**Билет 21**

Согласно Кенту, архитектура может считаться «простой», если она:

**1.обеспечивает прохождение всех тестов**

Система, тщательно протестированная и прошедшая все тесты, *контролируема*. На первый взгляд утверждение кажется очевидным, но это весьма важно. Невозможно проверить работу системы, которая не является контролируемой, а непроверенные системы не должны запускаться в эксплуатацию.

**2.не содержит дублирующегося кода**

Дублирование — главный враг хорошо спроектированной системы. Его последствия — лишняя работа, лишний риск и лишняя избыточная сложность. Дублирование проявляется во многих формах. Конечно, точное совпадение строк кода свидетельствует о дублировании. Похожие строки часто удается «причесать» так, чтобы сходство стало еще более очевидным; это упростит рефакторинг (последовательную переработку кода). Кроме того, дублирование может существовать и в других формах — таких, как дублирование реализации.

**3.выразительность**

Легко написать код, понятный для нас самих, потому что в момент его написания мы глубоко понимаем решаемую проблему. У других программистов, которые будут заниматься сопровождением этого кода, такого понимания не будет.

-*Хороший выбор имен* помогает выразить ваши намерения. Имя класса или функции должно восприниматься «на слух», а когда читатель разбирается в том, что делает класс, это не должно вызывать у него удивления.

-*Относительно небольшой размер функций и классов* также помогает выразить ваши намерения. Компактным классам и функциям проще присваивать имена; они легко пишутся и в них легко разобраться.

-*Хорошо написанные модульные тесты* тоже выразительны. Они могут рассматриваться как разновидность документации, построенная на конкретных примерах. Читая код тестов, разработчик должен составить хотя бы общее представление о том, что делает класс

**4.использует минимальное количество классов и методов**

Стремясь уменьшить объем кода наших классов и методов, мы можем наплодить слишком много крошечных классов и методов. Это правило рекомендует ограничиться небольшим количеством функций и классов. Наша цель — сделать так, чтобы система была компактной, но при этом одновременно сохранить компактность функций и классов. Однако следует помнить, что из четырех правил простой архитектуры это правило обладает наименьшим приоритетом.