Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 4

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/PW4_TB-311_Rohovskyi_Nazar_Tarasovych>

**Перевірив:**

Недашківський О.Л.

Київ 2024/2025

Практична робота № 4

**Завдання:**

Завдання 1

Створіть мобільний калькулятор для розрахунку струму трифазного КЗ, струму однофазного КЗ, та перевірки на термічну та динамічну стійкість у складі:

1. Вибрати кабелі для живлення двотрансформаторної підстанції системи внутрішнього

електропостачання підприємства напругою 10 кВ

2. Визначити струми КЗ на шинах 10 кВ ГПП

3. Визначити струми КЗ для підстанції Хмельницьких північних електричних мереж (ХПнЕМ), яка може мати три режими: нормальний режим; мінімальний режим; аварійний режим.

**Хід виконання:**

Для написання коду для розрахунку відповідних показників були використані наступні формули:

Розрахункового струму для нормального і післяаварійного режимів:

A mathematical equation with black text

Description automatically generated with medium confidence

Економічного перерізу:

A math equation with black and white text

Description automatically generated with medium confidence

Опори елементів ЕПС:

A math equations and formulas

Description automatically generated

Початкового діючого значення струму трифазного КЗ:

A mathematical equation with a square root

Description automatically generated

Схеми підстанції 110/10 кВ:

A diagram of a number and a number

Description automatically generated with medium confidence

Та дані відхідної лінії 10 кВ

A table with numbers and letters

Description automatically generated

Завдання 1

Код реалізації першої частини завдання знаходиться у файлі MainActivity.kt у класі FragmentTask1. Саме тут числа введені користувачем зчитуються:

A screen shot of a computer program

Description automatically generated

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

val calculateButton = view.findViewById<Button>(R.id.*calculateButton*)  
val resultsTextView = view.findViewById<TextView>(R.id.*resultsTextView*)  
  
calculateButton.setOnClickListener **{** try {  
 val Sm = inputFields[0].*text*.toString().*toDouble*()  
 val Unom = inputFields[1].*text*.toString().*toDouble*()  
 val T = inputFields[2].*text*.toString().*toDouble*()  
 val Ik = inputFields[3].*text*.toString().*toDouble*()  
 val Tf = inputFields[4].*text*.toString().*toDouble*()  
  
 val Im = Sm / (Math.sqrt(3.0) \* Unom)  
 val thermalResistance = Ik \* Ik \* Tf  
  
 resultsTextView.*text* = """  
 Трифазний струм: ${"%.2f".*format*(Im)} A  
 Тепловий опір: ${"%.2f".*format*(thermalResistance)} Дж  
 """.*trimIndent*()  
 } catch (e: Exception) {  
 resultsTextView.*text* = "Помилка: перевірте введені дані."  
 }

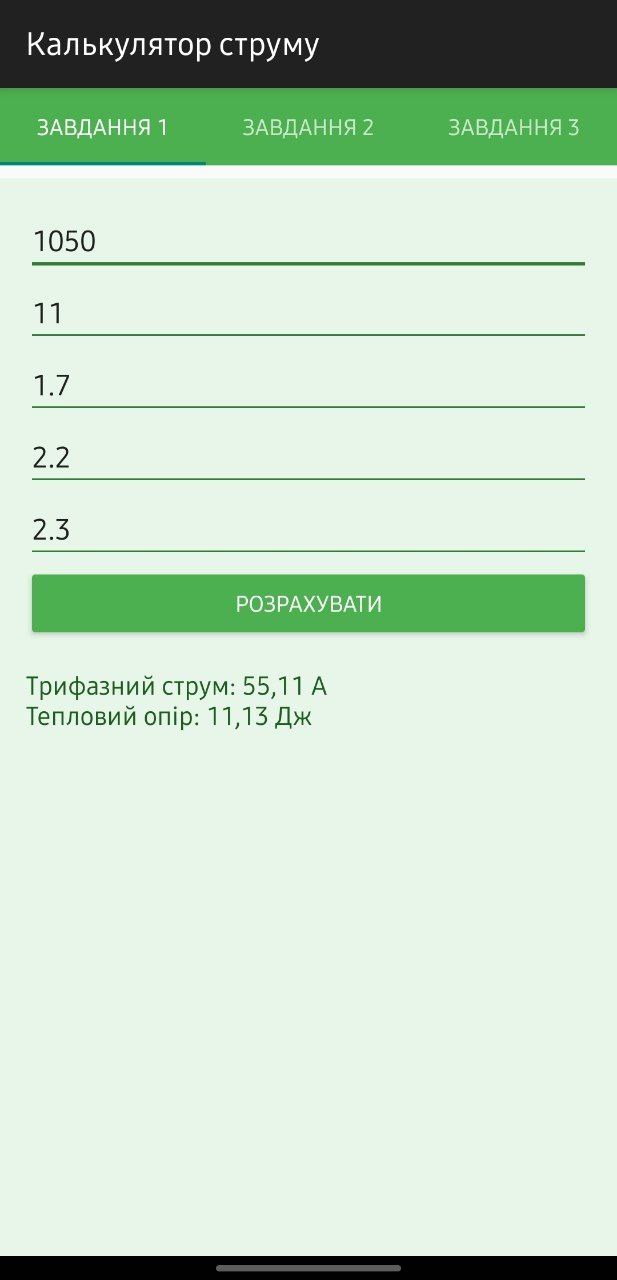
Сам зовнішній вигляд налаштовується у файлі “fragment\_task1.xml”, який відповідає для UI для першої частини завдання і виглядає наступним чином:

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 phone

Description automatically generated 

Код реалізації другої частини завдання знаходиться у файлі MainActivity.kt у класі FragmentTask2. Тут числа введені користувачем зчитуються:

A screen shot of a computer program

Description automatically generated

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

val calculateButton = view.findViewById<Button>(R.id.*calculateButton2*)  
val resultsTextView = view.findViewById<TextView>(R.id.*resultsTextView2*)  
  
calculateButton.setOnClickListener **{** try {  
 val Uc = inputFields[0].*text*.toString().*toDouble*()  
 val Sk = inputFields[1].*text*.toString().*toDouble*()  
 val Xt = inputFields[2].*text*.toString().*toDouble*()  
 val transformerImpedance = inputFields[3].*text*.toString().*toDouble*()  
  
 val Xtotal = Xt + transformerImpedance  
 val Ik = Uc / (Math.sqrt(3.0) \* Xtotal)  
  
 resultsTextView.*text* = """  
 Сумарний опір: ${"%.2f".*format*(Xtotal)} Ом  
 Трифазний струм КЗ: ${"%.2f".*format*(Ik)} A  
 """.*trimIndent*()  
 } catch (e: Exception) {  
 resultsTextView.*text* = "Помилка: перевірте введені дані."

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

Screens 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

Код реалізації третьої частини завдання знаходиться у файлі MainActivity.kt у класі FragmentTask3. Тут числа введені користувачем зчитуються:

A screen shot of a computer program

Description automatically generated

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

val calculateButton = view.findViewById<Button>(R.id.*calculateButton3*)  
val resultsTextView = view.findViewById<TextView>(R.id.*resultsTextView3*)  
  
calculateButton.setOnClickListener **{** try {  
 val RcNormal = inputFields[0].*text*.toString().*toDouble*()  
 val XtNormal = inputFields[1].*text*.toString().*toDouble*()  
 val RcMinimal = inputFields[2].*text*.toString().*toDouble*()  
 val XtMinimal = inputFields[3].*text*.toString().*toDouble*()  
 val Unom = inputFields[4].*text*.toString().*toDouble*()  
  
 val ZNormal = Math.sqrt(RcNormal \* RcNormal + XtNormal \* XtNormal)  
 val IkNormal = Unom / (Math.sqrt(3.0) \* ZNormal)  
  
 val ZMinimal = Math.sqrt(RcMinimal \* RcMinimal + XtMinimal \* XtMinimal)  
 val IkMinimal = Unom / (Math.sqrt(3.0) \* ZMinimal)  
  
 resultsTextView.*text* = """  
 Нормальний режим:  
 Сумарний опір: ${"%.2f".*format*(ZNormal)} Ом  
 Струм КЗ: ${"%.2f".*format*(IkNormal)} A  
  
 Мінімальний режим:  
 Сумарний опір: ${"%.2f".*format*(ZMinimal)} Ом  
 Струм КЗ: ${"%.2f".*format*(IkMinimal)} A  
 """.*trimIndent*()  
 } catch (e: Exception) {  
 resultsTextView.*text* = "Помилка: перевірте введені дані."

Сам зовнішній вигляд налаштовується у файлі “fragment\_task3.xml”, який відповідає для UI для третьої 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 computer

Description automatically generated

**Висновок**

В результаті виконання практичної роботи №4 було реалізовано мобільний калькулятор для розрахунку струмів короткого замикання та інших параметрів у трьох завданнях. Було покращено навички роботи з мовою Kotlin та реалізовано складні арифметичні обчислення за формулами. Закріплено процес створення функціонального інтерфейсу користувача в Android Studio, налаштовано обробку введених даних, їх коректну обробку та відображення результатів. Отримано досвід практичного використання мови Kotlin у сфері інженерних розрахунків.