Архитектура и шаблоны проектирования

ЗАЩИТА ПРОЕКТА

Тема: «Разработка прототипа антифрод-системы»

Рябцев Владимир

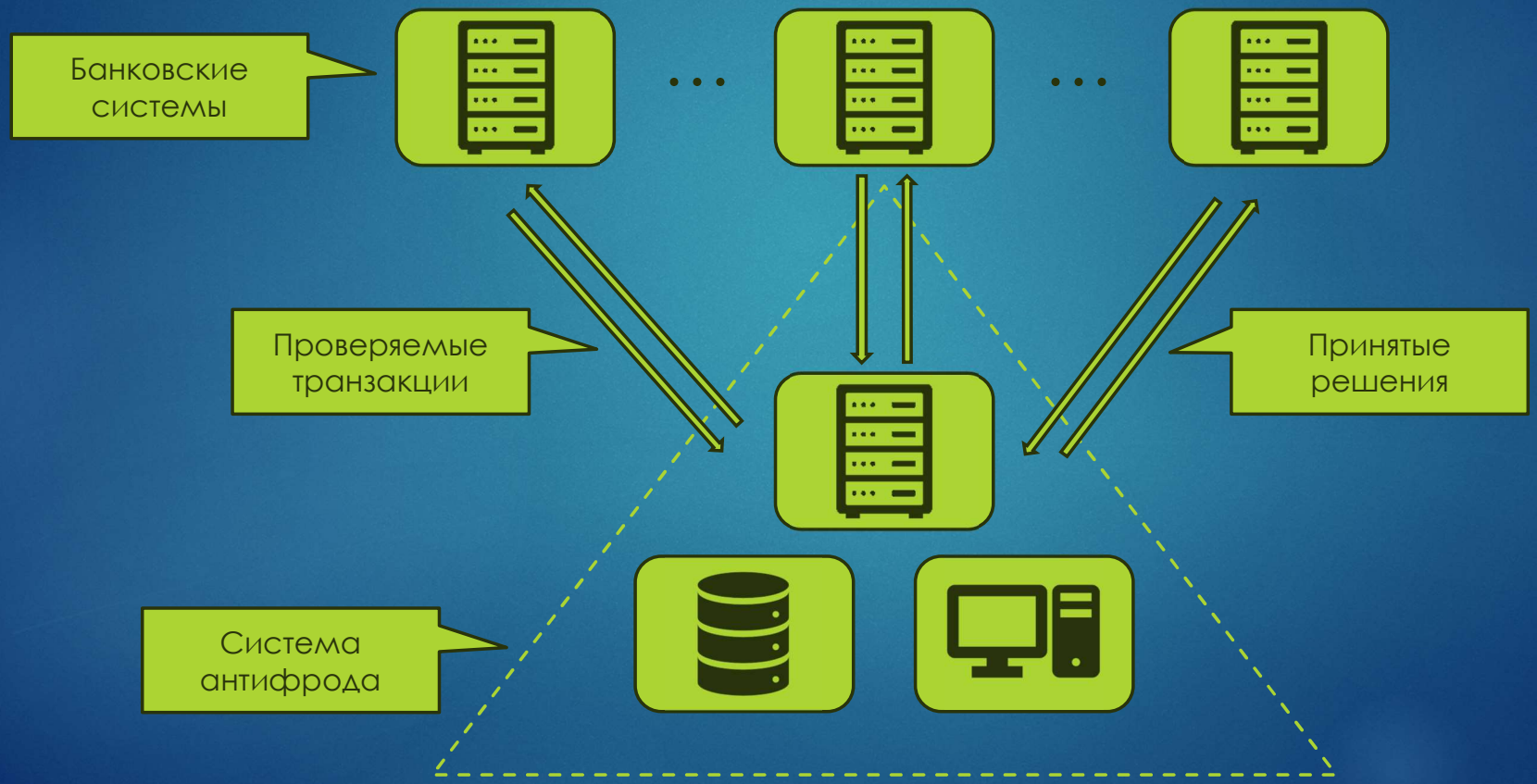
Главный инженер по разработке
ПАО «Сбербанк»

Меня хорошо видно и слышно?

