Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра “Системи автоматизованого проектування”

**ЗВІТ**

про проходження переддипломної практики

студента ІV курсу групи КНст-32

Зарічного Олега

**База практики** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва та адреса підприємства)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Термін практики** з «16» квітня 2018 р. по з «05» травня 2018 р.

**Керівники практики**

від кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доцент каф САП Фармага І.В.

(підпис) (наук.ст, вч.звання, прізвище, ім’я, по батькові)

від бази практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (посада, прізвище, ім’я, по батькові)

**ОЦІНКА** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата** \_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**Список скорочень**

БД – база даних

ПЗ – програмне забезпечення

ПП – програмний продукт

СУБД – система управління базою даних (система керування базою даних).

API - Application Programming Interface (Прикладний програмний інтерфейс)

REST – Representational State Transfer (передача репрезентативного стану)

Зміст

[1. Вступ 5](#_Toc513466405)

[2. Опис бази практики 9](#_Toc513466406)

[3. Тема бакалаврської кваліфікаційної роботи 10](#_Toc513466407)

[4. Методологія розробки ПП 10](#_Toc513466408)

[6. Огляд використаних технологій 12](#_Toc513466409)

[7. Огляд архітектури ПП 1](#_Toc513466410)5

[8. Опис структур даних 1](#_Toc513466411)6

[9. Висновок 1](#_Toc513466412)7

[10. Список використаної літератури](#_Toc513466413) 18

Додаток А…………………………………………………………………………………………………….19

1. Вступ

Перед дипломна практика за темою бакалаврської кваліфікаційної роботи студентів є невід’ємною складовою освітньо-професійної програми підготовки студентів вищих навчальних закладів України. Вона становить важливу та обов’язкову ланку в системі наскрізної практичної підготовки висококваліфікованих фахівців до майбутньої практичної діяльності. Переддипломна практика за темою бакалаврської кваліфікаційної роботи спрямована на закріплення теоретичних знань, отриманих студентами за час навчання, вдосконалення практичних навичок і умінь в роботі за обраною спеціальністю, аналіз та виконання завдань для оформлення та захисту бакалаврської кваліфікаційної роботи.

Основними завданнями переддипломної практики є:

* провести дослідження на тему власного диплому «Розробка програми для менеджменту завдань»;
* спроектувати реалізацію автоматизованої програми для менеджменту завдань ;
* ознайомлення з нормативно-технологічною документацією підприємства – бази практики, вивчення особливостей адміністративної та інформаційної взаємодії між його окремими підрозділами;
* ознайомлення з конфігурацією та архітектурою технічних засобів всередині організації;
* ознайомлення з програмним забезпеченням і програмними продуктами, які використовуються або створюються в процесі діяльності організації де проходила переддипломна практика;
* ознайомлення з технологічними процесами керування інформаційними потоками, організацією зберігання даних, створення баз даних і доступу до даних;
* вивчення послідовності, методів і засобів оброблення даних;
* опанування технологіями розроблення, впровадження та супроводу програмного забезпечення;
* вивчення основ конфігурування технічних засобів, інформаційних систем та мереж, систем автоматизованого проектування та керування;
* оволодіння сучасними технологіями оброблення інформації та застосування їх для виконання практичних завдань;
* набуття виробничих навичок, а саме: технічної, проектувальної, виконавської;
* набуття навичок групової роботи та роботи в колективі спеціалістів;
* підготовка до вивчення спеціальних дисциплін на старших курсах.

Під час переддиплоиної практики було розроблено макет програми для менеджменту робочого процесу компанії, організації, підприємства, тощо. За основу бралися певні проблеми в організації цього процесу та втрати даних про членів Молодіжної громадської організації «Рада студентів технічних університетів».

На сьогоднішній день існує багато таких систем, з великим функціоналом, але в основному такі сервіси є платними (по піписці на кожного користувача), або ж вони не наповнені достатнім функціоналом. Тому було обрано створити власний програмний продукт для таких цілей.

Все більше різноманітних компаній та організацій почитають працювати по аналогічних системах. Хочу замітити, що це відбувається не тільки в ІТ-компаніях.

Звичайно такі системи вже є реалізовані, проте шлях в якому ми їх просто купуємо доволі кошторисний. Тому більшість людей в таких випадках використовують два варіанти:

Перший варіант заснований на тому, щоб знайти сервіси по наданню послуг - JIRA, Harvest, Trello і тому подібне. Але такі сервіси є доволі дорогими (10$ на місяць з людини). Тому для багатьох компаній такий варіант відпадає одразу.

