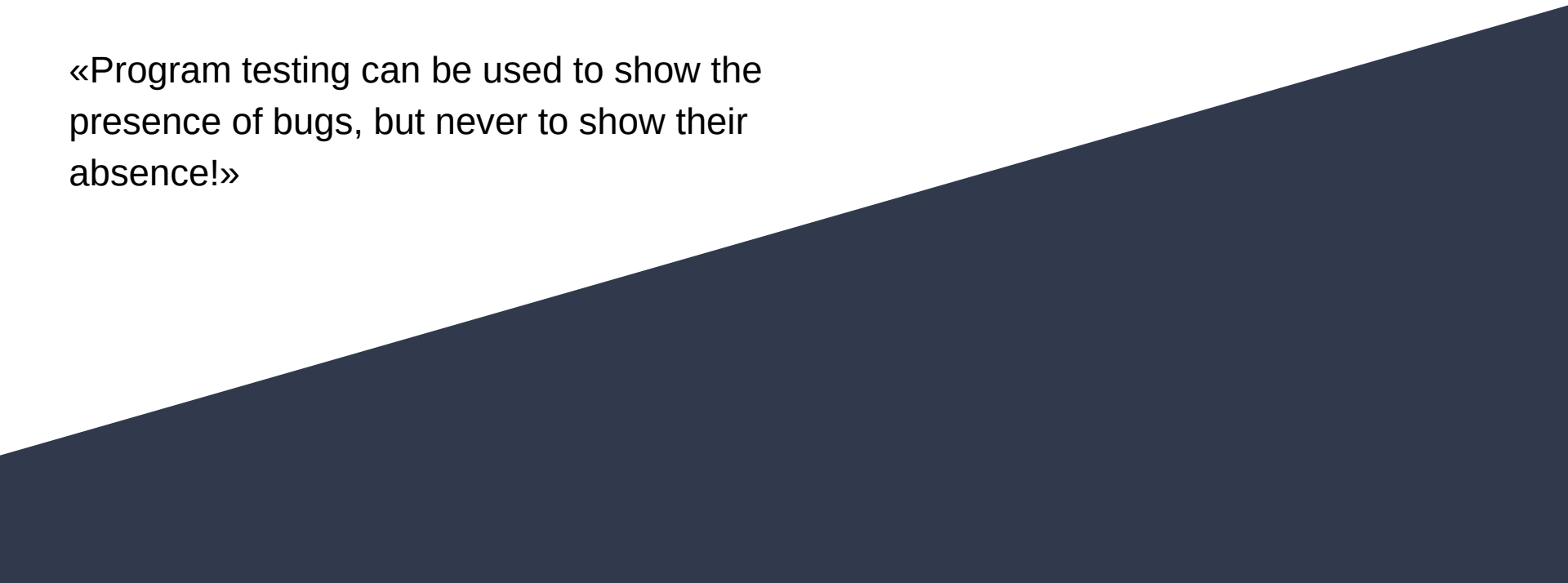
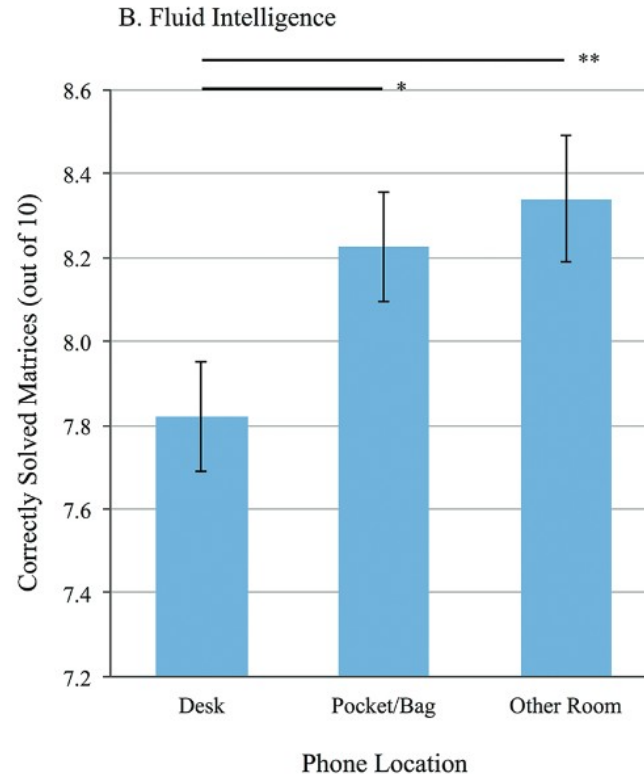
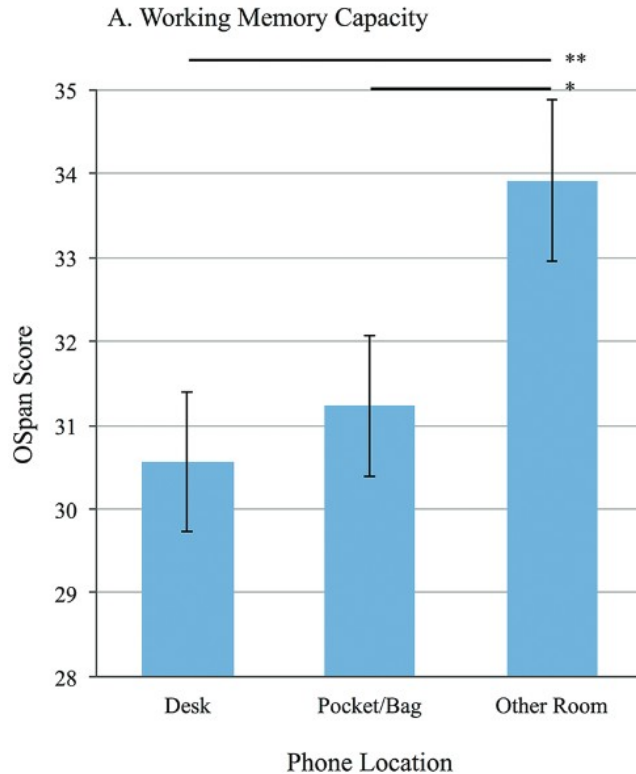


