ФОС по дисциплине  
«Объектно-ориентированное проектирование»

1. Повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста, называется

* штамп
* трафарет
* стандарт
* **шаблон**

1. Шаблоны проектирования, которые абстрагируют процесс инстанцирования, называются

* вызывающими
* **порождающими**
* зарождающими
* поднимающими

1. Шаблоны проектирования, которые определяют различные сложные структуры, изменяющие интерфейс уже существующих объектов или его реализацию, позволяя облегчить разработку и оптимизировать программу, называются

* конструктивными
* кантировочными
* **структурными**
* каркасными

1. Шаблоны проектирование, определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов, называются

* алгоритмическими
* реализующими
* бихевиористическими
* **поведенческими**

1. Порождающими называются шаблоны проектирования, которые

* создают интерфейс неизменяемого объекта
* **абстрагируют процесс инстанцирования**
* определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию
* определяют алгоритмы и способы реализации взаимодействия различных объектов и классов

1. Структурными называются шаблоны проектирования, которые

* создают интерфейс неизменяемого объекта
* абстрагируют процесс инстанцирования
* **определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию**
* определяют алгоритмы и способы реализации взаимодействия различных объектов и классов

1. Поведенческими называются шаблоны проектирования, которые

* создают интерфейс неизменяемого объекта
* абстрагируют процесс инстанцирования
* определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию
* **определяют алгоритмы и способы реализации взаимодействия различных объектов и классов**

1. Шаблон Абстрактная фабрика

* **представляет собой интерфейс для создания компонентов системы**
* представляет собой интерфейс для создания сложного объекта
* определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать
* определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор

1. Шаблон Строитель

* представляет собой интерфейс для создания компонентов системы
* **представляет собой интерфейс для создания сложного объекта**
* определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать
* определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор

1. Шаблон Фабричный метод

* представляет собой интерфейс для создания компонентов системы
* представляет собой интерфейс для создания сложного объекта
* **определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать**
* определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор

1. Шаблон Прототип

* представляет собой интерфейс для создания компонентов системы
* представляет собой интерфейс для создания сложного объекта
* определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать
* **определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор**

1. Представляет собой интерфейс для создания компонентов системы

* Шаблон Прототип
* Шаблон Фабричный метод
* Шаблон Строитель
* **Шаблон Абстрактная фабрика**

1. Представляет собой интерфейс для создания сложного объекта

* Шаблон Прототип
* Шаблон Фабричный метод
* **Шаблон Строитель**
* Шаблон Абстрактная фабрика

1. Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать

* Шаблон Прототип
* **Шаблон Фабричный метод**
* Шаблон Строитель
* Шаблон Абстрактная фабрика

1. Определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор

* **Шаблон Прототип**
* Шаблон Фабричный метод
* Шаблон Строитель
* Шаблон Абстрактная фабрика

1. Шаблон Адаптер - это

* Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо
* **Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс**
* Объект, который объединяет в себе объекты, подобные ему самому
* Класс, расширяющий функциональность другого класса без использования наследования

1. Шаблон Мост - это

* **Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо**
* Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс
* Объект, который объединяет в себе объекты, подобные ему самому
* Класс, расширяющий функциональность другого класса без использования наследования

1. Шаблон Компоновщик- это

* Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо
* Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс
* **Объект, который объединяет в себе объекты, подобные ему самому**
* Класс, расширяющий функциональность другого класса без использования наследования

1. Шаблон Декоратор- это

* Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо
* Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс
* Объект, который объединяет в себе объекты, подобные ему самому
* **Класс, расширяющий функциональность другого класса без использования наследования**

1. Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо, называется

* Адаптер
* **Мост**
* Компоновщик
* Декоратор

1. Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс, называется

* **Адаптер**
* Мост
* Компоновщик
* Декоратор

1. Объект, который объединяет в себе объекты, подобные ему самому, называется

* Адаптер
* Мост
* **Компоновщик**
* Декоратор

1. Класс, расширяющий функциональность другого класса без использования наследования, называется

* Адаптер
* Мост
* Компоновщик
* **Декоратор**

1. Фасад – это

* **Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое**
* Класс, который может иметь только один экземпляр
* Объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым
* Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него

1. Одиночка – это

* Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое
* **Класс, который может иметь только один экземпляр**
* Объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым
* Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него

1. Приспособленец – это

* Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое
* Класс, который может иметь только один экземпляр
* **Объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым**
* Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него

1. Заместитель – это

* Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое
* Класс, который может иметь только один экземпляр
* Объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым
* **Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него**

1. Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое, называется

* Одиночка
* **Фасад**
* Приспособленец
* Заместитель

1. Класс, который может иметь только один экземпляр, называется

* **Одиночка**
* Фасад
* Приспособленец
* Заместитель

1. Объект, представляющий себя как уникальный экземпляр в разных местах программы, но фактически не являющийся таковым, называется

* Одиночка
* Фасад
* **Приспособленец**
* Заместитель

1. Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него, называется

* Одиночка
* Фасад
* Приспособленец
* **Заместитель**

1. Шаблон Цепочка обязанностей

* Представляет действие
* **Предназначен для организации в системе уровней ответственности**
* Решает часто встречающуюся, но подверженную изменениям, задачу
* Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации

1. Шаблон Команда

* **Представляет действие**
* Предназначен для организации в системе уровней ответственности
* Решает часто встречающуюся, но подверженную изменениям, задачу
* Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации

1. Шаблон Интерпретатор

* Представляет действие
* Предназначен для организации в системе уровней ответственности
* **Решает часто встречающуюся, но подверженную изменениям, задачу**
* Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации

1. Шаблон Итератор

* Представляет действие
* Предназначен для организации в системе уровней ответственности
* Решает часто встречающуюся, но подверженную изменениям, задачу
* **Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации**

1. Шаблон проектирования, который представляет действие, называется

* Цепочка обязанностей
* **Команда**
* Итератор
* Интерпретатор

1. Шаблон проектирования, который предназначен для организации в системе уровней ответственности, называется

* **Цепочка обязанностей**
* Команда
* Итератор
* Интерпретатор

1. Шаблон проектирования, который решает часто встречающуюся, но подверженную изменениям, задачу, называется

* Цепочка обязанностей
* Команда
* Итератор
* **Интерпретатор**

1. Объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации, называется

* Цепочка обязанностей
* Команда
* **Итератор**
* Интерпретатор

1. Шаблон Посредник

* Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях
* **Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга**
* Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния

1. Шаблон Хранитель

* **Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях**
* Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга
* Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния

1. Шаблон Наблюдатель

* Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях
* Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга
* **Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии**
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния

1. Шаблон Состояние

* Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях
* Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга
* Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии
* **Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния**

1. Шаблон, который позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях, называется

* Посредник
* **Хранитель**
* Наблюдатель
* Состояние

1. Шаблон, который обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга, называется

* **Посредник**
* Хранитель
* Наблюдатель
* Состояние

1. Шаблон, который определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии, называется

* Посредник
* Хранитель
* **Наблюдатель**
* Состояние

1. Шаблон, который используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния, называется

* Посредник
* Хранитель
* Наблюдатель
* **Состояние**

1. Шаблон Стратегия

* **Предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости**
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния
* Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
* Описывает операцию, которая выполняется над объектами других классов. При изменении такого класса нет необходимости изменять обслуживаемые классы

1. Шаблон «Шаблонный метод»

* Предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния
* **Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом**
* Описывает операцию, которая выполняется над объектами других классов. При изменении такого класса нет необходимости изменять обслуживаемые классы

1. Шаблон Посетитель

* Предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости
* Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния
* Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом
* **Описывает операцию, которая выполняется над объектами других классов. При изменении такого класса нет необходимости изменять обслуживаемые классы**

1. Шаблон, который предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости, называется

* Состояние
* **Стратегия**
* Шаблонный метод
* Посетитель

1. Шаблон, который описывает операцию, которая выполняется над объектами других классов, так что при изменении такого класса нет необходимости изменять обслуживаемые классы, называется

* Состояние
* Стратегия
* Шаблонный метод
* **Посетитель**

1. В 1970-е годы был составлен составил набор шаблонов проектирования в области архитектуры. Авторы или автор?

* Эрих Гамма
* Джеймс Коплин
* Кент Бэк и Вард Каннигем
* **Кристофер Александр**

1. В 1991 году опубликована книга «Advanced C++ Idioms». Авторы или автор?

* Эрих Гамма
* **Джеймс Коплин**
* Кент Бэк и Вард Каннигем
* Кристофер Александр

1. В 1991 году опубликована книга «Приемы объектно-ориентированного проектирования. Паттерны проектирования». Авторы или автор?

* **Банда Четырех**
* Джеймс Коплин
* Кент Бэк и Вард Каннигем
* Кристофер Александр

1. Паттерн Стратегия является

* Основным
* Порождающим
* Структурным
* **Поведенческим**

1. Паттерн Одиночка является

* Основным
* **Порождающим**
* Структурным
* Поведенческим

1. Паттерн Адаптер является

* Основным
* Порождающим
* **Структурным**
* Поведенческим

1. Паттерн Посетитель является

* Основным
* Порождающим
* Структурным
* **Поведенческим**

1. Паттерн Приспособленец является

* Основным
* Порождающим
* **Структурным**
* Поведенческим