Исаев Кирилл, 2 курс, 7 группа

Вариант 1

0. ООП (объектно-ориентированное программирование) - это подход, при котором программа рассматривается как набор некоторых элементов, которые могут взаимодействовать между собой.

В контексте ООП выделяют 3 основных “кита” ООП: наследование, полиморфизм, инкапсуляция.

Инкапсуляция - процесс сохранения функционала внутри одной “коробки” – объекта. Кроме того, некоторые поля и методы этого объекта делаются недоступными вне класса (модификаторы доступа). Например, при создании класса “user” у мессенджера пароль должен храниться в поле private.

Наследование - это создание объекта с помощью части полей и методов другого класса. Пример: класс автомобиль (например поля – мощность двигателя, оцинковка кузова, тип коробки передач). От него наследуем марки крутых авто (например, бмв, детали которого в ремонте дороже самого авто, ауди, лада). Эти классы наследуют класс авто – это оптимизирует код, позволяя не писать несколько раз одно и тоже).

Полиморфизм - это свойство ООП, позволяющее обращаться с объектами разных типов через единый интерфейс, при этом поведение этого интерфейса определяется конкретным классом.

1. Декомпозиция - это метод проектирования, при котором сложная система разбивается на меньшие, более простые части (объекты, классы, модули), которые взаимодействуют между собой.

Свойства декомпозиции: каждый класс должен решать одну конкретную задачу, внутренняя реализация компонента скрыта, мы можем видеть только необходимый интерфейс, связь между компонентами минимальна.

Правильная декомпозиция проекта напрямую влияет на его тестируемость и масштабируемость.

Суть ООДекомпозиции - перейти от вопроса “что делает программа” к “кто что делает в программе”.

2. Дизайн ПО - это процесс создания архитектуры и структуры программы.

Выделяется 5 этапов дизайна ПО:

- определяем, что должна делать система (постановка задачи)

- выявление сущностей и их ответственностей (разбиение на классы программы)

- проектирование взаимодействий

- определение структур данных для хранения информации

- проектирование ключевых алгоритмов (основа задачи)

Основные принципы хорошего дизайна:

- KISS - упрощение кода, каждый метод на вход принимает до 5 (отличается в зависимости от шаблонов проектирования, может быть например и 7) аргументов (по возможности уменьшить количество), каждый класс не более 500 строк кода, каждый метод не более 100 строк кода. Также стоит учитывать “магическое число Миллера (7+-2)” (например, то же кол-во аргументов).

- DRY - если блок кода встречается 3 или более раз в коде, значит его нужно вынести в отдельный метод.

3. Магическое число Миллера - предел человека одновременно удерживать в рабочей памяти 7+-2 элементов, учитываемый при разработке.

Примеры:  
1) в классе 20 методов/полей -> разбиваем на 3 класса по 7+-2 методов/полей в каждом (бэкенд)

2) при создании UI (например, касса самообслуживания в магазине) разделяем иконки по цветам, размерам, местоположению (кнопка оплаты справа снизу и больше других, помощь кассира красная, отдельно от других), делая интерфейс интуитивно понятнее

3) количество параметров функции (аналогично пункту 1 разбиваем на несколько методов) (бэкенд)

4) доска проекта в YouGile, содержащая 5 колонок, в каждой 7+-2 задачи с таким же кол-вом подзадач (при необходимости разбивать проект на 2 доски (или даже 2 проекта))

5) тесты (каждый тест - 7+-2 тестов метода)

6) документация

7) количество задач, которые параллельно может взять выполнять 1 разработчик

4. 5 признаков сложных систем по Гради Бучу:

- иерархичность

- относительность элементарного

- разделение на слабосвязанные модули

- устойчивые промежуточные формы

- эмерджентное поведение

Пример: Kubernetes

- иерархичность

Cluster → Namespace → Deployment → Pod → Container

- относительность элементарного

Для разработчика: Container - элементарная единица

Для бизнеса: весь Cluster - элементарная единица

- слабосвязанные модули

kubelet, API Server, scheduler общаются через API, а не напрямую

- устойчивые промежуточные формы

Rolling update: v1.0 → v1.1 (50%) → v1.1 (100%) - каждый этап рабочий

- эмерджентное поведение

Автоматическое самовосстановление и масштабирование - свойство всей системы, а не отдельных компонентов