Представим систему для онлайн-банкинга, где пользователи могут выполнять три основные операции: переводы средств между счетами, проверка баланса и оплата услуг. Мы разрабатываем backend для этой системы, состоящий из двух основных классов:

1. **Класс Account** (представляет банковский счет)
   * **Метод** getBalance(): возвращает текущий баланс счета.
   * **Метод** withdraw(amount): снимает указанную сумму со счета.
   * **Метод** deposit(amount): добавляет указанную сумму на счет.
2. **Класс BankService** (представляет банковский сервис)
   * **Метод** transfer(fromAccount, toAccount, amount): переводит средства с одного счета на другой.
   * **Метод** payBill(account, amount, biller): оплачивает услуги.
   * **Метод** getAccountBalance(account): возвращает баланс указанного счета (вызывает метод getBalance класса Account).

**Бизнес-требования:**

1. Пользователи должны иметь возможность проверять баланс своих счетов в любой момент.
2. Пользователи могут переводить средства между своими счетами и на счета других пользователей.
3. Пользователи могут оплачивать различные услуги (например, коммунальные платежи, мобильную связь) через систему.

**Уровни тестирования:**

**1. Модульное тестирование (Unit Testing)**

Модульные тесты проверяют отдельные методы классов Account и BankService на корректность выполнения.

**2. Интеграционное тестирование (Integration Testing)**

Интеграционное тестирование проверяет взаимодействие между классами Account и BankService. Например, убедимся, что при выполнении перевода между счетами корректно обновляются балансы обоих счетов.

**Описание интеграционного тестирования:**

* Создаем два счета с разными начальными балансами.
* Используем метод transfer для перевода средств с одного счета на другой.
* Проверяем, что баланс первого счета уменьшился, а второго — увеличился на правильную сумму.

**Тестовые сценарии:**

* Перевод суммы, равной балансу счета.
* Перевод суммы, превышающей баланс счета (ожидаем ошибку).
* Перевод суммы 0 (ожидаем, что баланс обоих счетов не изменится).

**3. Системное тестирование (System Testing)**

Системное тестирование проверяет систему в целом, включая взаимодействие с внешними сервисами (например, сервисы оплаты). Оно включает проверку всей бизнес-логики и пользовательских сценариев.

**Описание системного тестирования:**

* Проверка всех функций системы через интеграцию с внешними системами (например, банковскими платежными шлюзами, системами оплаты услуг).
* Тестирование сценариев использования: пользователь проверяет баланс, переводит деньги между счетами, оплачивает счета.
* Проверка на предмет отказоустойчивости (например, что произойдет, если во время перевода денег система потеряет связь с сервером).

**4. Приемочное тестирование (Acceptance Testing)**

Приемочное тестирование проверяет выполнение системы в соответствии с бизнес-требованиями. Этот уровень тестирования проводится в условиях, приближенных к реальному использованию, и включает сценарии, основанные на пользовательских историях.

### Заключение

Модульные тесты проверяют работу отдельных методов, интеграционные тесты проверяют взаимодействие между классами, системные тесты проверяют всю систему в целом, а приемочные тесты проверяют соответствие системы бизнес-требованиям. Такой подход обеспечивает всестороннее тестирование и минимизирует риск появления ошибок в коде.