимодействие с сетью (быстрая передача данных с сервера на клиент и их дальнейшее представление в игре), взаимодействие различных составляющих с физическим и графическим движками.

Превращение сложных компонентов в элементарные: Графика всё время совершенствуется и становится всё проще в реализации благодаря новым игровым движкам (наглядный пример – CS 2 с ее новой графикой, отличной от CS GO)