Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота

з курсу

«Інтеграційні програмні системи »

Виконали: студенти 4курсу
ФІОТ гр. ІО-31
Бригада “single” у складі:
Нікітін М.Д.

Київ 2017

1. Опис проекту

- Посилання на репозиторій проекту: https://github.com/Niki-Max-911/lab-3
- Посилання на список задач спланованих до виконання проекту: https://www.pivotaltracker.com/n/projects/1902549

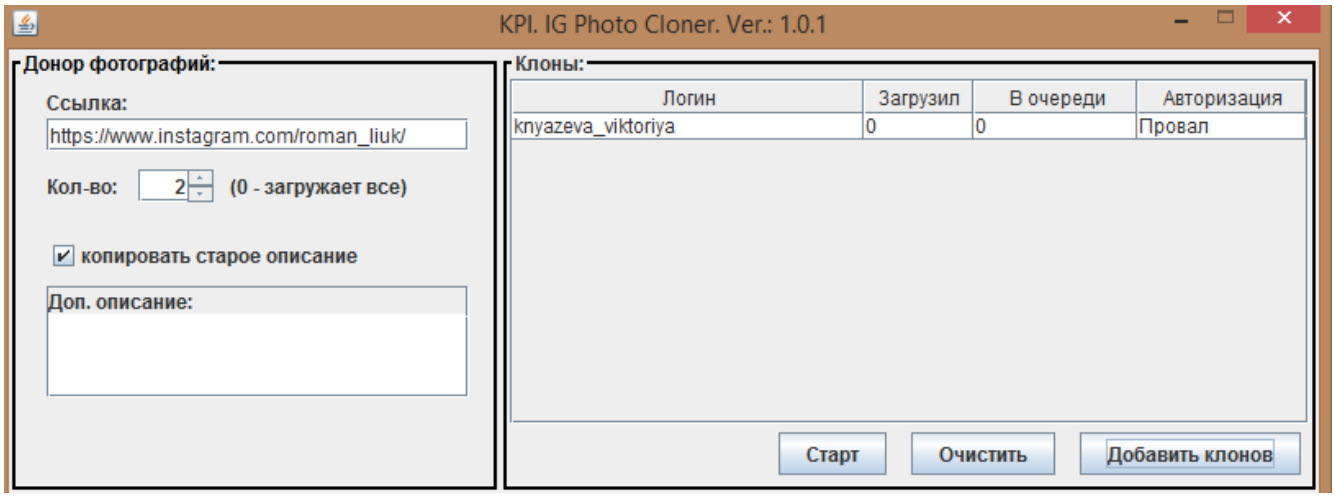
Розроблений проект признається для допомоги інтернет маркетологам, що працюють в соціальній мережі **Instagram**.

Instagram – це соц.мережа, де основна ціль: викладати фотографії власника аккунта, тоді як інші користувачі можуть їх переглядати та оцінювати.

Трапляються ситуації коли необхідно створити кілька аккаунтів для просування товарів/послуг та «пройтись» по цільовій аудиторій в різних географічних регіонах. Кожна з сторінок рекламує один і той же товар чи послугу але стратегії просування аккаунтів та роботи з адиторією різні(залежать від географії ЦА). Виникає необхідність завантажувати одні й ті ж Фотографії на ці аккаунти, надалі їх називатиму клони.

Отож, програма виконує клонування фотографій з вказаного одного аккаунта на інші – вказані у файлі(логін та пароль для авторизації). Файл вибирається при завантаженні програми.

Скріншот вікна програми:



Програма виконана на мові Java SE з використанням бібліотеки SWING яка слугує для створення деск-топ додатків. Інтеграція з сервісом **Instagram** відбувається через їх API для мобільного додатку.

Для з'єднання з сервера використовувалась бібліотека ***org.apache.httpcomponents.***

Для тестування використані наступні бібліотеки: ***junit***, ***mockito***(мок-тестування).

Цілком сплановані задачі (2 лаб.) виконані абсолютно усі для досягнення мети. Програма є робочою, протестованою та готова довжитку.

2. Maven - система атоматичної збірки

Maven - це засіб автоматизації роботи з програмними проектами, який спочатку використовувався для Java проектів. Використовується для управління (management) та складання (build) програм. Створений Джейсоном ван Зилом (*Jason van Zyl*) у 2002 році. За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML. XML-файл описує проект, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни. Сервер із додатковими модулями та додатковими бібліотеками розміщується на серверах. Раніше Maven, де він був частиною *Jakarta Project*.

Для опису програмного проекту який потрібно побудувати (*build*), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач - таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).

