**НАЦІОНАЛЬНЫЙ ТЕХНІЧНИЙ УНИВЕРСИТЕТ УКРАЇНИ «Київський політехнічний інститут»** Факультет інформатики та обчислювальної техніки Кафедра автоматизованої обробки інформації та управління

**КУРСОВА РОБОТА**

по дисципліні

Об’єктно-орієнтоване програмування

(назва дисципліни)

**Варіант № 21**

Виконав: студент групи ІП-82

Троцюк П.С.

(Прізвище та ініціали)

Прийняла: доцент Іванова Л.М.

Київ – 2020

ЗМІСТ

[ЗАВДАННЯ 3](#_Toc42114981)

[ОПИС ЗАСТОСУВАННЯ 4](#_Toc42114982)

[АРХІТЕКТУРА ЗАСТОСУВАННЯ 4](#_Toc42114983)

[ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ 7](#_Toc42114984)

[ДІАГРАМА КЛАСІВ ТА ТАБЛИЦІ 8](#_Toc42114985)

[ДІАГРАМА КЛАСІВ 8](#_Toc42114986)

[ТАБЛИЦІ ОПИСУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 11](#_Toc42114987)

[РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ ЗАСТОСУВАННЯ 19](#_Toc42114988)

[ТЕХНІЧНА ХАРАКТЕРИСТИКА 19](#_Toc42114989)

[РЕКОМЕНДАЦІЇ ПО ВСТАНОВЛЕННЮ ТА НАЛАШТУВАННЮ 21](#_Toc42114990)

[РЕКОМЕНДАЦІЇ КОРИСТУВАЧЕВІ 24](#_Toc42114991)

[ВИХІДНИЙ КОД ЗАСТОСУВАННЯ 26](#_Toc42114992)

[СПИСОК ЛІТЕРАТУРИ 27](#_Toc42114993)

ЗАВДАННЯ

Написати консольне застосування за архітектурним шаблоном MVC з розподіленням відповідальності між компонентами, яке виконує обробку набору даних згідно з варіантом завдання (додаток А) та забезпечує наступне:

* Збереження даних у файлі (формат файлу буль-який);
* Читання даних у пам'ять при запуску застосунку;
* Збереження даних у той же файл при завершені роботи застосунку, якщо дані були змінені;
* Збереження проміжних даних у будь-який файл (користувач вводить ім’я файлу);
* Інтеракивність з користувачем (мова інтерфейсу має обиратися при запуску застосунку на виконання);
* Логування подій та помилок в роботі застосунку.

Варіант 21

Факультатив:

* Назва, Викладач, Дата початку та закінчення, Список студентів, Список оцінок;
* Конструктор, Методи доступу, Метод toString().

Функціонал:

1. Отримати список факультативів, що викладаються вказаним викладачем.
2. Отримати середній бал студентів для вказаного факультативу.

ОПИС ЗАСТОСУВАННЯ

АРХІТЕКТУРА ЗАСТОСУВАННЯ

Розробка даного програмного забезпечення грунтувалась на архітектурному шаблоні MVC (model-view-controller). На рисунку нижче ви можете бачити як цей шаблон працює:

