Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 5

з курсу: «Розробка програмного забезпечення мобільних пристроїв»

**Виконав:**  
студент 4-го курсу,  
групи ТВ-з11

Роговський Назар Тарасович

Посилання на GitHub репозиторій: <https://github.com/dragonfir2016/PW5_TB-311_Rohovskyi_Nazar_Tarasovych>

**Перевірив:**

Недашківський О.Л.

Київ 2024/2025

Практична робота № 5

**Завдання:**

Завдання 1

Створіть мобільний калькулятор для порівняння надійності одноколової та двоколової систем електропередачі та розрахунку збитків від перерв електропостачання у разі застосування однотрансформаторної ГТП у складі:

1. Порівняти надійність одноколової та двоколової систем електропередачі
2. Розрахувати збитки від перерв електропостачання у разі застосування однотрансформаторної ГПП

**Хід виконання:**

Для написання коду для розрахунку відповідних показників були використані дані з наступних таблиць:

A table with numbers and symbols

Description automatically generated

A table with numbers and letters

Description automatically generated

Частоти відмов одноклової системи:

A mathematical equation with numbers and symbols

Description automatically generated

Середньої тривалості відновлення:

A mathematical equation with numbers and symbols

Description automatically generated

Коефіційєнти аварійного і планового простоїв одноколової системи:

Частоти відмови двоколової системи з урахуванням вимикача:

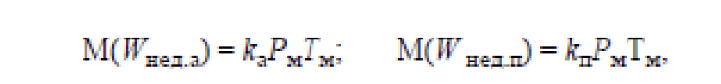
A black and white math equation

Description automatically generated with medium confidence

Коефіційєнти аварійного простою двоколової системи:



Математичного сподівання планового і аварійного недовідпущень електроенергії:



Завдання 1

Код реалізації першої частини завдання знаходиться у файлі Task1Fragment.kt у класі Task1Fragment. Спочатку ініціалізується таблиця зі значеннями всіх елементів доступних юзеру:

A computer screen shot of a code

Description automatically generated

Потім елементи вибрані користувачем зчитуються:

A computer screen shot of a program

Description automatically generated

Після цього якщо користувач натискає кнопку “Розрахувати Надійність”, запускаються всі розрахунки згідно з формулами і після цього показуютья результати користувачу:

calculateButton.setOnClickListener **{** try {  
 val calculations = selectedElements.*map* **{** element **->** elements[element] ?: throw Exception("Елемент не знайдено")  
 **}** val omegaOC = calculations.*sumOf* **{ it**.first **}** val tbOC = calculations.*sumOf* **{ it**.first \* **it**.second **}** / omegaOC  
 val kaOC = omegaOC \* tbOC / 8760.0  
 val kpMax = 43.0  
 val kpOC = 1.2 \* kpMax / 8760.0  
 val omegaDK = 2 \* omegaOC \* (kaOC + kpOC)  
 val omegaSC = omegaDK + 0.02  
  
 resultsTextView.*text* = """  
 Вибрані елементи: ${selectedElements.*joinToString*(", ")}  
   
 Частота відмов одноколової системи: ${df.format(omegaOC)} рік⁻¹  
 Середня тривалість відновлення: ${df.format(tbOC)} год  
 Коефіцієнт аварійного простою: ${df.format(kaOC)}  
 Коефіцієнт планового простою: ${df.format(kpOC)}  
   
 Частота відмов для двоколового варіанту: ${df.format(omegaDK)} рік⁻¹  
 Частота відмов двоколової системи (з секційним вимикачем): ${df.format(omegaSC)} рік⁻¹  
 """.*trimIndent*()  
 } catch (e: Exception) {  
 resultsTextView.*text* = "Помилка обчислення: ${e.message}"

Сам зовнішній вигляд налаштовується у файлі “fragment\_task1.xml”, який відповідає для UI для першої частини завдання і виглядає наступним чином:

A screenshot of a computer

Description automatically generated

При запуску контрольного прикладу та при довільних даних введених користувачем виходять наступні результати:

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Код реалізації другої частини завдання знаходиться у файлі Task2Fragment.kt у класі Task2Fragment. Task1Fragment. Спочатку ініціалізується таблиця зі значеннями всіх елементів доступних юзеру:

A screenshot of a computer program

Description automatically generated

Потім елемент вибраний користувачем зчитуються:

A screen shot of a computer program

Description automatically generated

Після цього якщо користувач натискає кнопку “Розрахувати Збитки”, запускаються всі розрахунки згідно з формулами і після цього показуютья результати користувачу:

calculateButton.setOnClickListener **{** try {  
 val selectedElement = spinner.*selectedItem*.toString()  
 val (omega, tb, tp) = elements[selectedElement]  
 ?: throw Exception("Елемент не знайдено")  
  
 val accidentCost = inputAccidentCost.*text*.toString().*toDouble*()  
 val plannedCost = inputPlannedCost.*text*.toString().*toDouble*()  
  
 val Pm = 5120.0  
 val Tm = 6451.0  
  
 val MwAccident = omega \* tb \* 0.001 \* Pm \* Tm  
  
 val MwPlanned = tp \* 0.001 \* Pm \* Tm  
  
 val totalLosses = (accidentCost \* MwAccident) + (plannedCost \* MwPlanned)  
  
 resultsTextView.*text* = """  
 Вибраний елемент: $selectedElement  
   
 Математичне сподівання аварійного недовідпущення: ${df.format(MwAccident)} кВт·год  
 Математичне сподівання планового недовідпущення: ${df.format(MwPlanned)} кВт·год  
   
 Загальні збитки: ${df.format(totalLosses)} грн  
 """.*trimIndent*()  
 } catch (e: Exception) {  
 resultsTextView.*text* = "Помилка: ${e.message}"  
 }

Сам зовнішній вигляд налаштовується у файлі “fragment\_task2.xml”, який відповідає для UI для другої частини завдання і виглядає наступним чином:

A screenshot of a computer

Description automatically generated

При запуску контрольного прикладу та при довільних даних введених користувачем виходять наступні результати:

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated

**Висновок**

В результаті виконання практичної роботи №5 було розроблено калькулятор надійності електрообладнання та економічних збитків від перерв у електропостачанні. Під час роботи було покращено навички програмування мовою Kotlin, закріплено досвід створення функціонального інтерфейсу користувача в Android Studio, а також освоєно динамічне додавання елементів інтерфейсу. Налаштовано коректну обробку введених даних та відображення результатів.