Часть 1
N1.2
Абстрактная фабрика представляет собой порождающий паттерн проектирования, который предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов без указания их конкретных классов.
Инкапсулирует:
Создание семейства связанных объектов.
Определение интерфейса для создания объектов без указания их конкретных классов.
Изменяемые аспекты:
Можно добавлять новые семейства продуктов, не изменяя существующий код.
Можно изменять конкретные классы продуктов, не затрагивая клиентский код.
Пример:
Рассмотрим создание GUI-элементов (кнопок, полей ввода) для разных операционных систем (Windows, macOS).
AbstractFactory определяет методы createButton() и createInputField(), которые реализуют соответствующие конкретные фабрики для каждой ОС.

Часть 2

N2.3
Инкапсуляция - это механизм ограничения доступа к членам класса и скрытия деталей его реализации от внешнего мира. Это позволяет изменять внутреннюю реализацию класса без влияния на его внешнее использование.

N2.6
- Истинный полиморфизм: Это способность объектов разных типов отвечать на одну и ту же операцию (метод) с учетом их конкретного типа.
- Псевдо полиморфизм: В контексте языков программирования, не поддерживающих полноценный полиморфизм, псевдополиморфизм может означать использование других механизмов (например, перегрузка функций) для эмуляции полиморфных поведений.

N2.9
- Определение: Моделирование в IT - это процесс создания абстрактных представлений системы или ее компонентов для облегчения понимания, анализа и документирования системы.
- Примеры: UML-диаграммы (классов, вариантов использования, последовательности), диаграммы ER (сущность-связь) в базах данных и т.д.

N2.1
- Определение: Процедурная декомпозиция - это методология разработки, при которой задача разбивается на более мелкие подзадачи, которые затем решаются отдельно.
- Примеры: Разделение большой программы на функции или процедуры, каждая из которых отвечает за определенный функциональный блок.

Часть 3

N3.5
Определение: Unit testing - это процесс проверки отдельных модулей (или компонентов) программы на корректность их работы.
Варианты использования:
- Проверка функциональности: Убеждение, что каждый модуль выполняет свою функцию правильно.
- Регрессионное тестирование: Проверка, что изменения в коде не влияют на существующую функциональность.
- Автоматизированное тестирование: Использование инструментов для автоматизации запуска и анализа результатов тестов.

N3.7

- DataProvider - это механизм, предоставляемый некоторыми тестовыми фреймворками, для подачи в тест различных наборов входных данных.
- Примеры: В Java, аннотация @DataProvider в TestNG позволяет методу поставлять данные для тестов из разных источников (например, массивов, файлов).

N3.10

- Generic - это механизм в Java, который позволяет создавать классы, интерфейсы и методы с параметризованными типами данных.
- Использование в контексте Mock frameworks: Позволяет создавать универсальные моки (заглушки) для различных типов объектов. Например, Mockito может создавать моки с использованием generic типов.

N3.2

Цикломатическая сложность - это одна из метрик программного обеспечения. Она измеряет сложность программы на основе количества линейно независимых путей в ее коде. Чем выше цикломатическая сложность, тем сложнее поддерживать, тестировать и понимать код.
M=E–N+2P
Где:

M - цикломатическая сложность.
E - количество ребер в графе контроля потока программы.M
N - количество узлов в графе контроля потока программы.
P - количество связных компонентов в графе контроля потока программы.