# Тестирование ПО: вводное занятие

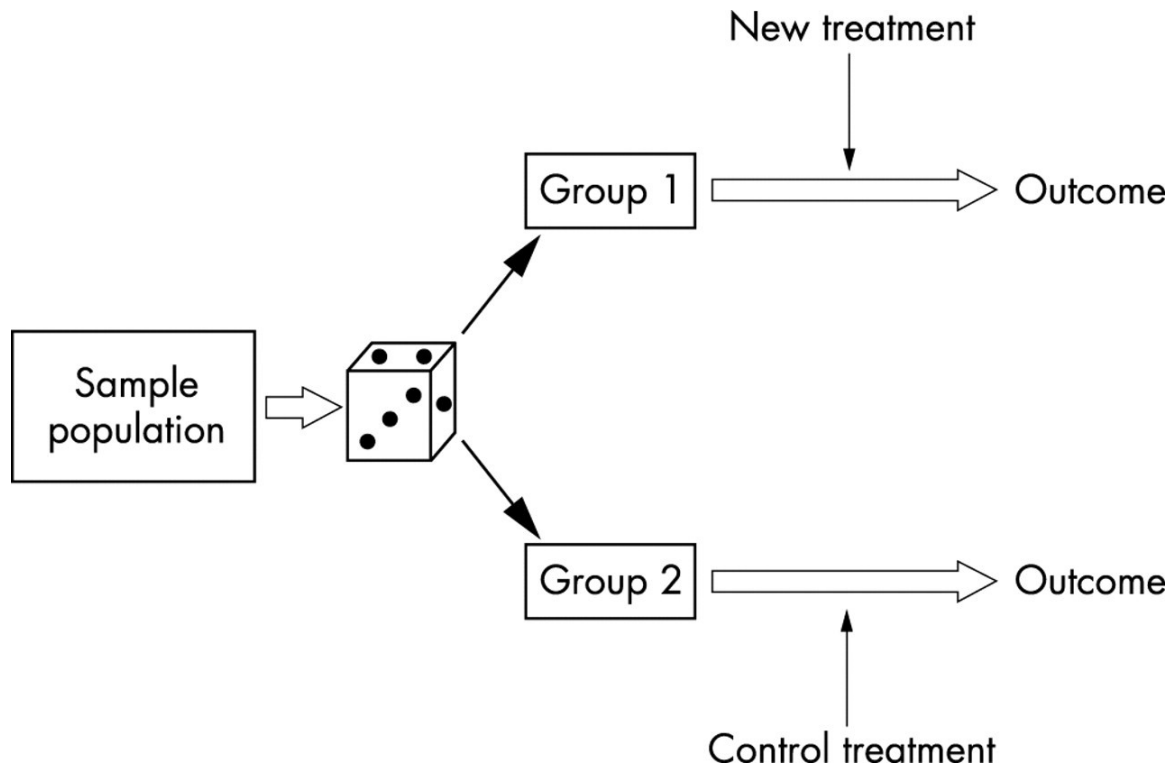
«Program testing can be used to show the presence of bugs, but never to show their absence!»