Іншим варіантом є створення власного програмного забезпечення. Такий варіант і було обрано компанією, в якій я проходив практику. Крім витрачання меншої кількості грошей, однією з головних переваг є можливість модифікування цього програмного забезпечення власноруч. Якщо необхідно додати якийсь функціонал, характерний лише для вашого підприємства, а платні сервіси такого функціоналу не мають, створення власного програмного забезпечення це єдиний вихід.

1. Опис бази практики

Базою практики було обрано та одобрено Молодіжної громадської організації «Рада студентів технічних університетів»(далі BEST).

[**BEST**](http://best.eu.org/) (Board of European Students of Technology) або Рада студентів технічних університетів – неприбуткова, неполітична та нерелігійна організація, яка представлена у 94 осередках 32 країн світу. З 1989 року BEST забезпечує обмін знаннями та співробітництво між студентами, компаніями та університетами на теренах Європи.

**Візія: Сила в різноманітті**

Люди розуміють та поважають різні культури та суспільства. Середовище, де сила в різноманітності, підтримує людей у реалізації їх потенціалу та відповідальній діяльності.

**Місія: Розвиток студентів**

BEST допомагає студентам досягти міжнародності мислення, кращого розуміння культур та народів, а також розвинути навички роботи в міжнародному середовищі.

**BEST Spirit**

Це те, завдяки чому ми залишаємося змотивованими, те, що підштовхує нас постійно рухатися далі. BEST Spirit – це натхнення у всьому, що ми робимо. Ось основні **5 цінностей**, які визначають BEST Spirit: **Flexibility** (Гнучкість), **Friendship** (Дружба), **Fun** (Фан), **Improvement** (Розвиток), **Learning** (Навчання).

**Локальний осередок BEST Lviv** існує з 2002 року. Діяльність львівського осередку забезпечується її Виконавчою радою та декількома робочими групами. Окрім Ради, у осередку налічується приблизно 70 активних членів організації. Також варто додати, що BEST Lviv співпрацює з більше ніж 50 українськими та міжнародними компаніями та працює для 31 000 студентів Львівської Політехніки.

1. Тема бакалаврської кваліфікаційної роботи

Розробка застосунку для менеджменту завдань

1. Методологія розробки ПП

Як методологію розробки було обрано **Scrum**. Scrum (чит.як скрам) — підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ-лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос-функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990-х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом[en] та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM.

Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту. Схему роботи цієї методології ви можете побачити на Рис 1.

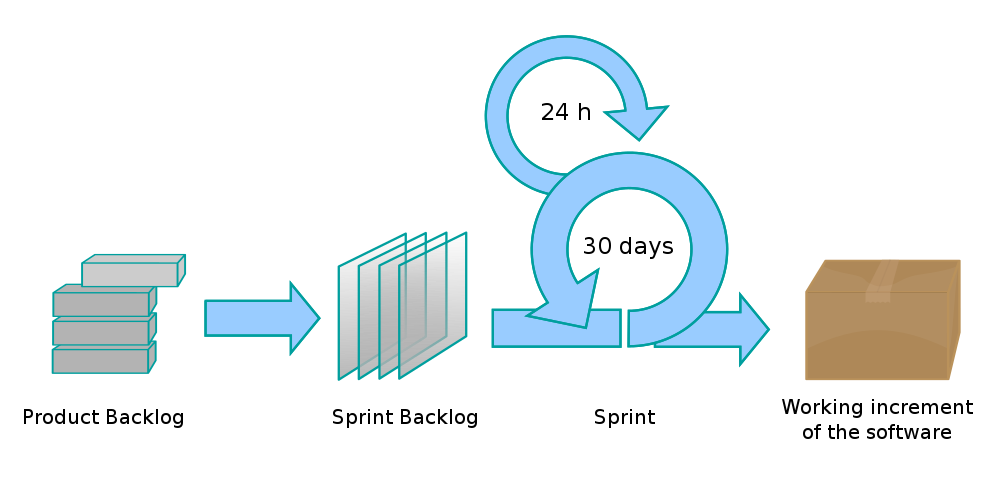


Рис 1.Скрам процес

1. Огляд використаних технологій

Основною технологією для даного проекту є **Spring Framework**, який став свого роду стандартом для веб-застосунків, реалізованих з використанням мови програмування Java. Spring – великий проект, який на даний час складається з багатьох модулів. З них можна виділити декілька основних, які найбільше використовувалися впродовж виконання даної роботи:

* Spring Boot
* Spring Data JPA
* Spring Web MVC
* Spring Security

Дамо невелику характеристику кожного з них.

**Spring Boot**

Spring Boot дозволяє легко створювати програмні продукти на основі Spring, які можна просто запустити. Більшість застосунків створених з використанням Spring Boot потребують дуже малої конфігурацій, на відміну від того, що було до появи цієї технології. Це значно пришвидшує процес розробки та вивчення даного фреймворку.

Основні цілі проекту:

* надати радикально швидший та широко доступний інструмент для старту розробки зі Spring.
* мати змогу в виборі налаштувань «з коробки» та при необхідності швидко змінити все, коли вимоги починають відхилятися від умов «за замовчуванням».
* надати цілий ряд нефункціональних засобів, які є загальними для великих класів проектів (наприклад, вбудовані сервери, безпека, показники, перевірки здоров'я та зовнішня конфігурація);
* зовсім відмовитись від генерації коду і незручної конфігурації за допомогою XML.

**Spring Data**

Ціль Spring Data полягає у забезпеченні знайомої та послідовної Spring-подібної моделі програмування для доступу до даних, зберігаючи при цьому особливі риси основного сховища даних.

Це спрощує використання технологій доступу до даних, реляційних і нереляційних баз даних, хмарних служб даних тощо. Це великий проект, який містить багато підпроектів, специфічних для заданої бази даних, які розробляються завдяки співпраці з багатьма компаніями та розробниками, які підтримують ці технології.

Класи, які відповідають за операції обміну даними з сховищем називають **репозиторіями.**

Основні функції:

* швидкі репозиторії з індивідуальними абстракціями для об'єктно-реляційного відображення;
* динамічна генерація запитів з імен методів репозиторію;
* реалізація базових класів предметної області відповідно з наданими налаштуваннями;
* підтримка прозорого аудиту виконаних операцій;
* можливість інтеграції власного коду в реалізацію репозиторію;
* проста інтеграція зі Spring за допомогою JavaConfig та XML-конфігурації
* розширена інтеграція зі Spring MVC контролерами

**Spring Web MVC**

Spring Web MVC - це оригінальна веб-структура, побудована на Servlet API і включена в Spring Framework з самого початку. Формальне ім'я "Spring Web MVC" походить від назви свого модуля spring-webmvc, але воно більш широко відома як "Spring MVC".

Spring Web model-view-controller (MVC) або модель-представлення-контролер, побудований навколо DispatcherServlet, який розподіляє запити по оброблювачам. Вона налаштовується на запити мови, локалі, тимчасові зони та багато іншого. Обробник за замовчуванням будується на основі анотацій @Controller і @RequestMapping, які надають широкий набір гнучких методів обробки запитів. Після версії 3.0 Spring. також дозволяє створювати RESTful веб-сервіси, використовувати анотацію @PathVariable та інші можливості.

Spring MVC побудовано навколо центрального сервлету, який розподіляє запити між контролерами, а також надає інші широкі можливості при розробці веб-додатків. Насправді DispatcherServlet - повністю інтегрований сервлет у Spring IoC контейнер і таким чином отримує доступ до всіх можливостей Spring.

**Spring Security**

Spring Security надає комплексні засоби безпеки для застосунків на базі Java EE. Особливий акцент робиться на підтримку проектів, створених за допомогою Spring Framework, що є провідним рішенням Java EE для розробки програмного забезпечення.

Spring Security популярний з багатьох причин, але в основному через значну перевагу з точки зору функціоналу в порівнянні з Java EE Servlet специфікацією або EJB специфікацією.

"Автентифікація" та "авторизація" (або "контроль доступу") – це два основних напрями, що відповідають цілям Spring Security. "Автентифікація" - це процес встановлення особи користувача, який намагається отримати доступ до ресурсів програми. "Авторизація" – процес прийняття рішення чи користувачу дозволено виконувати дії, які він намагається здійснити. На момент, коли потрібне рішення про авторизацію, ідентичність користувача повинна бути встановлена ​​за допомогою процесу автентифікації. Ці поняття є загальними і зовсім не специфічними для Spring Security.

Spring Security підтримує широкий спектр моделей автентифікації. Більшість із цих моделей надаються або сторонніми особами, або розробляються відповідними органами стандартизації.

1. Огляд архітектури ПП

Вибір архітектури є одним з найважливіших етапів життєвого циклу розробки програмного забезпечення. На даний момент вагоме місце серед архітектурних стилів посідає REST, який поєднує в собі простоту та ефективність.

**REST** — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований 2000 року Роєм Філдінгом, одним із творців протоколу HTTP. В основі REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP. Філдінг розробив REST паралельно з HTTP 1.1 базуючись на попередньому протоколі HTTP 1.0.

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON). Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабованість системи і дозволяє їй еволюціонувати з новими вимогами.

Схематичне зображення роботи ПЗ на основі REST зображено на рис. 2. Варто зауважити, що даний підхід дозволяє розробити серверну частину таким чином, що в якості клієнта ми можемо використовувати різного роду технології та ПЗ (мобільний застосунок, програма для настільного ПК, браузер тощо). Це спричинено

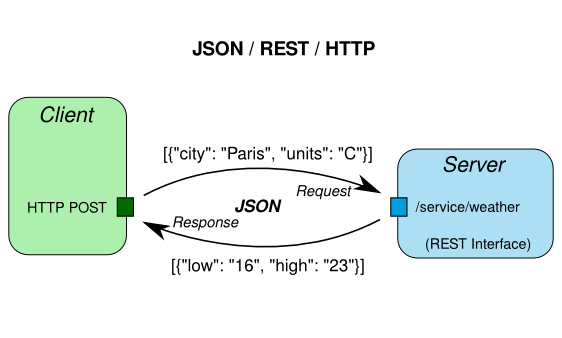


Рис. 2. Схематичне зображення роботи REST

1. Опис структур даних

Основним логічним ядром створюваного ПЗ є обробка даних, пов’язаних з розкладом занять, та пошуком вільних аудиторій згідно заданих параметрів. Тому потрібно добре продумати структуру бази даних.

Для зберігання було обрано PostgreSQL – об'єктно-реляційна система управління базами даних (СУБД). Є альтернативою як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

Загальновідомі проблеми об’єктно-реляційного відображення – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування.

Суть проблеми полягає в перетворенні таких об'єктів у форму, в якій вони можуть бути збережені у файлах або базах даних, і які легко можуть бути витягнуті в подальшому, зі збереженням властивостей об'єктів і відносин між ними. Ці об'єкти називають «постійними» (англ. persistent). Історично існує кілька підходів до рішення цієї задачі. Для даного ПП було обрано Java ORM технологію – Java Persistence API (JPA) – стандартизований інтерфейс для Java ORM фреймворків. Як реалізацію JPA було обрано фреймворк Hibernate, який на даний момент є одним з найпопулярніших ORM рішень в екосистемі Java.

1. Висновок

Під час проходження практики було проведено немало роботи над дослідженням способів реалізації теми проекту. Це дало змогу отримати навики в розробці архітектури ПЗ, плануванні робіт для досягнення максимального результату, вибору правильної методології на основі команди розробників.

Під час розробки програмного продукту я набув навиків реалізації застосунків з використанням RESTfull підходу. Це цінний досвід, оскільки цей архітектурний підхід зараз дуже поширений через свою простоту та ефективність.

Перед тим, як прийнятися за проектування та розробку додатку, було проведено аналіз вже існуючих програмних рішень в цій сфері, з метою визначити доцільність даної системи.

Переддипломна практика в МГО «Рада студентів технічних університетів» дало змогу отримати цінний досвід роботи в команді, дізнатися більше про життєвий цикл розробки програмного забезпечення. Також співпраця з обраною базою практики дозволила побачити реальний стан справ в ІТ галузі України.

1. Список використаної літератури
2. Spring Web MVC.

<https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>

1. Spring Boot Reference Guide.

<https://docs.spring.io/spring-boot/docs/2.0.1.RELEASE/reference/htmlsingle/>

1. Spring Security Reference.

<https://docs.spring.io/spring-security/site/docs/5.0.4.RELEASE/reference/htmlsingle/>

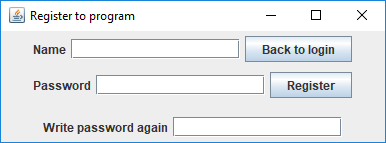
1. Spring in Action. Fourth edition – Craig Walls – Manning Publications 2015. – 626 с.