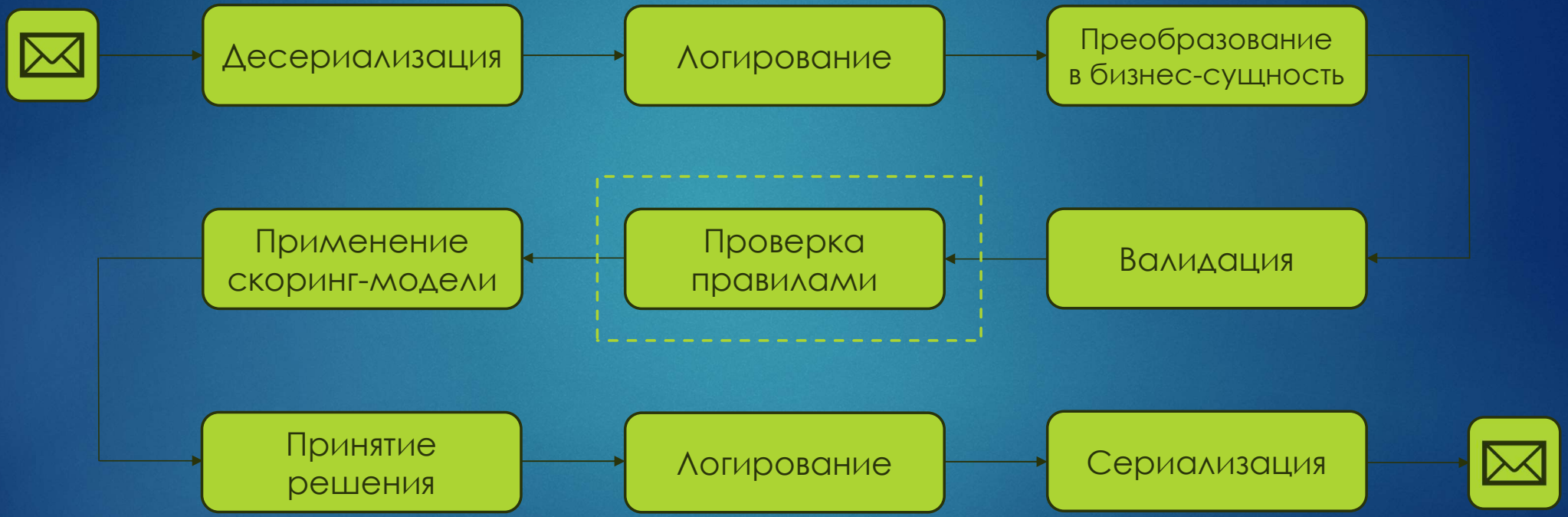
Мотивация

Практически всё время прохождения обучения работал в компании, занимавшейся, в частности, разработкой и интеграцией антифрод-решений. Было интересно предложить альтернативу тому решению, которое было на проекте, на котором работал я.

Общая схема реального решения



Общая схема обработки транзакции



Источники роста сложности системы

Основными источниками сложности являются:

- добавление новых типов транзакций в существующие системы;
- постоянные требования к добавлению новых и изменению существующих правил проверки транзакций;
- возрастающее число систем, с которыми должна производится интеграция.

Последствия добавления новых типов транзакций

Реальный код правила

```
String messageTriggered = null;
if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = process[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    messageTriggered = processCrossBorder[redacted](transaction);
} else if (transaction instanceof [redacted]) {
    final [redacted] payment = ([redacted]) transaction;
    messageTriggered = process[redacted](payment.getBik(), payment.getPayeeAccount(), payment.getPayeeInn());
}
```

Изменённый код того же правила

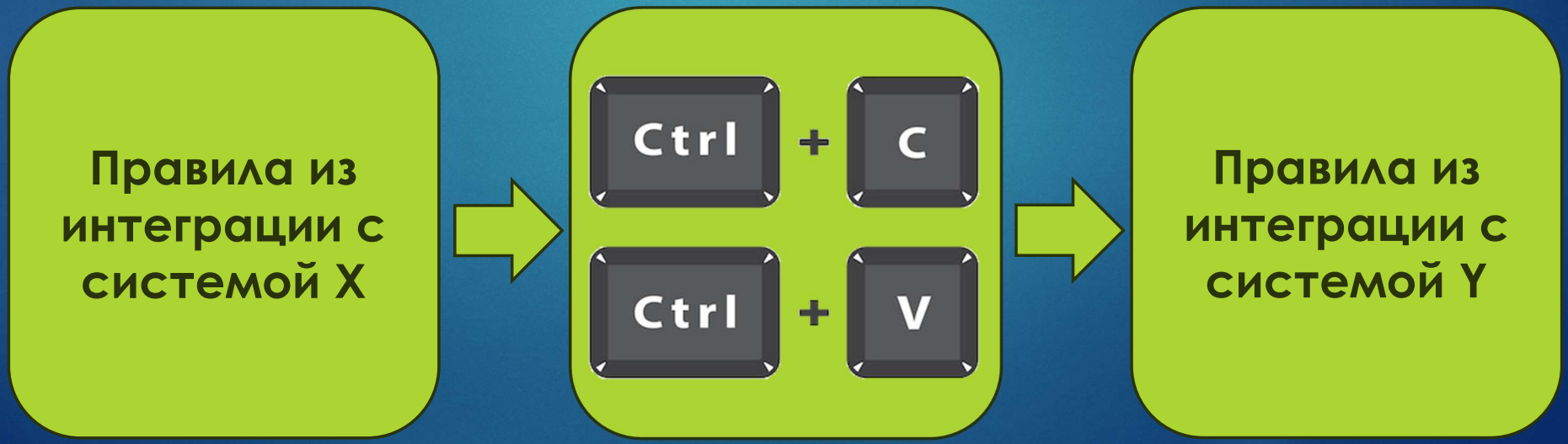
```
private String processPayment(final Transaction transaction) {
    final Payment payment = (Payment) transaction;
    if (null != entitiesCache.getEntityFromCache(payment.getPayeeINN(), BlackListINN.class)) {
        return RESULT_BLOCK_INN;
    }
    return null;
}

private String processAccount(final Transaction transaction) {
    final Account account = (Account) transaction;
    if (null != entitiesCache.getEntityFromCache(account.getPayeeINN(), BlackListINN.class)) {
        return RESULT_BLOCK_INN;
    }
    return null;
}
```

Добавление новых типов транзакций ведёт к разрастанию кода классов, ухудшению его читаемости и сложностям с дальнейшей поддержкой правил.

Последствия добавления новых интеграций

Однажды заказчику понадобилось добавить интеграцию с системой Y, при этом транзакции, поступающие из данного источника, должны обрабатываться аналогично транзакциям, поступающим из системы X.

Использованное решение:



Цели проекта

Разработка прототипа антифрод-системы, позволяющего:

- принимать данные о транзакциях, полученные от различных банковских систем;
- обрабатывать полученные данные согласно правилам;
- присваивать транзакциям балл, на основании которого по транзакциям принимается решение;
- принимать решение по транзакции, используя модели принятия решений, обрабатывающих оценку, полученную в предыдущем пункте;
- сохранять данные транзакций;
- отправлять решения, принятые по транзакциям, в те системы, от которых поступили исходные данные по транзакциям.

Поставленные задачи

Для достижения целей проекта были сформулированы следующие задачи:

- 1. Использовать полученные в ходе изучения курса знания для создания аналога элементов реальной антифрод-системы.
- 2. Произвести разработку решения, опираясь на принципы SOLID, в особенности на OCP и ISP.
- 3. Предложить универсальные решения, позволяющие бороться с возрастающей сложностью разработки системы.

Использованные технологии

При решении поставленной задачи были использованы:

- Java 17;
- Spring Framework 3.2.0.

Применённые паттерны проектирования

В ходе решения поставленной задачи были использованы следующие паттерны проектирования:

- декоратор;
- цепочка ответственностей;
- фабричный метод;
- строитель.

Полученные результаты

В результате проделанной работе получилось:

- разработать более устойчивую к добавлению новых типов транзакций и новых интеграций систему обработки транзакций правилами, основанную не на конкретных данных, а на «свойствах» (traits) транзакций (как например, возможность предоставления данных о телефоне, ИНН получателя) представленных соответствующими интерфейсами.
- разработать модель реализации правил, основанную на концепции «условных действий», облегчающую тестирование и повторное использование кода.

Ссылка на репозиторий:
https://github.com/TrueMerc/otus-design-patterns-coursework

Выводы и планы по развитию

В ходе проведённой работы удалось достичь следующих результатов:

- реализовать более устойчивую к добавлению новых типов транзакций и новых интеграций систему обработки транзакций правилами;
- реализовать более устойчивую к изменениям модель реализации правил, облегчающую тестирование и изменение кода в соответствии с требованиями заказчика;
- полученные результаты удалось частично внедрить в реальный проект, но дальнейшее внедрение не представляется возможным в силу сложившихся обстоятельств.

Спасибо за внимание!