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

# Организационные моменты



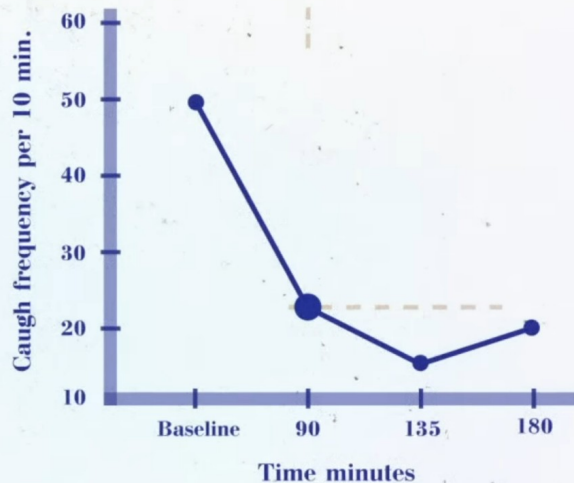
# Двойное слепое рандомизированное плацебо-контролируемое исследование



# WAAAGH!



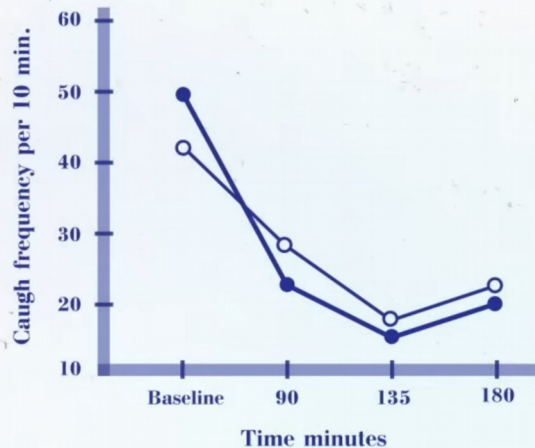
# Лекарство от кашля



**Fig. 1** Median cough frequency for patients with cough associated with URTI. A single dose for 30 mg dextromethorphan powder in a hard gelatin capsule or matched placebo containing lactose powder, was ingested by the patient with a small amount of water.

Placebo (○) n=22, and dextromethorphan (●) n=21, treatment groups.

# Лекарство от кашля и плацебо



**Fig. 1** Median cough frequency for patients with cough associated with URTI. A single dose for 30 mg dextromethorphan powder in a hard gelatin capsule or matched placebo containing lactose powder, was ingested by the patient with a small amount of water.

Placebo (○) n=22, and dextromethorphan (●) n=21, treatment groups.

# Калорийность еды

160 калорий



320 калорий



# Виды тестирования ПО

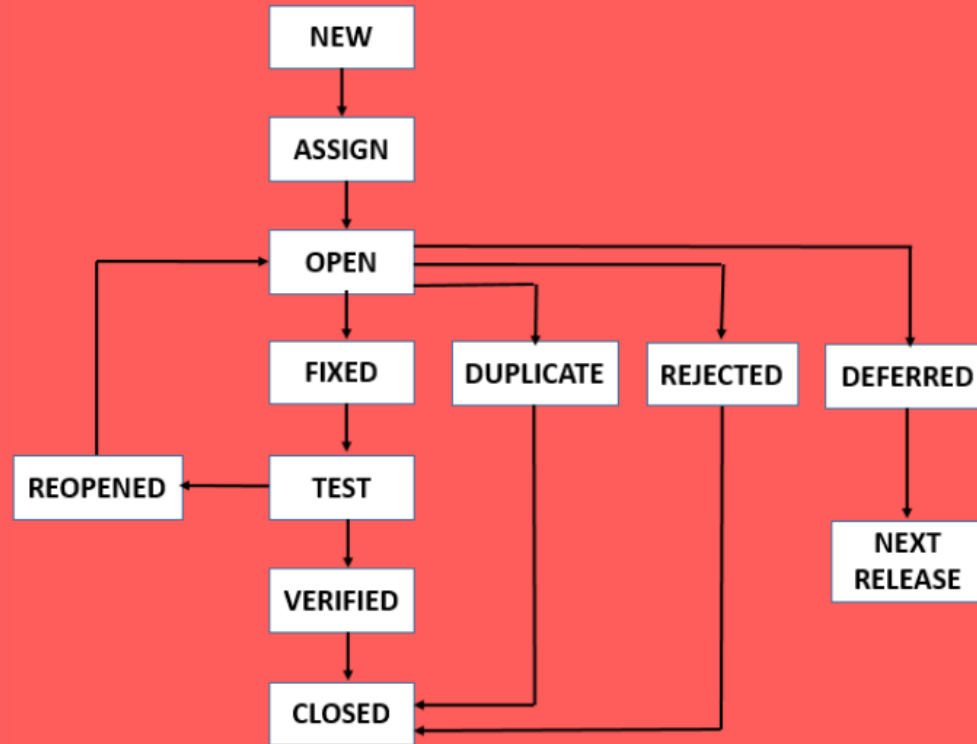
1. Функциональное тестирование
2. Тестирование производительности
3. Нагрузочное тестирование
4. Стресс-тестирование
5. Тестирование стабильности
6. Конфигурационное тестирование
7. Юзабилити-тестирование
8. Тестирование безопасности
9. Тестирование локализации
10. Тестирование совместимости



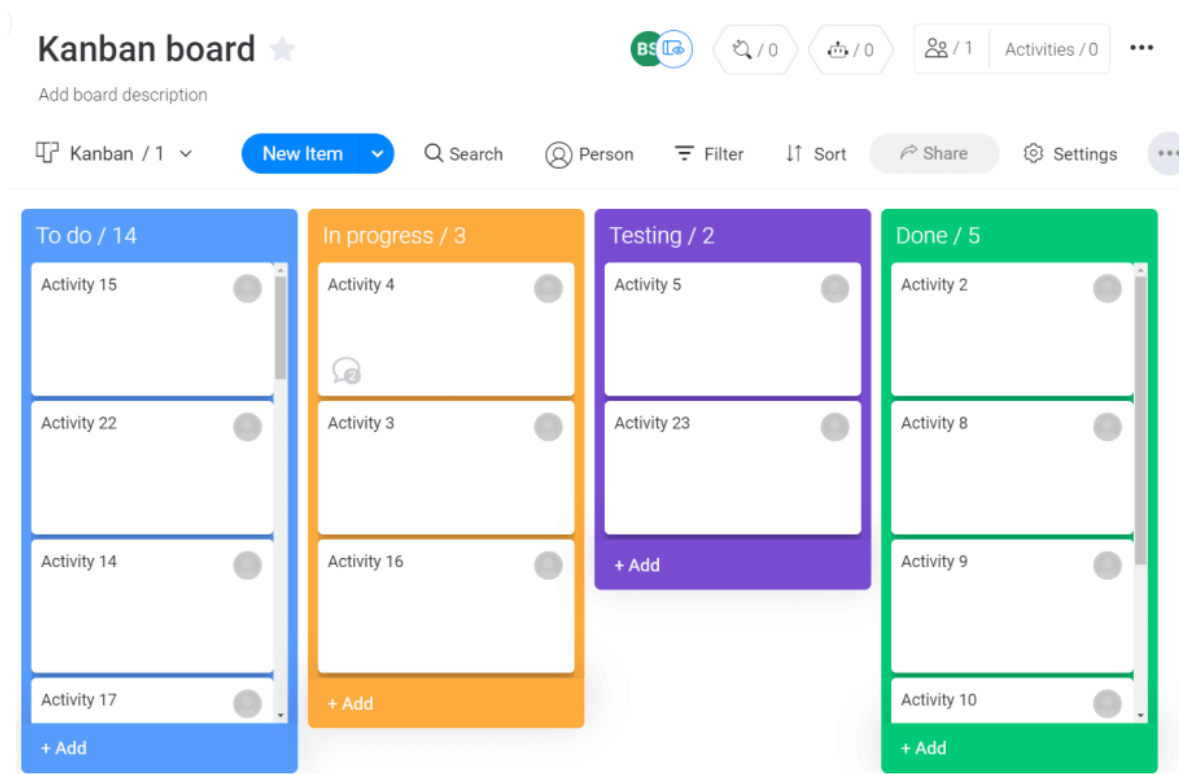
# По знанию внутреннего строения системы

- “тестирование черного ящика” (black box testing), когда исходный код системы недоступен и производится проверка соответствия системы формальной спецификации или ожидаемому поведению; **Проверка с точки зрения пользователя.**
- “тестирование белого ящика” (white box testing), когда для тестирования доступен ее исходный код; **Unit test как пример.**
- “тестирование серого ящика” (grey box testing), когда для тестирования доступен ее исходный код частично, например, некоторых компонентов или интерфейсы классов, либо код недоступен, но что-то известно про внутреннюю структуру (например, используемые алгоритмы).

# BUG LIFE CYCLE



# Канбан доска для визуализации



# Хороший багрепорт содержит информацию

- **when** (когда найден баг или при каких условиях он происходит);
- **where** (где найден баг);
- **why** (почему случается данный баг – предположение);
- и **how** (что, по мнению тестировщика, нужно сделать, чтобы исправить данный баг).

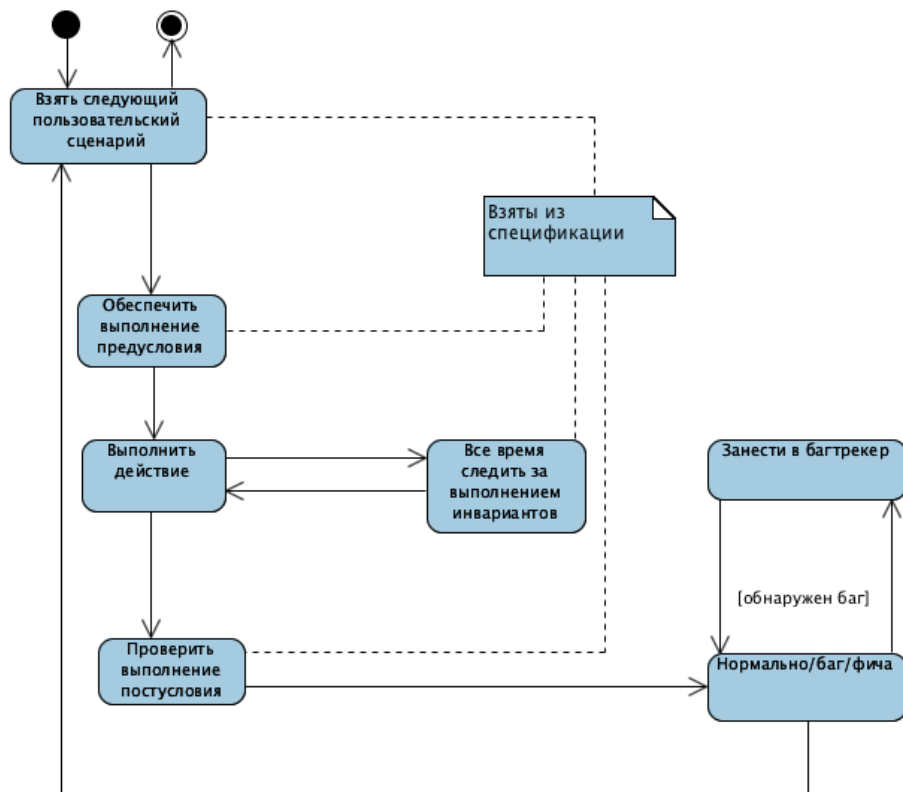
# Спецификация в виде сценариев

Предусловие – т. е. те условия, при которых возможно возникновение описываемого действия в системе.