**Додаток А**

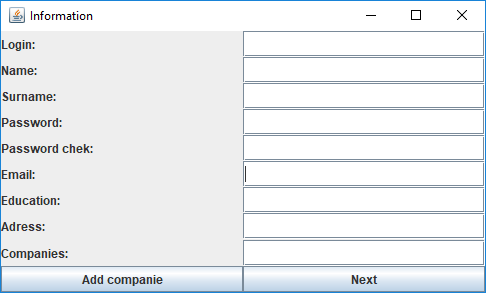
**Зображення додатку**



*Рис.1 Вікно авторизації*



*Рис.2 Вікно реєстрації*



*Рис.3 Вікно заповнення інформації*

**Код програми**

**package** Diplom.Diplom;

**import** java.awt.BorderLayout;

**import** java.awt.GridLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.WindowEvent;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

**import** javax.swing.JPanel;

**import** javax.swing.JPasswordField;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTabbedPane;

**import** javax.swing.JTextArea;

**import** javax.swing.JTextField;

**public** **class** Frame {

**void** login()

{

**final** JFrame loginframe = **new** JFrame("Login to program");

loginframe.setSize(400,120);

loginframe.setLocationRelativeTo(**null**);

loginframe.setDefaultCloseOperation(JFrame.***HIDE\_ON\_CLOSE***);

JButton loginbutton = **new** JButton("Login");

JButton regbutton = **new** JButton("Register");

**final** JTextField namefield = **new** JTextField(15);

JPasswordField passfield = **new** JPasswordField(15);

JLabel namelbl = **new** JLabel("Name");

JLabel passlbl = **new** JLabel("Password");

JPanel uptpanel = **new** JPanel();

JPanel downpanel = **new** JPanel();

uptpanel.add(namelbl, BorderLayout.***WEST***);

downpanel.add(passlbl, BorderLayout.***WEST***);

uptpanel.add(namefield, BorderLayout.***CENTER***);

downpanel.add(passfield, BorderLayout.***CENTER***);

uptpanel.add(loginbutton, BorderLayout.***EAST***);

downpanel.add(regbutton, BorderLayout.***EAST***);

loginframe.add(uptpanel, BorderLayout.***NORTH***);

loginframe.add(downpanel, BorderLayout.***CENTER***);

loginframe.setVisible(**true**);

regbutton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

register();

loginframe.dispatchEvent(**new** WindowEvent(loginframe, WindowEvent.***WINDOW\_CLOSING***));

}

});

loginbutton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

workspase(namefield.getText());

loginframe.dispatchEvent(**new** WindowEvent(loginframe, WindowEvent.***WINDOW\_CLOSING***));

}

});

}

**void** register()

{

**final** JFrame registerframe = **new** JFrame("Register to program");

registerframe.setSize(400,150);

registerframe.setLocationRelativeTo(**null**);

registerframe.setDefaultCloseOperation(JFrame.***HIDE\_ON\_CLOSE***);

JButton loginbutton = **new** JButton("Back to login");

JButton regbutton = **new** JButton("Register");

**final** JTextField namefield = **new** JTextField(15);

JPasswordField passfield = **new** JPasswordField(15);

JPasswordField repassfield = **new** JPasswordField(15);

JLabel namelbl = **new** JLabel("Name");

JLabel passlbl = **new** JLabel("Password");

JLabel repasslbl = **new** JLabel("Write password again");

JPanel uptpanel = **new** JPanel();

JPanel midpanel = **new** JPanel();

JPanel downpanel = **new** JPanel();

uptpanel.add(namelbl, BorderLayout.***WEST***);

midpanel.add(passlbl, BorderLayout.***WEST***);

downpanel.add(repasslbl, BorderLayout.***WEST***);

uptpanel.add(namefield, BorderLayout.***CENTER***);

midpanel.add(passfield, BorderLayout.***CENTER***);

downpanel.add(repassfield, BorderLayout.***CENTER***);

uptpanel.add(loginbutton, BorderLayout.***EAST***);

midpanel.add(regbutton, BorderLayout.***EAST***);

registerframe.add(uptpanel, BorderLayout.***NORTH***);

registerframe.add(midpanel, BorderLayout.***CENTER***);

registerframe.add(downpanel, BorderLayout.***SOUTH***);

registerframe.setVisible(**true**);

regbutton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

userinformation();

registerframe.dispatchEvent(**new** WindowEvent(registerframe, WindowEvent.***WINDOW\_CLOSING***));

}

});

loginbutton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

login();

registerframe.dispatchEvent(**new** WindowEvent(registerframe, WindowEvent.***WINDOW\_CLOSING***));

}

});

}

**void** workspase(String name) {

JFrame frame = **new** JFrame(name);

frame.setSize(800, 500);

frame.setVisible(**true**);

frame.setLocationRelativeTo(**null**);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setExtendedState(JFrame.***MAXIMIZED\_BOTH***);

JTabbedPane tabby = **new** JTabbedPane();

JPanel panel1= **new** JPanel();

JPanel panel2 = **new** JPanel();

//Creating Menu

JMenuBar menuBar = **new** JMenuBar();

JMenu fileMenu = **new** JMenu("lalala");

JMenuItem newFile = **new** JMenuItem("lalala");

JMenuItem openFile = **new** JMenuItem("lalala");

JMenuItem saveFile = **new** JMenuItem("lalala");

JMenuItem saveAsFile = **new** JMenuItem("lalala");

JMenuItem exitFile = **new** JMenuItem("lalala");

fileMenu.add(newFile);

fileMenu.add(openFile);

fileMenu.addSeparator();

fileMenu.add(saveFile);

fileMenu.add(saveAsFile);

fileMenu.addSeparator();

fileMenu.add(exitFile);

JMenu editMenu = **new** JMenu("lalala");

JMenuItem canselEdit = **new** JMenuItem("lalala");

JMenuItem cutEdit = **new** JMenuItem("lalala");

JMenuItem copyEdit = **new** JMenuItem("lalala");

JMenuItem pasteEdit = **new** JMenuItem("lalala");

JMenuItem deleteEdit = **new** JMenuItem("lalala");

JMenuItem findEdit = **new** JMenuItem("lalala");

JMenuItem replaceEdit = **new** JMenuItem("lalala");

editMenu.add(canselEdit);

editMenu.addSeparator();

editMenu.add(cutEdit);

editMenu.add(copyEdit);

editMenu.add(pasteEdit);

editMenu.add(deleteEdit);

editMenu.addSeparator();

editMenu.add(findEdit);

editMenu.add(replaceEdit);

JMenu analysMenu = **new** JMenu("lalala");

menuBar.add(fileMenu);

menuBar.add(editMenu);

menuBar.add(analysMenu);

tabby.addTab("First", panel1);

tabby.addTab("Second", panel2);

frame.setJMenuBar(menuBar);

frame.add(tabby);

}

**void** userinformation() {

JFrame frame = **new** JFrame("Information");

frame.setSize(500, 300);

frame.setVisible(**true**);

frame.setLocationRelativeTo(**null**);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

JLabel loginlbl = **new** JLabel("Login:");

JLabel namelbl = **new** JLabel("Name:");

JLabel surnamelbl = **new** JLabel("Surname:");

JLabel passlbl = **new** JLabel("Password:");

JLabel repasslbl = **new** JLabel("Password chek:");

JLabel maillbl = **new** JLabel("Email:");

JLabel educationlbl = **new** JLabel("Education:");

JLabel adresslbl = **new** JLabel("Adress:");

JLabel copmpanieslbl = **new** JLabel("Companies:");

JTextField loginfld = **new** JTextField(15);

JTextField namefld = **new** JTextField(15);

JTextField surnamefld = **new** JTextField(15);

JTextField passfld = **new** JTextField(15);

JTextField repassfld = **new** JTextField(15);

JTextField mailfld = **new** JTextField(15);

JTextField educationfld = **new** JTextField(15);

JTextField adressfld = **new** JTextField(15);

JTextField companiesfld = **new** JTextField(15);

JButton compbut = **new** JButton("Add companie");

JButton nextbut = **new** JButton("Next");

JPanel pan = **new** JPanel(**new** GridLayout(0, 2) );

JPanel dowpan = **new** JPanel(**new** GridLayout(2, 2) );

frame.add(pan, BorderLayout.***CENTER***);

frame.add(dowpan, BorderLayout.***SOUTH***);

pan.add(loginlbl);

pan.add(loginfld);

pan.add(namelbl);

pan.add(namefld);

pan.add(surnamelbl);

pan.add(surnamefld);

pan.add(passlbl);

pan.add(passfld);

pan.add(repasslbl);

pan.add(repassfld);

pan.add(maillbl);

pan.add(mailfld);

pan.add(educationlbl);

pan.add(educationfld);

pan.add(adresslbl);

pan.add(adressfld);

dowpan.add(copmpanieslbl);

dowpan.add(companiesfld);

dowpan.add(compbut);

dowpan.add(nextbut);

}

}