1. В первом задании применён паттерн «Адаптер», который позволяет классам с несовместимыми интерфейсами работать вместе. В данном случае два датчика температуры работают с разными шкалами температуры, но с помощью адаптера могут работать вместе
2. Применён паттерн «Абстрактная фабрика», который предоставляет интерфейс для создания семейств взаимосвязных объектов без указания их конкретных классов. Новый автомобиль легко добавляется без изменения существующего кода, а только добавлением новых классов
3. Паттерн «Фабричный метод» определяет интерфейс для создания объекта, но позволяет подклассам изменять тип создаваемого объекта. Классы новой услуги и транспортной компании добавляются без изменения существующего кода.
4. «Одиночка» обеспечивает существование единственного экземпляра класса. Добавлено статическое приватное поле и открытое статическое свойство для создания экземпляра в случае первого обращения или возвращения существующего объекта
5. «Стратегия» позволяет отделить алгоритмы прокладки маршрутов от основного класса навигатора, что позволяет добавлять новые алгоритмы, не меняя существующего кода
6. «Шаблонный метод» определяет каркас алгоритма в базовом классе, делегируя реализацию отдельных шагов подклассам
7. «Фасад» создаёт простой интерфейс к сложной подсистеме
8. «Цепочка обязанностей» передаёт запрос по цепочке обработчиков, где каждый принимает решение о его обработке. Изменение порядка вызова напрямую влияет на результат
9. «Команда» инкапсулирует запрос как объект, позволяя параметризовать объекты запросами, ставить их в очередь и отменять
10. «Декоратор» позволяет добавлять новые функциональности, не меняя их исходный класс. Уменьшает зависимость между объектами