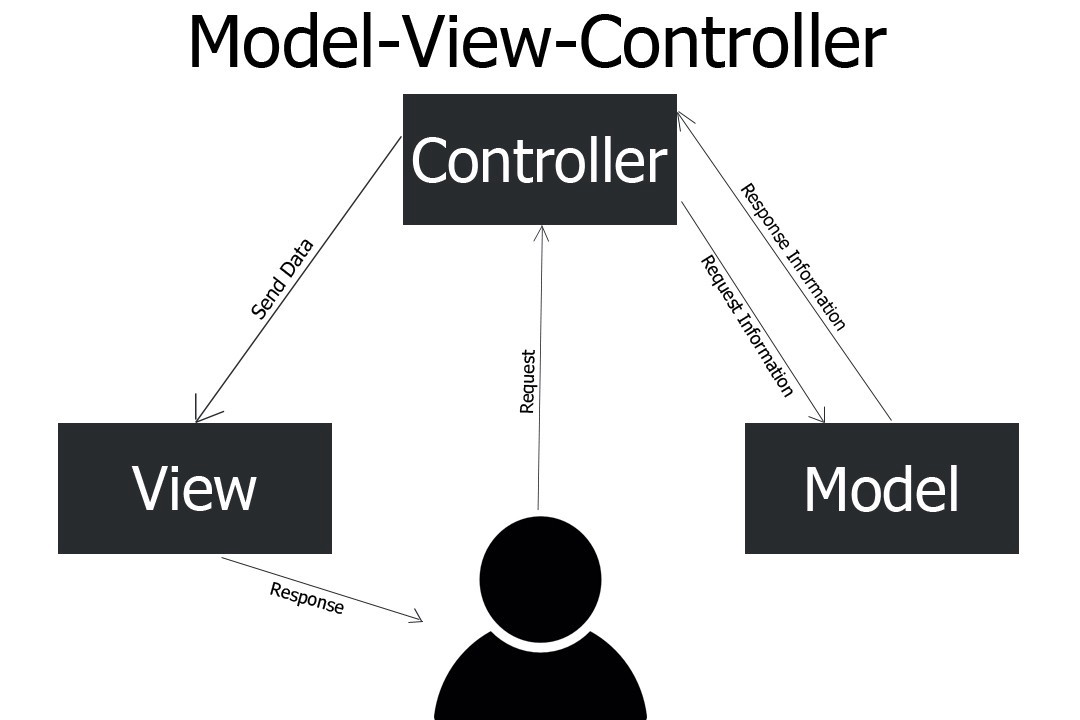


Рисунок 1 – Модель MVC

Як можемо бачити з рисунка, дані відправлені користувачем надходять до контроллера, далі йде запит до Моделі, вона оброблює цей запит та відсилає результат назад контроллеру. Потім контроллер відправляє ці дані на відображення користувачу в View. І вже View повинна знати як відобразити ці дані нашому користувачу. Все просто і ясно. Отож подивимось як я реалізував даний шаблон на діаграмі компонентів, яка показує чітку взаємодію фізичних компонентів програмного забезпечення.

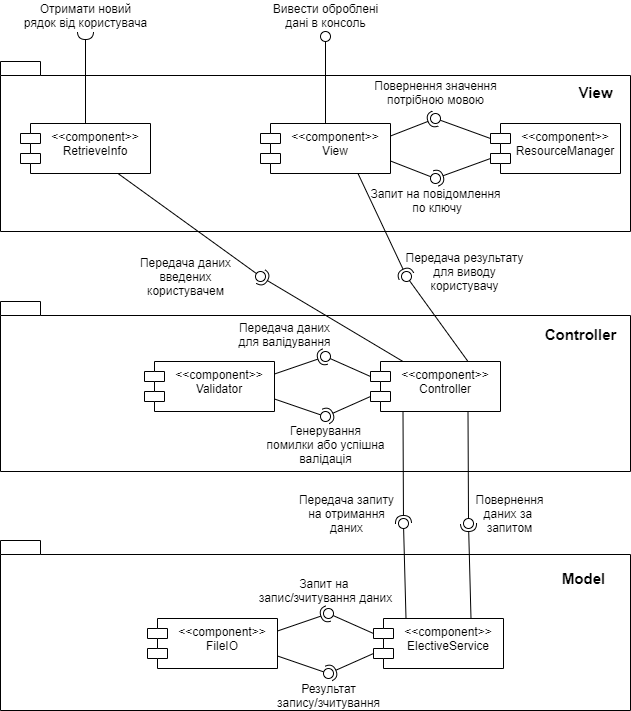


Рисунок 2 – Діаграма компонентів

Отож на рисунку ми чітко можемо бачити, що у нас є 3 головні пакети: View, Controller, Model. Все відповідає стандарту MVC. Як ми можемо бачити, я розмістив компонент, який зчитує інформацію користувача в пакеті View. На мою думку, тут немає нічого поганого, що View приймає запит, адже вся обробка та валідація йде в контроллері. Тепер за повну комунікацію з користувачем відповідає пакет View та його компоненти RetrieveInfo та View. Компонент RetrieveInfo містить в собі тільки одну функцію, яка зчитує дані введені користувачем, та відразу ж передає їх контроллеру. Компонент View приймає дані від контроллера та в красивому вигляді подає їх користувачу. Але перед тим як вивести їх користувачу, він в деяких випадках відправляє їх компоненту ResourceManager, який в свою чергу дивиться на локаль користувача та відповідно повертає значення потрібною мовою. І тільки тоді компонент View виводить інформацію нашому користувачу.

2 пакет, а саме Controller, відповідає за те, як працює наша програма, він керує пакетами View та Model. В ньому містяться 2 компоненти: Validator та Controller. Controller це саме ядро нашої програми. Він отримує дані від компонента RetrieveInfo та відправляє їх на валідацію до компонента Validator. Компонент Validator в свою чергу, перевіряє ці дані і якщо вони не пройшли валідацію, то генерується помилка, яка направляється саме компоненту Controller. Далі Controller при згенерованій помилці відправляє дані пакету View компоненту View, щоб вивести помилку. Якщо ж помилка не була створена, то Controller відправляє запит пакету Model компоненту ElectiveService. І коли запит був оброблений і відправлений контроллеру, той в свою чергу відправляє дані пакету View, який успішно виводить повідомлення.

Отож перейдемо до останнього нашого пакету – Model. Він відповідає за нашу бізнез-логіку. Головним його компонентом є ElectiveService, який приймає запити від контроллера та оброблює їх. В залежності від ситуації він звертається до компонента FileIO, який в свою чергу зчитує/записує дані з/до файлу. Після вдалої обробки запиту, ElectiveService відправляє результат запиту назад до контроллера.