Постусловие – результат выполнения действия, т. е. что получается в системе после того, как действие сработает и чего нам надо ожидать от действия.

Инвариант – глобальные условия в системе, которые постоянно выполняются на протяжении либо всей программы, либо заданного момента времени.

## Работа тестировщика



# Примеры сценариев

У пользователя должна быть какая-то реалистичная **цель**.

## Почтовый клиент:

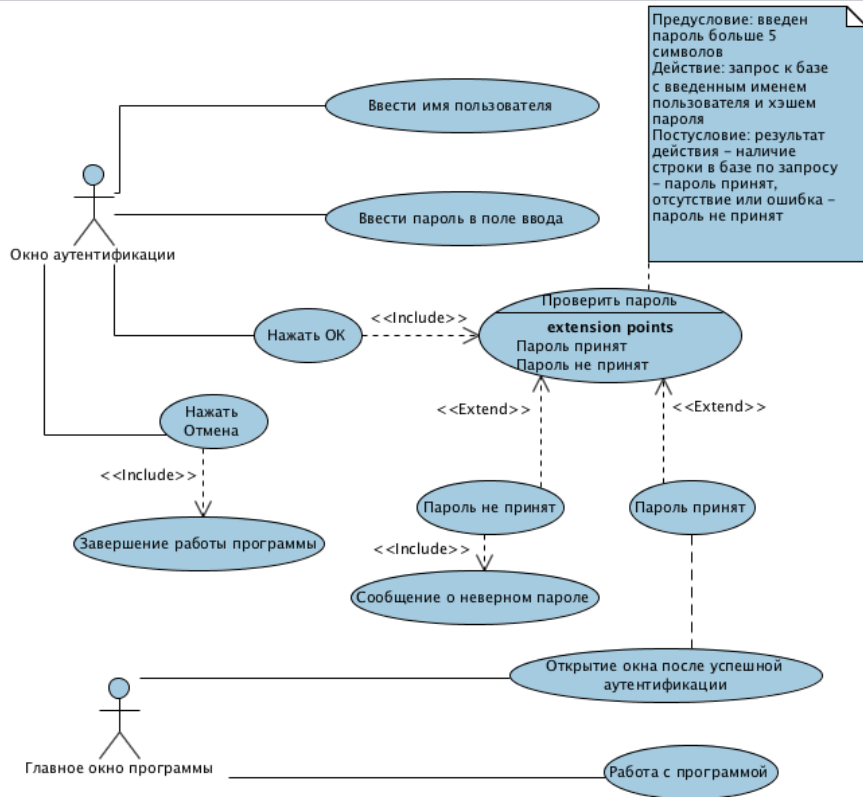
1. Добавление нового почтового ящика
2. Ответ на новое входящее сообщение

## Видеоплеер:

1. Просмотр фильма с выбором звуковой дорожки и субтитров
2. Прослушивание музыкального альбома с выбором определенного пресета эквалайзера.

**Важно:** посмотреть свойства объекта, или выйти из программы — цель слишком мелкая. Если мы на каждую кнопку в программе будем писать отдельный сценарий, их будет слишком много, а проверять/описывать они толком ничего не будут, потому как поведение программы важно не только в ответ на каждое действие в отдельности, но и в комплексе.

