# Тестирование ПО

05.04.2023

# Цели тестирования

- Обнаружение дефектов программного кода
- Уменьшение рисков
- Улучшение качества ПО
- Сокращение времени разработки

# Уровни тестирования

- Unit testing
  - Проверяют микрофункциональность в рамках данного класса, методы класса (в случае ООП)
  - Если используется принцип SOLID, то класс отвечает только за 1 ответственность. При unit-тестировании тест «не полезет» в другие классы (а чаще всего будет использовать mock-объекты, чтобы реализовать принцип разделения)
- Integration testing
  - Проверяет взаимодействие между разными модулями приложения
  - Цель – проверить, что все компоненты, работающие по-отдельности, вместе выдают ожидаемый результат
  - Параллельно можно проводить performance/functional testing, etc.
- System testing
  - Цель – проверить, что вся система работает, как это задумано
  - Проверяются различные сценарии/взаимодействия с конечным пользователем
- Acceptance testing
  - Проверяет, что ПО соответствует требованиям и готово для релиза

# Mocks, Stubs

- Чаще всего используются на уровне unit-тестов и с ростом уровня тестирования используется меньше.
  - Больше используются реальные объекты и сервисы
- В классическом случае вызовы другого класса «мокаются»
- Популярный фреймворк – **Mockito**
- Stub – урезанный объект

# Типы тестов

- Функциональное тестирование
  - Проверяет, что ПО удовлетворяет заявленным функциональным требованиям (указаны в документации, user stories)
  - Может быть ручным/автоматизированным (JUnit, Selenium, TestNG)
- Нефункциональное тестирование
  - Проверяет производительность, масштабируемость, надежность, удобство использования, безопасность
  - Акцент на характеристиках приложения, а не на фичах/функциональности
  - JMeter, Gatling, LoadRunner
- Регресс-тестирование
  - Как правило, перед каждым релизом происходит регресс-тестирование
  - Проверяет, работает ли продукт так же, как и перед добавлением новой фичи
  - Прогон всех сценариев (функциональности)
  - JUnit, Selenium, TestNG

# Quality Assurance (QA)

- QA engineer:
  - Отвечает за то, чтобы ПО удовлетворяло определённым требованиям и стандартам качества
  - Создает тестовые планы (**test suites**) и кейсы (**test cases**) и с их помощью обнаруживает дефекты/баги
  - Работают вплотную с разработчиками
  - Стремится к тому, чтобы ПО было качественным, надёжным и удовлетворяло требованиям конечных пользователей

# Немного терминологии

- **Test suite** – набор кейсов, которые тестируют одну функциональность
- **Test case** – конкретный сценарий теста:
  - **Preconditions** – конфигурации, что настроить для тестирования (среда, данные)
  - Steps – конкретные шаги, которые надо исполнить во время теста
  - **Expected result** vs **Actual result**
  - **Clean-up**

# Фреймворки

- JUnit
  - Open source, интегрируется с Maven & Gradle
  - Понятный & простой API, широкий выбор методов- assertions для проверки ожидаемых значений
  - Встречается чаще всего
- TestNG
  - Более совершенные методы, нежели чем в Junit (Data-driven development, etc.)
- Mockito
  - Предоставляет простой API для создания mock/stub-объектов
  - Позволяет изолировать части кода для тестирования и симулировать поведение зависимостей (других объектов других классов)
  - Интегрируется с Junit и TestNG
- Selenium – для end-to-end тестов с помощью браузера