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

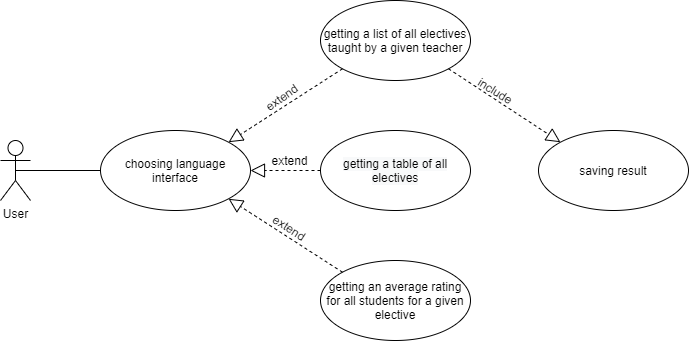


Рисунок 3 – Діаграма прецедентів

На даній діаграмі ми можемо бачити що ця програма не є досить великою, точніше вона є малою. В нас всього один актор і це звичайний користувач. Як ми можемо бачити з цієї діаграми з самого початку користувач повинен вибрати якою мовою йому буде зручніше за все працювати з програмою. Після цього йому пропонується всього три дії, які ви можете бачити на рисунку вище. І тільки в одному випадку, коли користувач вибрав опцію “Отримати список всіх факультативів, який веде заданий вчитель”, йому пропонується зберегти ці дані в файл.

ДІАГРАМА КЛАСІВ ТА ТАБЛИЦІ

ДІАГРАМА КЛАСІВ

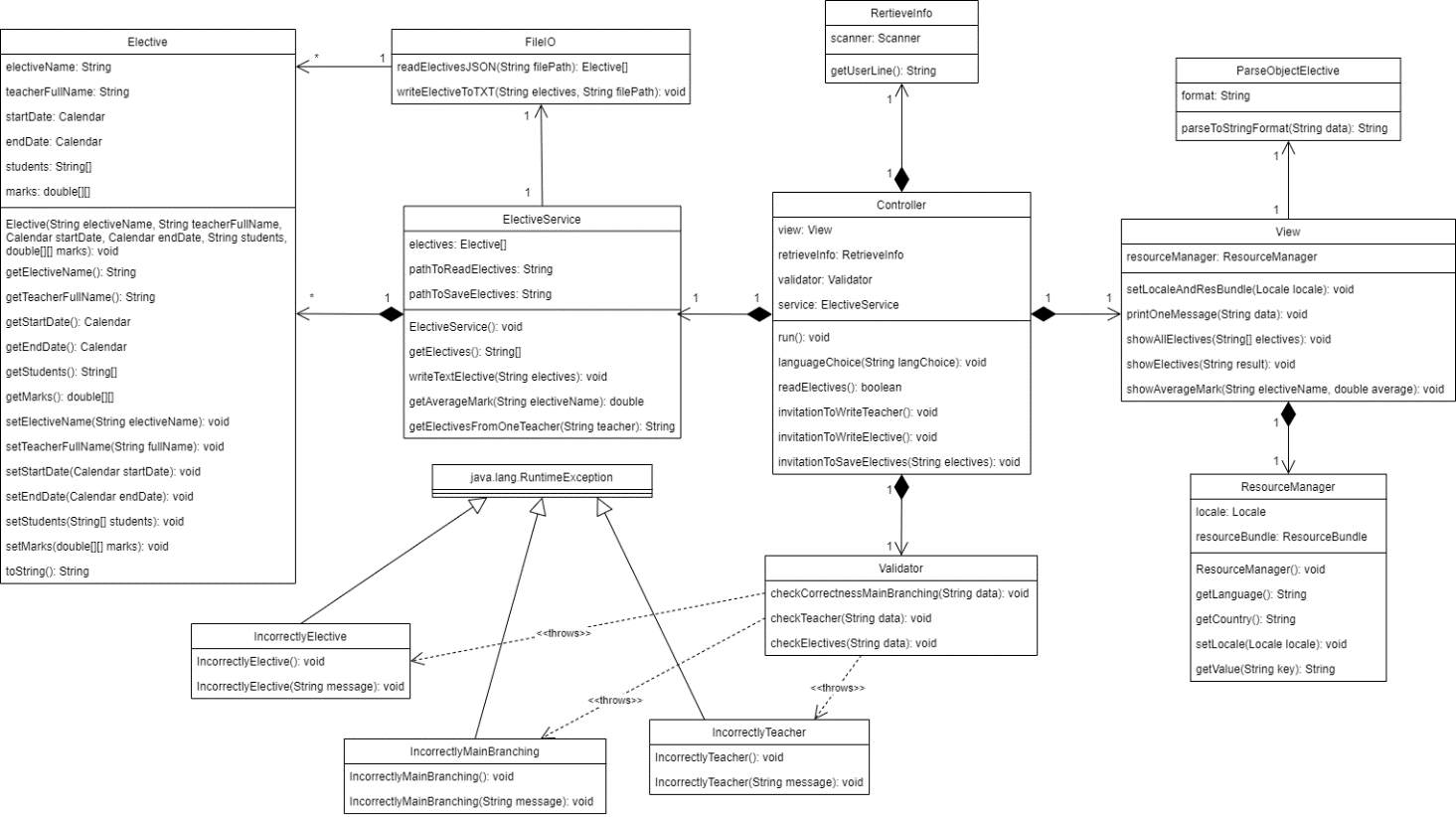


Рисунок 4 – Діаграма класів

Залишу ще посилання на цю діаграму:  
<https://drive.google.com/file/d/1JUIzDkGB6aozHjo6OZX4hmN0JPxOAtcH/view?usp=sharing>

На даній діаграмі можемо бачити як наші програмні класи залежать друг від друга. По середині стоїть головний клас Controller, який композує в собі 4 класи: Elective Service, Validator, View, RetrieveInfo. Цей клас, контролює взагалі весь процес виконання програми, збирає в собі інші не менш важливі для роботи класи . Чому на діаграмі використана саме композиція? Тому що, клас контроллер створює єдині екземпляри цих класів, які без класа контроллера “н живуть”. Далі подивимось на клас Validator. Він міститься єдиним екземпляром в класі Controller та виконує функцію перевірки введених даних на різних стадіях виконанні програми. Також ми можемо бачити, що він “прокидує” помилки, написані мною, при неправильних даних, введених користувачем. Ці помилки наслідуються від класа java.lang.RuntimeException. В них є по 2 конструктори: перший – конструктор за замовчуванням, а другий, коли ми передаємо туди повідомлення про помилку.

Клас RetrieveInfo теж міститься єдиним екземпляром в класі Controller та виконує функцію зчитування даних введених користувачем. Клас View як і попередні, теж в нашому центральному класі та виконує функцію спілкування з користувачем, тобто красиво виводить дані на екран, а також змінює мову інтерфейсу, в залежності від вибору користувача. З допомогою цього класу, користувач взагалі бачить інформацію, яку видає йому програма (в нашому випадку це все в консолі). Він в свою чергу містить в собі єдиний екземпляр класа ResourceManager, який також не може існувати без класа View, тому це композиція. ResourceManager, в свою чергу, відповідає за визначення локалі користувача та спочатку програми встановлює її в вибрану локаль користувачем. Тобто цей клас відповідає за мову на якій буде цей інтерфейс. Також клас View використовує поле та метод класа ParseObjectElective, тому між ними є асоціація. Клас ParseObjectElective виконує єдину функцію – красивий вивід всієї інформації про факультативи в табличному вигляді. З тим як інформація виводиться на екран розібрались, тому перейдемо в розділ де ці всі дані отримуються та оброблюються.

Клас ElectiveService відповідає за надання інформації по нашим факультативам і міститься в єдиному екземплярі в класі Controller. Він містить методи, в яких реалізовані ті 2 завдання, які були описані в розділі “Завдання”. Також він контролює всі наші факультативи. В його конструкторі, ми зчитуємо дані про факультативи з допомогою методу класа FileIO та заповнюємо наш масив факультативів, з яким вподальшому оперуємо. Зв’язок між цим класом та між класом Elective є композицією, адже факультативи створюються тільки в ньому і тільки один раз при виклику його конструктора. Зв’язок між цим класом та FileIO є асоціацією, адже ElectiveService використовує лише функції даного класу. В свою чергу клас FileIO відповідає за зчитування та запис даних, тобто за input/output. Він містить дві функції, перша для зчитування факультативів з формату .json, друга для запису результату в .txt. FileIO використовує клас Elective, для зручного переведення об’єктів .json в об‘єкти джава, тому між ними існує асоціація. Ну і останній класі в цій діаграмі – Elective. Це наш клас факультативу, в якому містяться всі потрібні поля, гетери та сетери, а також перевизначений метод toString.

