# Завдання

Написати програму для проведення тестування. Адміністратор повинен мати можливість додавати в програму запитання (вказуючи при цьому варіанти відповідей і відзначаючи правильний варіант). Користувачі проходять тестування, відповідаючи на ці питання і отримуючи оцінку у вигляді відсотка правильних відповідей.

Бажано, щоб програма дозволяла перемикатися між тестами. Тобто адміністратор може мати можливість створювати нові тести і при додаванні питання вказувати, до якого тесту він ставиться, а користувач - вибирати потрібний тест зі списку.

Основні алгоритми: обчислення результату тестування

# Вступ

Метою даної курсової роботи є розробка оболонки системи тестування задля отримання результату накопичених знань у процесі навчання або підвищення кваліфікації опитуваним.

Мета проекту: створення системи, що наддасть змогу провести перевірку знань користувача, додавання нових опитувань та підготовку результатів після проходження.

Актуальністю даної теми є те, що тематичні тести дозволяють швидко і об'єктивно оцінити рівень знань учня з певної теми або в певній галузі науки. Вони також дозволяють оцінити великої кількості людей за малий проміжок часу.

# Аналіз предметної області

* 1. Опис предметної області

Дана програма відображає список тестів будь-якої тематики, кількості та складності. Формуються питання та їх варіанти відповідей.  
 Широке застосування комп'ютерних технологій у навчальному процесі висунуло завдання автоматизувати роботу компанії, комісії, навчальних закладів щодо перевірки знань опитуваних.

Для роботи системи була створена програма, що включає в себе базу даних MySQL.

Спроектована база даних містить повну систему контролю и збереження нових запитань. Вона заснована на реляційній моделі управління БД, тобто кожне запитання містить інформацію, що відноситься до одного тесту. Для зв'язку таблиць створюється ключове поле, яке дозволяє закріпити за кожним тестом питання та відповіді,  не вводячи повторювані дані. Для ефективного пошуку даних використовують [індекси](http://ua-referat.com/%D0%86%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%B8), які негайно дозволяють відшукати потрібний нам тест із загального списку.

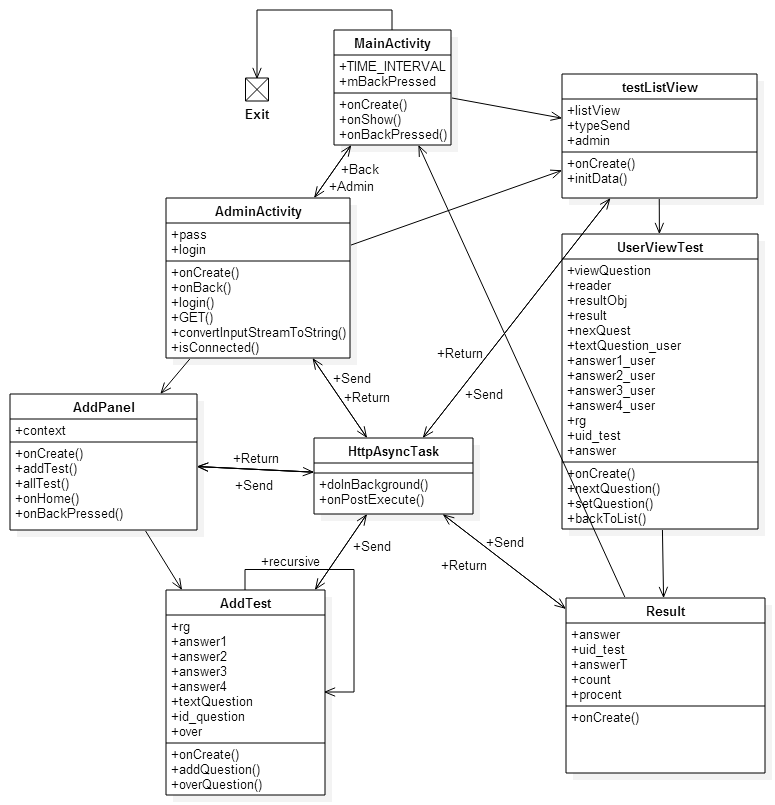
Окремо було виділенні вікна для адміністрування питання та проходження тестування опитуваними. Адміністраторська панель має вікно авторизації, без якого доступу до керування тестами не буде. Користувач має змогу із списку вибрати тест, а після закінчення отримати результат його проходження.

# Реалізація поставленої задачі

* 1. Формулювання вимог до програми

Програма повинна донести інформацію до користувача у зрозумілій і простій формі. Інформація представлена у вигляді фактів і простих даних статистики досліджень, що підвищує її сприйняття користувачем розробленої програми. Графічне супровід збільшує наочність.

  Програма повинна бути зрозуміла як досвідченому користувачеві, так і починаючому. Використані кнопки переходу до наступних сторінок, що полегшує орієнтування в програмі. Інформація поділена за розділами, що так само полегшує орієнтування. Не використовуються дратівливі очей кольору. У тесті діалог здійснюється за допомогою вибору відповідей із запропонованого списку. У програмі не використовувалися зайві компоненти для уникнення некоректних дій з боку користувача.



* 1. Опис об’єктної моделі

Front-End

Наша програма буде побудована на платформі Android за допомогою мови програмування Java. Всі активні класи будуть успадковувати головний клас – *Activity*. *Activity* – клас, що представляє візуальну активність додатки, і визначає дії, які може виробляти користувач. Графічна оболонка програми буде складатися з основних елементів керування та мати розташування на екрані за допомогою дерева-розмітки XML.

1. Першим та головним класом програми буде називатися *MainActivity.* Його основна задача – привітання у системі тестування та розподілення між вікнами адміністратора чи користувача.
2. При виборі вікна адміністратора він потрапить до панелі, котра запропонує йому, адміністратору, авторизуватися для подальшої роботи. Цей клас обробки буде мати назву *AdminActivity.*
3. Після авторизації, адміністратор потрапляє в панель керування, в котрому має змогу додати нові тести та питання до них, назначити коректну відповідь та переглянути список усіх тестів. Клас панелі - *AddPanel*
4. Додавання тесту складається з 2-ох етапів. Перший етап – діалогове вікно з можливістю вводу назви тестую. Другий етап – додавання питання та варіантів відповідей. Завершенням цих етапів є кнопка підтвердження закінчення усіх дій. Клас додання тестів – *AddTest*
5. Також у адміністратора є можливість переглянути список існуючих тестів за допомогою класу *testListView.*
6. Клас *testListView* також використовуєтьсядля вибору тесту опитуваним, після чого користувач потрапить у вікно проходження тесту, дію якого контролює *UserViewTest.*
7. Завершаючи етапом роботи тесту – є клас *Result,* що поверне результати проходження тестування.

Додаткові класи:  
8. *HttpAsyncTask –* клас, що дає можливість зв’язку між користувачем та сервером для обміну даних.

1. *CustomListViewAdapter –* клас-адаптер, що дає змогу змінити стандартній адаптер виведення списку тестів на екран.

Back-End

Для збереження та обробки тестів використовується база даних за допомогою Веб-технологій, а саме мови програмування PHP та бази даних MySQL. Для пришвидшення роботи та коректності обробки тестів, було запропоновано використання framework-у Yii. База складається з 3-ох таблиць: Адміністратор, Тест, Запитання.

1. Таблиця Адміністратор (User) має рядки: E-mail та Password за допомогою яких відбувається авторизація користувача.
2. Таблиця Тестів (Test) має рядок: Name котре слугує для задання назві тесту.
3. Таблиця Запитань (Question) має рядки: Question – певне запитання тесту, чотири answ1..4 – варіанти відповідей, answtrue – варіант коректної відповіді та зв’язана з таблицею Test через унікальний ідентифікатор uid\_test.
4. Клас Users слугує для обробки вхідних даних та повернення підтвердження авторизації адміністратора.
5. Клас Test слугує для обробки вхідних даних щодо тесту та збереження його в БД(базу даних)
6. Клас Question слугує для обробки вхідних даних щодо запитань та варіантів відповідей, їх збереження до БД(бази даних) та повернення результату проходження тесту.