**Отчёт по тестам**

В ходе работы над проектом были разработаны тесты для нескольких ключевых компонентов системы. Основной целью тестов было убедиться в корректности работы различных частей системы, таких как обработка платежей, уведомлений и обработка заказов.  
  
1. Тесты для платёжных адаптеров:  
 - Были протестированы классы PaymentAdapter, PayPalPaymentAdapter и CreditCardPaymentAdapter, которые реализуют интерфейс для обработки различных типов платежей.  
 - Тесты были написаны для:  
 - Проверки корректности обработки платежей через разные платёжные системы.  
 - Проверки корректности работы метода process\_payment:  
 - Для PayPalPaymentAdapter проверяли, что система возвращает правильное сообщение о платеже через PayPal.  
 - Для CreditCardPaymentAdapter проверяли, что система возвращает правильное сообщение о платеже через кредитную карту.  
 - Дополнительно был протестирован PaymentAdapter как абстрактный класс. Было проверено, что при попытке вызвать метод process\_payment() на базовом классе PaymentAdapter, который не реализует метод, выбрасывается исключение NotImplementedError.  
  
2. Тесты для уведомлений:  
 - Были написаны тесты для NotificationFactory, которая отвечает за создание объектов уведомлений разных типов (например, email и SMS).  
 - Тесты были направлены на проверку:  
 - Корректности создания объектов уведомлений через фабрику:  
 - Для EmailNotification проверяли, что фабрика создает объект правильного типа и метод send() возвращает сообщение о том, что email был отправлен.  
 - Для SMSNotification проверяли, что фабрика создаёт объект правильного типа и метод send() возвращает сообщение о том, что SMS было отправлено.  
 - Проверка обработки некорректных типов уведомлений:  
 - Тестировалось, что при передаче некорректного типа уведомления (например, "push") выбрасывается исключение ValueError с сообщением об ошибке.  
  
3. Тесты для обработки заказов:  
 - Были протестированы основные этапы обработки заказов, такие как:  
 - Проверка наличия товара:  
 - Для заказа, в котором товар отсутствует, была проверена корректность работы обработки в цепочке обязанностей с возвратом сообщения "Product out of stock".  
 - Проверка оплаты:  
 - Тестировалась корректность обработки неоплаченных заказов, с возвращением сообщения "Order payment pending".  
 - Отправка товара:  
 - Для успешно оплаченного заказа и с наличием товара, тестировалась корректность завершения заказа с отправкой товара и возвратом сообщения "Order is being shipped".  
  
4. Тесты для функции обработки платежей process\_order\_payment:  
 - Были написаны тесты для функции process\_order\_payment(), которая использует адаптеры для различных платёжных методов.  
 - Тесты проверяли, что функция корректно обрабатывает:  
 - Платёж через PayPal.  
 - Платёж через Credit Card.  
 - Обработку неизвестных платёжных методов, что приводит к выбросу ValueError.  
  
5. Использование unittest:  
 - Все тесты были организованы с помощью unittest, что позволило:  
 - Автоматически запускать и проверять тесты.  
 - Быстро выявлять ошибки в различных частях системы.  
 - Обеспечить изоляцию тестов, чтобы изменения в одной части системы не влияли на другие её компоненты.  
  
6. Преимущества тестирования:  
 - Надёжность: Тестирование позволило убедиться, что ключевые части системы, такие как обработка заказов, платёжные адаптеры и уведомления, работают корректно.  
 - Обнаружение ошибок на ранних этапах: Были выявлены потенциальные проблемы с обработкой платежей через неоплаченные заказы и неправильным выбором типов уведомлений.  
 - Легкость в расширении: Тесты обеспечили гибкость для будущих изменений в системе, таких как добавление новых платёжных методов или типов уведомлений, без нарушения существующего функционала.  
  
Тестирование было важной частью разработки системы, обеспечившей её надёжность и стабильную работу в реальных условиях.