# Спецификация в виде Use Case-диаграмм UML





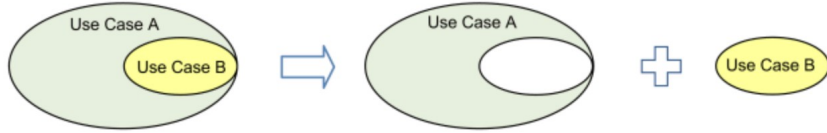
# Use Case

*Сценарий использования*, вариант использования, прецедент использования (англ. use case) — в разработке программного обеспечения и системном проектировании это описание поведения системы, **когда она взаимодействует с кем-то (или чем-то) из внешней среды**. Система может отвечать на внешние запросы Актора (англ. actor), может сама выступать инициатором взаимодействия. Другими словами, сценарий использования описывает, «кто» и «что» может сделать с рассматриваемой системой, или что система может сделать с «кем» или «чем».

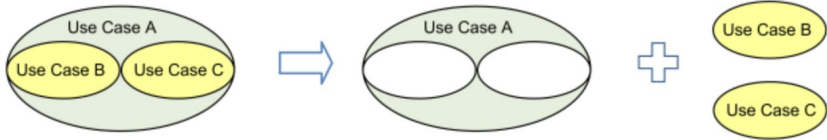
*Актор* — множество логически связанных ролей в UML, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Актором может быть человек или другая система, подсистема или класс, **которые представляют нечто вне сущности**.

Я не согласен с определением актора в методичке, но рисунок с предыдущего слайда имеет право на жизнь, если мы рассматриваем на диаграмме не всю систему целиком, а какую-то часть подробно, какую-то абстрактно, тогда можно сделать поясняющую подпись к актору, для понимания, с каким окном он в данном случае работает.

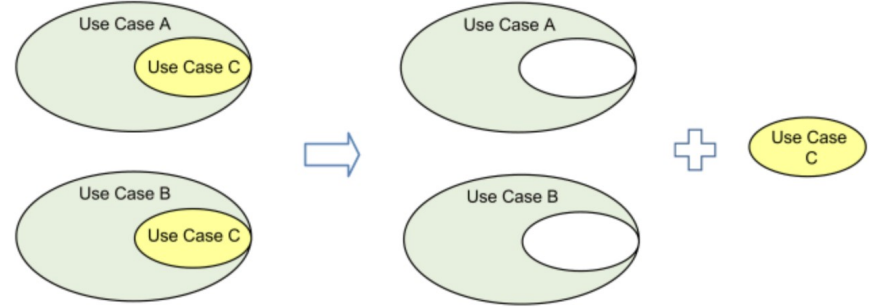
# Include



*Use case B is extracted from larger use case A into a separate use case.*

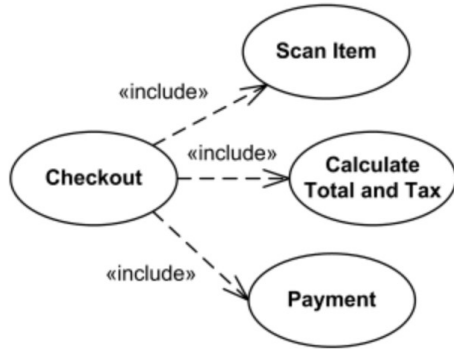


*Use cases B and C are extracted from larger use case A into separate use cases.*

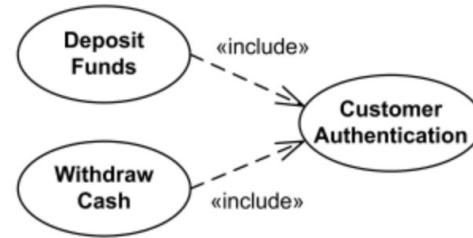


*Use case C is extracted from use cases A and B to be reused by both use cases using UML include relationship.*

# Include

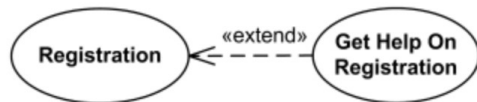


*Checkout use case includes several use cases - Scan Item, Calculate Total and Tax, and Payment*