Ключовою особливістю Maven є його мережева готовність (network-ready).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних Java-проектів з відкритим кодом, від Apache та інших організацій та окремих розробників. Цей репозиторій та його реорганізований наступник, - Maven 2 репозиторій, - намагається бути де-факто механізмом для дистрибуції Java програм, але прийняття його в такій якості йде повільно. Результат виконання maven compile для збірки проекту:

```
"C:\Program Files\Java\jdk1.8.0_66\bin\java" "-Dmaven.home=C:\Program Files (x86)\JetBrains\IntelliJ IDEA 14.1.4\plugi
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building InstagramCloner 1.3
[INFO] ------------------------------------------------------------------------
[WARNING] The artifact org.apache.commons:commons-io:jar:1.3.2 has been relocated to commons-io:commons-io:jar:1.3.2
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ InstagramCloner ---
[debug] execute contextualize
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory F:\KPI\7semestr\integr_systems\lab-3\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.3:compile (default-compile) @ InstagramCloner ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.268s
[INFO] Finished at: Thu Jan 12 01:36:38 EET 2017
[INFO] Final Memory: 7M/77M
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

3. Сервер безперервної інтеграції. Travis-ci

https://travis-ci.org/Niki-Max-911/lab-3/builds

Як виявилося, термін «continuous integration» досить старий. Він був введений Мартіном Фаулером (Martin Fowler) у 2000-му році і викладений у статті «Continuous Integration» або «безперервна інтеграція». Це частина процесу розробки, в якій проект збирається / тестується в різних середовищах автоматично і безперервно. Задумувалася дана методика для найбільш швидкого виявлення помилок / протиріч інтеграції проекту, а отже зниження витрат на наступні простої.

Принцип досить простий: на окремій машині працює служба, в обов'язки якої входить отримання вихідного коду проекту, його збірка, тестування, логування.

Для реалізації безперервної інтеграції виділити окремий сервер і підтримувати його в робочому стані, забезпечити наявність необхідних програмних комплексів, налаштувати середовища виконання, робити резервні копії даних і т.д. Цілком логічно делегувати цю відповідальність на сторонні сервіси. Наприклад **travis-ci** - «хостинг безперервної інтеграції для open source співтовариства».

Travis-ci підтримує безліч мов програмування. Щоб почати користуватися сервісом необхідно зареєструватись як розробник від Git-Hub, вказати репозиторій проекту для безперервної інтеграції. В корені проекту необхідно створити файл/скрипт ***.travis.yml***. Приклад такого легко знайти в документації сервісу ***.travis.yml.***

Скрипт виконуватиметься завжди при оновленні гіт-репозиторія (коли відбувається пуш нових комітів). В скрипті проекту вказано команду *Maven(*`mvn package`*)* яка виконує збірку та тестування проекту(тестування – проміжна стадія). Команда *Maven(*`mvn pmd:pmd`*)* виконує статичний аналіз коду *maven* плагіном; результат можна глянути на сторінці віртуальної консолі.

4. Експоненціальна витримка

Для вирішення задачі раптового зникнення з'єднання з вказаним хостом соц.-мережі інстаграм було вирішено задачу експоненціальної витримки. Для цього я задіяв надбудову до бібліотеки Apache HttpClient, та використав готовий клас заготовку. Конфігурація стратегії що забезпечує експоненціальну витримку представлена нижче:

```
public class LocalhostConnector extends Connector {

    public LocalhostConnector() {
        CacheConfig cc = CacheConfig.DEFAULT;
        ExponentialBackOffSchedulingStrategy expopencialBackoffStartegy = new ExponentialBackOffSchedulingStrategy(cc);

        config = DEF_REQUEST_CONFIG;
        cookieStore = new BasicCookieStore();
//      browser = HttpClients.custom().setDefaultRequestConfig(config)
//              .setDefaultCookieStore(cookieStore).setUserAgent(USER_AGENT)
//              .setRetryHandler(MY_RETRY_HANDLER)
//              .setDefaultSocketConfig(SOCKET_CONFIG).build();

        browser = CachingHttpClients.custom()
                .setSchedulingStrategy(expopencialBackoffStartegy)
                .setDefaultRequestConfig(config)
                .setDefaultCookieStore(cookieStore)
                .setUserAgent(USER_AGENT)
//              .setRetryHandler(MY_RETRY_HANDLER)
                .setDefaultSocketConfig(SOCKET_CONFIG).build();
        context = HttpClientContext.create();
    }
}
```

Отриманий графік можна переглянути нижче: