Контрольные вопросы:

1. У предварительного проектирования повышенный уровень абстракции.

2. Database-centric Architecture

* Client-Server Architecture
* Three-Tier Architecture
* Front-End and Back-End
* Web Application
* Data Flow Architecture

3. Модели управления:

*Централизованные:*

* Вызов-возврат
* Модель диспетчера

*Событийное управление:*

* Широковещательная модель
* Управление прерываниями

4. *Связность модуля*  – мера независимости его частей. Чем выше связность, тем больше связей он «упрятывает» в себя.

Типы связности:

***Функциональная связность***

Функционально связный модуль содержит объекты, предназначенные для решения одной единственной задачи.

***Последовательная связность***

В последовательно связном модуле его объекты охватывают подзадачи, для которых выходные данные одной из подзадач являются входными для другой

***Информационная связность***

Содержит объекты, использующие одни и те же входные или выходные данные. Так, по ISBN книги, можно узнать ее название, автора и год издания. Эти три процедуры (определить название, определить автора, определить год издания) связаны между собой тем, что все они работают с одним и тем же информационным объектом – ISBN.

***Процедурная связность***

Процедурно связный модуль – это такой модуль, объекты которого включены в различные (возможно, несвязанные) подзадачи, в которых управление переходит от одной подзадачи к следующей

***Временная связность***

Модуль с временной связностью – это такой модуль, в котором объекты модуля привязаны к конкретному промежутку времени.

***Логическая связность***

Модуль с логической связностью – это такой модуль, объекты которого содействуют решению одной общей подзадачи, для которой эти объекты отобраны во внешнем по отношению к модулю мире.

***Связность по совпадению***

Модуль со связностью по совпадению содержит объекты, которые слабо связаны друг с другом (сходить в кино, поужинать, посмотреть телевизор, проверить электронную почту).

5.*Сцепление (coupling)* - мера взаимозависимости модулей по данным.

* ***Независимое сцепление***возможно, если модули не вызывают друг друга и не обрабатывают одну и ту же информацию.
* Модули ***сцеплены по данным*,**если они имеют общие простые элементы данных, которые передаются от одного модуля к другому как параметры.

(пр, структура – массив, а передается в качестве параметра элемент массива; при этом изменения в структуре данных не повлияют на другой модуль). Модули со сцеплением по данным не имеют общей области данных (глобальных переменных).

* Модули ***сцеплены по образцу***, если в качестве параметров используются структуры данных (например, в качестве параметра передается массив). Недостаток: оба модуля должны содержать информацию о др модуле(внутренней структуре данных).т.е изменения должны отображаться и в др. модуле.
* Модули ***сцеплены по общей области,*** если они разделяют одну и ту же глобальную структуру данных.
* Модули ***сцеплены по управлению,*** если какой-либо из них управляет решениями внутри другого с помощью передачи флагов, переключателей и т.д.
* Модуль ***сцеплен по внешним ссылкам,*** если у него есть доступ к данным другого модуля через внешнюю точку входа.
* Модули **сцеплены по кодам,** если коды их команд объединены друг с другом, использовать общий участок памяти.

6. ***Делегирование*** — основной шаблон проектирования, в котором объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту. Шаблон делегирования является фундаментальной абстракцией, на основе которой реализованы другие шаблоны - композиция (также называемая агрегацией), примеси (mixins) и аспекты (aspects)

***Неизменяемый объект*** - это объект , состояние которого не может быть изменено после его создания.

***Интерфейс*** — основной шаблон проектирования, являющийся общим способом структурирования программ для того, чтобы их было проще понять.

***MVC (Model-View-Controller)*** — это шаблон программирования, который позволяет разделить логику приложения на три части:

* **Model (модель)**. Получает данные от контроллера, выполняет необходимые операции и передаёт их в вид.
* **View (вид или представление)**. Получает данные от модели и выводит их для пользователя.
* **Controller (контроллер)**. Обрабатывает действия пользователя, проверяет полученные данные и передаёт их модели.