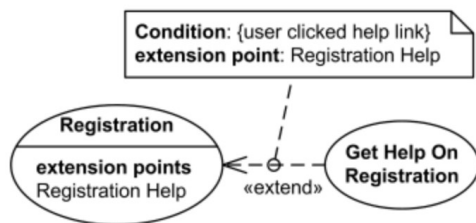


*Deposit Funds and Withdraw Cash use cases include Customer Authentication use case.*

# Extend

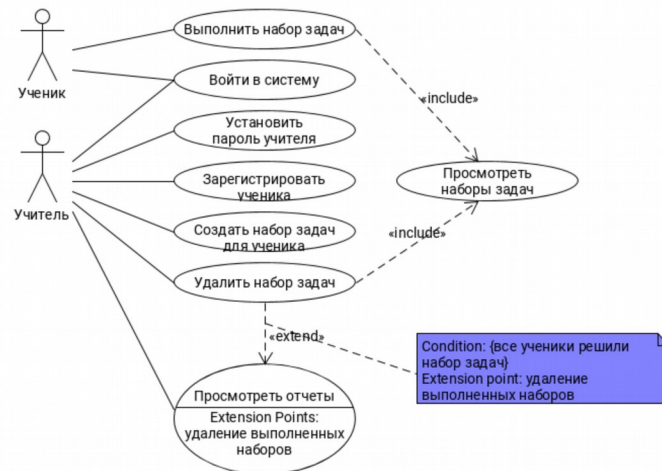


*Registration use case is complete and meaningful on its own.  
It could be extended with optional Get Help On Registration use case.*



*Registration use case is conditionally extended by Get Help On Registration use case in extension point Registration Help.*

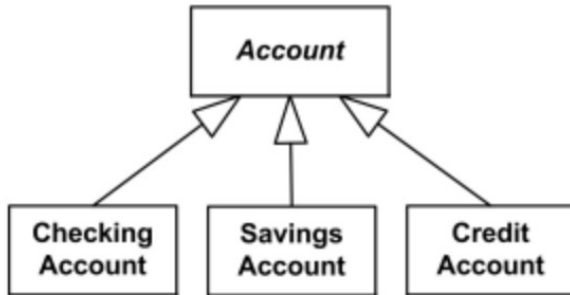
обязательным для выполнения основного прецедента. На диаграмме такой вид отношения изображается стрелкой, направленной к расширяемому прецеденту, в отдельном разделе которого может быть описана **точка расширения**, а **условия расширения** могут быть приведены в комментарии с ключевым словом **Condition**. Таким образом, расширение позволяет моделировать **необязательное поведение системы**, которое является **условным** и не **изменяет поведение** основного прецедента. Например отношение расширения нужно применить, если по техническому заданию **требуется возможность** удаления набора задач в прецеденте просмотра отчетов при условии, что все ученики решили этот набор.



Отношение расширения на диаграмме использования

Таким образом можно показать, что у учителя появляется возможность (но не обязанность) удалить набор задач при просмотре отчетов если все ученики выполнили этот набор.

# Наследование (generalization)



*Checking, Savings, and Credit Accounts are **generalized** by Account*

# Общие рекомендации:

1. Не нужно засорять диаграмму слишком мелкими действиями. Объедините все общие действия в одну группу под общим названием, чтобы было просто читать диаграмму.
2. Старайтесь не допускать пересечений соединительных линий. Это может затруднить чтение диаграммы для вас и для ваших коллег.
3. Не дублируйте варианты использования на диаграмме. Если приходится дублировать варианты использования, то элементы диаграммы надо постараться расставить по-другому.

# Ссылки

- [Биржа США провела кошмарное IPO. Акции рухнули на 99,9% за 2 секунды](#)
- [Почему айфон перезагружается от арабской смс](#)
- [Использование диаграммы вариантов использования UML при проектировании программного обеспечения](#)
- [Основы UML](#)
- <https://www.uml-diagrams.org/use-case-include.html>
- <https://www.uml-diagrams.org/use-case-extend.html>
- [Fuzzing-тестирование + и ещё](#)
- [UML в виде кода](#)