ТАБЛИЦІ ОПИСУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Таблиця 1 – Специфікація методів основних класів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 1 | Controller | run | Основна функція, яка запускається в класі Main для взаємодії з користувачем | -не має- | -не має- |
| 2 | Controller | language Choice | Функція, яка викликає функцію з класа View та передає вибір користувача, щоб встановити нову локаль для користувача | Вибір користувача, рядок | -не має- |
| 3 | Controller | read Electives | Функція, яка створює новий об’єкт ElectiveService, який в свою чергу зчитує всі факультативи з файлу та розміщає їх в оперативній пам’яті | -не має- | Булева змінна, яка повідомляє про те, чи успішно ініціалізувався Elective Service |
| 4 | Controller | invitation ToWrite Teacher | Допоміжна функція, для функції run, яка видає користувачеві запрошення для вводу вчителя | -не має- | -не має- |
| 5 | Controller | invitation ToWrite Elective | Допоміжна функція, для функції run, яка видає користувачеві запрошення для вводу факультативу | -не має- | -не має- |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 6 | Controller | invitation ToSave Electives | Допоміжна функція, для функції invitationToWrite Teacher, яка видає користувачеві запрошення для збереження результату пошуку | Результат пошуку (факультативи), рядок | -не має- |
| 7 | View | setLocale andRes Bundle | Функція, яка викликає функцію з ResourceManager для встановлення нової локалі для користувача | Локаль передана з контроллера (вибір користувача) | -не має- |
| 8 | View | printOne Message | Функція, яка виводить повідомлення в консоль | Повідомлення, рядок | -не має- |
| 9 | View | showAll Electives | Функція, яка в табличному вигляді виводить всі факультативи | Факультативи, масив рядків | -не має- |
| 10 | View | show Electives | Функція, яка перевіряє чи результат пошуку факультативів, і в залежності від цього виводить їх або виводить повідомлення, що їх немає | Результат пошуку (факультативи), рядок | -не має- |
| 11 | View | show Average Mark | Функція, яка перевіряє чи оцінка != -1, і в залежності від цього виводить оцінку або повідомлення про відсутність такого факультативу | Назва факультативу та середня оцінка, рядок та дійне число > 0 | -не має- |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 12 | Retrieve Info | getUser Line | Зчитування введеного користувачем рядка | -не має- | Введена користувачем інформація, рядок |
| 13 | Parse Object Elective | parseTo String Format | Допоміжна функція для функції showAllElectives в класі View, яка допомагає красиво та в табличному вигляді вивести факультативи | Дані (інформація про факультативи), які перечислені через кому, рядок | Відформатовані дані, String.format |
| 14 | Resource Manager | Resource Manager | Конструктор, в якому відбувається ініціалізація локалі за замовчанням та resourceBundle за замовчуванням | -не має- | -не має- |
| 15 | Resource Manager | get Language | Повертає мову користувача (викликається функція getLanguage в об’єкта Locale) | -не має- | Мова по заданій локалі, рядок |
| 16 | Resource Manager | getCountry | Повертає країну користувача (викликається функція getCountry в об’єкта Locale) | -не має- | Країна по заданій локалі, рядок |
| 17 | Resource Manager | setLocale | Встановлює нову локаль для користувача | Локаль | -не має- |
| 18 | Resource Manager | getValue | Повертає значення по ключу відповідно до локалі користувача | Ключ, рядок | Значення по ключу, рядок |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 19 | Validator | check CorrectnessMain Branching | Перевірка на те, чи правильно користувач вибрав опцію в головному меню, якщо ні, то кидається помилка | Дані введені користувачем, рядок | -не має- |
| 20 | Validator | check Teacher | Перевірка на те, чи правильно користувач ввів дані в запрошені для вводу вчителя, якщо ні, то кидається помилка | Дані введені користувачем, рядок | -не має- |
| 21 | Validator | check Elective | Перевірка на те, чи правильно користувач ввів дані в запрошені для вводу факультативу, якщо ні, то кидається помилка | Дані введені користувачем, рядок | -не має- |
| 22 | Elective Service | Elective Service | Конструктор, в якому з файла зчитуються факультативи та записуються в оперативну пам’ять | -не має- | -не має- |
| 23 | Elective Service | get Electives | Функція, за допомогою якої, ми передаємо дані про факультативи | -не має- | Масив факультативів переведених в рядки, масив рядків |
| 24 | Elective Service | writeText Electives | Викликає функцію для запису даних в файл та передає туди наші результати пошуку факультативів | Наші результати(факультативи), рядок | -не має- |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 25 | Elective Service | getAverageMark | Функція, яка знаходить середню оцінку всіх учнів, які вчаться по заданому факультативу | Назва факультативу, рядок | Середня оцінка, дійсне число |
| 26 | Elective Service | get Electives FromOne Teacher | Функція, яка знаходить факультативи, які веде заданий вчитель | Повне ім’я вчителя, рядок | Назви факультативів, рядок |
| 27 | FileIO | read Electives JSON | Функція, яка зчитує дані про факультативи з файлу розширенням .json | Шлях до файлу, рядок | Зчитані факультативи, масив об’єктів Elective |
| 28 | FileIO | write ElectivesToTXT | Функція, яка записує дані про факультативи в файл з роширенням .txt | Дані (в нашому випадку факультативи) та шлях до файлу, два рядки | -не має- |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 29 | Elective | Elective | Конструктор, який ініціалізує всі поля | Назва факультативу, повне ім’я вчителя, дата початку, дата кінця, список студентів (масив рядків), журнал оцінок (подвійний масив дійсних чисел) | -не має- |
| 30 | Elective | getElective Name | Повертає значення поля electiveName | -не має- | Назва факультативу, рядок |
| 31 | Elective | getTeacherFullName | Повертає значення поля teacherFullName | -не має- | Повне ім’я вчителя, рядок |
| 32 | Elective | getStart Date | Повертає значення поля startDate | -не має- | Дата початку факультативу, Calendar |
| 33 | Elective | getEnd Date | Повертає значення поля endDate | -не має- | Дата кінця факультативу, Calendar |
| 34 | Elective | getStudents | Повертає посилання на масив студентів (поле students) | -не має- | Студенти факультативу, масив рядків |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 35 | Elective | getMarks | Повертає посилання на подвійний масив оцінок (поле marks) | -не має- | Оцінки всіх студентів факультативу, подвійний масив дійсних чисел |
| 36 | Elective | setElective Name | Встановлює нове значення полю electiveName | Назва факультативу, рядок | -не має- |
| 37 | Elective | setTeacherFullName | Встановлює нове значення полю teacherFullName | Повне ім’я вчителя, рядок | -не має- |
| 38 | Elective | setStart Date | Встановлює нове значення полю startDate | Дата початку факультативу, Calendar | -не має- |
| 39 | Elective | setEnd Date | Встановлює нове значення полю endDate | Дата кінця факультативу, Calendar | -не має- |
| 40 | Elective | setStudents | Встановлює посилання на новий масив студентів (поле students) | Студенти факультативу, масив рядків | -не має- |
| 41 | Elective | setMarks | Встановлює посилання на новий подвійний масив оцінок (поле marks) | Оцінки всіх студентів факультативу, подвійний масив дійсних чисел | -не має- |

Продовження таблиці 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Назва класу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів |
| 42 | Elective | toString | Змога привести об’єкт Elective до звичайного рядка | -не має- | -не має- |
| 43 | Incorrectly Elective | Incorrectly Elective | Конструктор за замовчанням | -не має- | -не має- |
| 44 | Incorrectly Elective | Incorrectly Elective | Конструктор в який ми передаємо повідомлення, щоб описати помилку | Повідомлення помилки, рядок | -не має- |
| 45 | Incorrectly Main Branching | Incorrectly Main Branching | Конструктор за замовчанням | -не має- | -не має- |
| 46 | Incorrectly Main Branching | Incorrectly Main Branching | Конструктор в який ми передаємо повідомлення, щоб описати помилку | Повідомлення помилки, рядок | -не має- |
| 47 | Incorrectly Teacher | Incorrectly Teacher | Конструктор за замовчанням | -не має- | -не має- |
| 48 | Incorrectly Teacher | Incorrectly Teacher | Конструктор в який ми передаємо повідомлення, щоб описати помилку | Повідомлення помилки, рядок | -не має- |

РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ ЗАСТОСУВАННЯ

ТЕХНІЧНА ХАРАКТЕРИСТИКА

Windows

* Windows 10 (8u51 або більш пізня)
* Windows 8.x (настільна версія)
* Windows 7 з пакетом оновлення 1 (SP1)
* Windows Vista SP2
* Windows Server 2008 R2 з пакетом оновлення 1 (SP1) (64-розрядна версія)
* Windows Server 2012 и 2012 R2 (64-розрядна версія)
* RAM: 128 МБ
* Простір на диску: 124 МБ для JRE; 2 МБ для оновлення Java
* Процесор: мінімальна вимога - Pentium 2 266 МГц
* Браузери: Internet Explorer 9 і вище, Firefox

Mac OS X

* Mac на базі процесора Intel під управлінням Mac OS X 10.8.3+, 10.9+
* Повноваження адміністратора для установки
* 64-бітний браузер. Для запуска Oracle Java для Mac вимагається 64-розрядний браузер (наприклад, Safari).

Linux

* Oracle Linux 5.5+
* Oracle Linux 6.x (32-розрядна версія), 6.x (64-розрядна версія)
* Oracle Linux 7.x (64-розрядна версія) (8u20 або більш пізня)
* Red Hat Enterprise Linux 5.5+1, 6.x (32-розрядна версія), 6.x (64-розрядна версія)
* Red Hat Enterprise Linux 7.x (64-розрядна версія) (8u20 або більш пізня)
* Suse Linux Enterprise Server 10 SP2+, 11.x
* Suse Linux Enterprise Server 12.x (64-розрядна версія) (8u31 або більш пізня)
* Ubuntu Linux 12.04 LTS, 13.x
* Ubuntu Linux 14.x (8u25 або більш пізня)
* Ubuntu Linux 15.04 (8u45 або більш пізня)
* Ubuntu Linux 15.10 (8u65 або більш пізня)
* Браузери: Firefox

РЕКОМЕНДАЦІЇ ПО ВСТАНОВЛЕННЮ ТА НАЛАШТУВАННЮ

Перед встановленням програмного забезпечення, потрібно переконатися, що на комп'ютері встановлений JDK (Java Development Kits) 8+ версії. Для цього в консолі вашої ОС пропишіть наступну команду, яка знаходиться всередині лапок: “java -version”. Якщо ж у вас немає даного програмного забезпечення, тоді вам потрібно його встановити. Можете перейти за наступним посиланням:  
<https://www.oracle.com/java/technologies/javase-jdk14-downloads.html>

І так, з самого початку перед тим як встановлювати мою програму, створіть папку в яку ви зможете помістити дане програмне забезпечення. Після того як ви створили папку, переходьте по посиланню нижче:  
<https://github.com/PauLOk777/object-oriented-programming-labs/tree/master/jarForExecute>. Тут ви зможете побачити 3 файли: перший це сама програма (coursework.jar), другий це ресурс в якому містяться всі дані про факультативи і без якого неможливий початок роботи програми (data.json) і останній це файл в який будуть зберігатись ваші проміжкові результати (electives.txt).

Перед тим як перейти до встановлення програмного забезпечення, все ж рекомендую вам створити ці два останні файли і заповнити перший (data.json). Ось як це має виглядати:

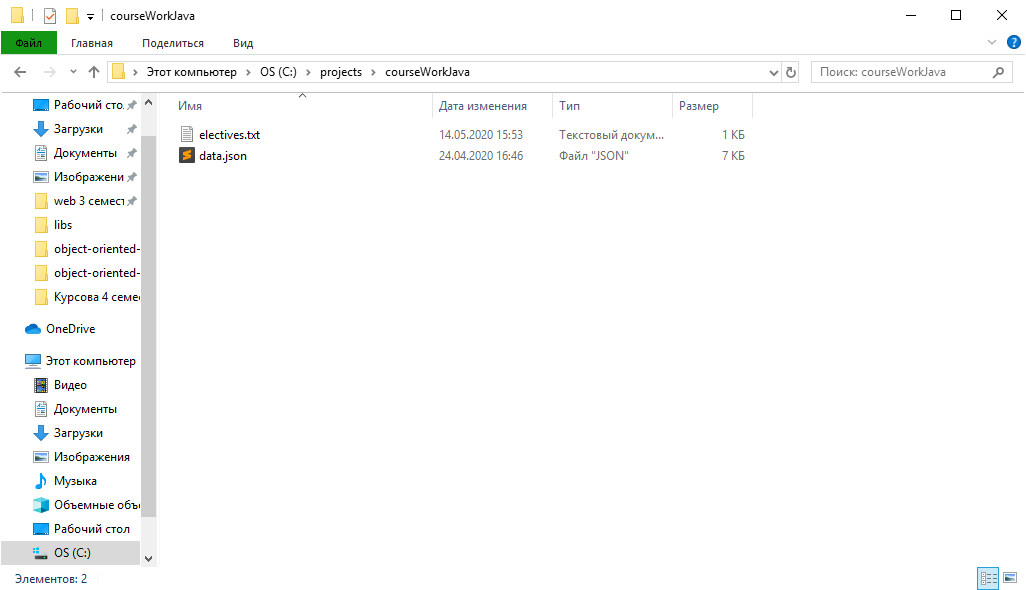


Рисунок 5 – Приклад папки до встановлення пз

Після того як ви створили папку та наповнили її цими файлами (обов’язково з цими іменами і розширеннями), можна перейти до встановлення програмного забезпечення. Ще раз переходимо по посиланню вище та нажимаємо на courseWork.jar:

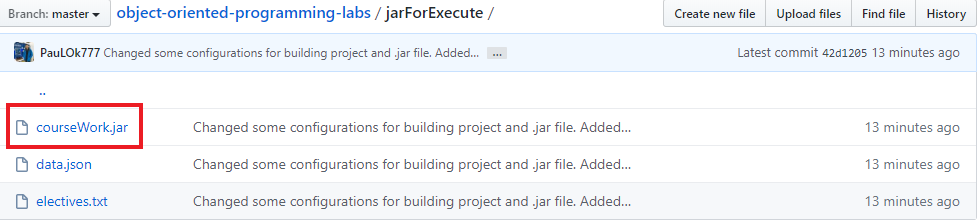


Рисунок 6 – Вигляд папки Github з якої потрібно встановити пз

Далі в вас появиться нове вікно в якому потрібно знайти кнопку “Download” та натиснути на неї:

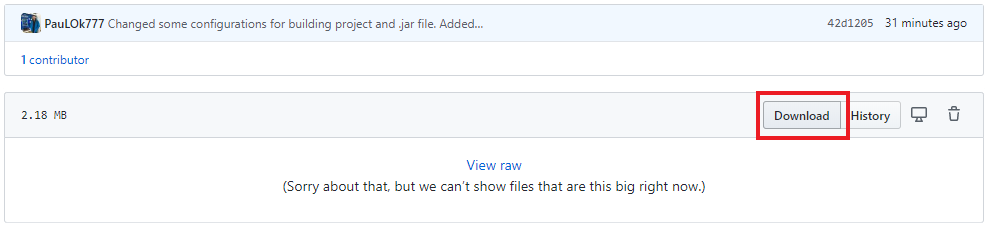


Рисунок 7 – Кнопка “Download”

Далі вам буде запропоновано зберегти цей файл в директорії на ваш вибір. Вам потрібно вибрати ту папку, яку ви створювали для цього програмного забезпечення і в якій вже лежать 2 файли. Після того як ви встановили це все, можна приступати до налаштування. Впринципі саме робота програми вже повністю налаштована для вас. Ви можете тільки змінювати дані в data.json:

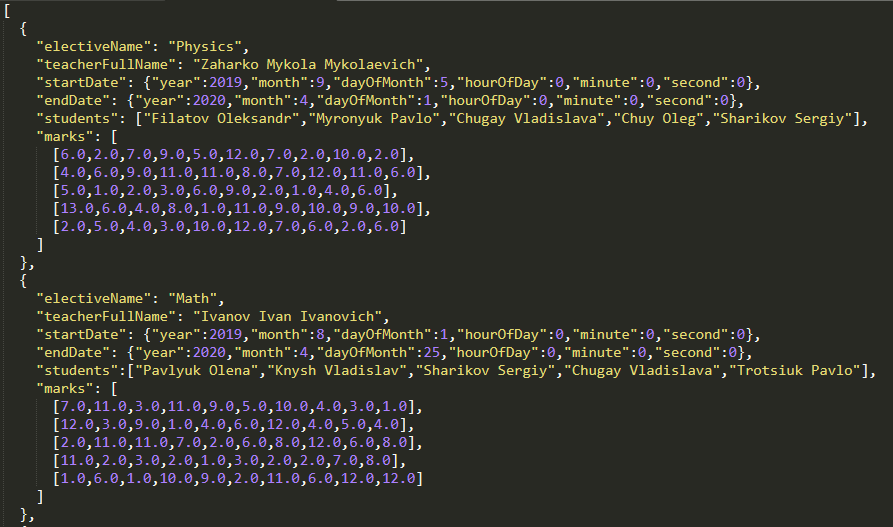


Рисунок 8 – Вигляд файла JSON

Як ви можете бачити з рисунка в файлі data.json міститься великий масив всередині якого знаходяться об’єкти. Якщо ви захочете добавити якийсь новий факультатив, у вас повинні бути присутні всі 6 полів. Перед тим як змінювати щось, ознайомтесь з JSON.

РЕКОМЕНДАЦІЇ КОРИСТУВАЧЕВІ

І нарешті ми дібрались до питання: “А як нам запустити програму?”. Після цього як ви все встановили і, можливо, щось доналаштували, можна приступати саме до запуску програми. Для початку вам необхідно відкрити консоль вашої операційної системи. Потім вручну дібратись до вашої папки за допомогою консолі, в якій має бути моє пз. І потім виконати наступну команду: java -jar <name>.jar. Замість <name> вам необхідно написати courseWork, тобто запит буде виглядати наступним чином: java -jar courseWork.jar:

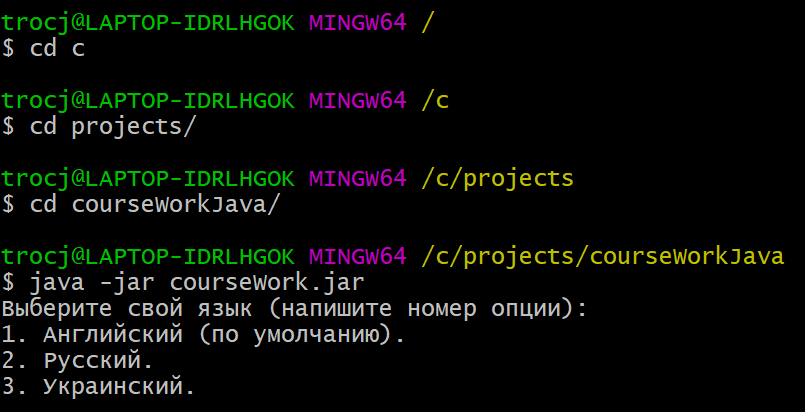


Рисунок 9 – Запуск програми через консоль

Тепер ви можете спокійно працювати з даною програмою. Також може виникнути питання: “Звідки беруться дані і куди зберігаються в процесі роботи програми?”. В розділі “Рекомендації по встановленню та налаштуванню” я розповідав про 2 файли: data.json і electives.txt. З першого файли дістаються дані для роботи, а в другий файл записуються проміжкові дані.

Також дана програма створює файл app.log, в який буде записуватись вся інформація про програму і про те як ви її використовуєте. Якщо щось піде не так, ви завжди можете скинути цей файл мені і я вже буду аналізувати що ж там пішло не так.

Всього може виникнути 5 помилок передбачуваних програмою: FileNotFound, FileException, IncorrectMainBranching, IncorrectTeacher, IncorrectElective. Не лякайтесь, якщо вам це нічого не говорить. Нижче буде описано як з ними боротися та причини їх виникнення:

* FileNotFound. Виникає при старті програми, коли не вдається знайти файл data.json. Переконайтесь, що ви правильно виконали всі пункти описані в розділі “ Рекомендації по встановленню та налаштуванню ”. Також може виникати, коли ви намагаєтесь зберегти проміжні результати. Ситуація аналогічна попередній і вирішується аналогічно так же.
* FileException. Виникає при зчитуванні/запису з/в файл(у). Сталась або системна помилка або порушена файлова структура. Попробуйте перезапустити програму і створити заново файл.
* IncorrectMainBranching. Виникає через вашу неуважність (які і всі нижче описані). Виникає коли ви невірно ввели номер опції. Ще раз уважно перечитайте повідомлення, яке вам виводиться і виберіть потрібну вам опцію за допомогою цифри або напишіть “quit” для завершення програми.
* IncorrectTeacher. Також виникає через те, що ви неправильно ввели повне ім’я вчителя. В повідомленні вказується приклад, в якому можна побачити, що повне ім’я складається з трьох слів і з англійських букв. І також не забувайте поважати вчителя – кожне слово з великої літери.
* IncorrectElective. Все аналогічно вище описаній помилці. Тільки тут річ йде за назву факультативу. Не забувайте, що кожен факультатив пишеться з великої літери, в ньому наявні лише англійські літери, пробіли та “-”.

ВИХІДНИЙ КОД ЗАСТОСУВАННЯ

Посилання на репозиторій GitHub:

<https://github.com/PauLOk777/object-oriented-programming-labs>

СПИСОК ЛІТЕРАТУРИ

1. Стаття про архітектурний шаблон MVC. Режим доступу:  
   <https://uk.wikipedia.org/wiki/Модель-вид-контролер>
2. “Java 8. Руководство для начинающих” Герберт Шилдт.
3. “Java методы программирования” И.Н. Блинов, В.С. Романчик.
4. Офіційна документація Oracle. Режим доступу:  
   <https://docs.oracle.com/javase/8/docs/api/index.html>
5. Документація по бібліотеці Gson. Режим доступу:

<https://sites.google.com/site/gson/gson-user-guide>

1. Документація по бібліотеці Log4j2. Режим доступу: <https://logging.apache.org/log4j/2.x/>