Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Практична робота № 1

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/PW1_TB-311_Rohovskyi_Nazar_Tarasovych>

**Перевірив:**

Недашківський О.Л.

Київ 2024/2025

Лабораторна робота № 6

**Завдання:**

Завдання 1

Написати мобільний калькулятор для розрахунку складу сухої та горючої маси палива та нижчої теплоти згоряння для робочої, сухої та горючої маси за заданим складом компонентів палива, що задаються у вигляді значень окремих компонентів типу: H\_P, %; C\_P, %; S\_P, %; N\_P, %;O\_P, %; W\_P, %; A\_P, % (див. табл. 1.3.).

Завдання 2

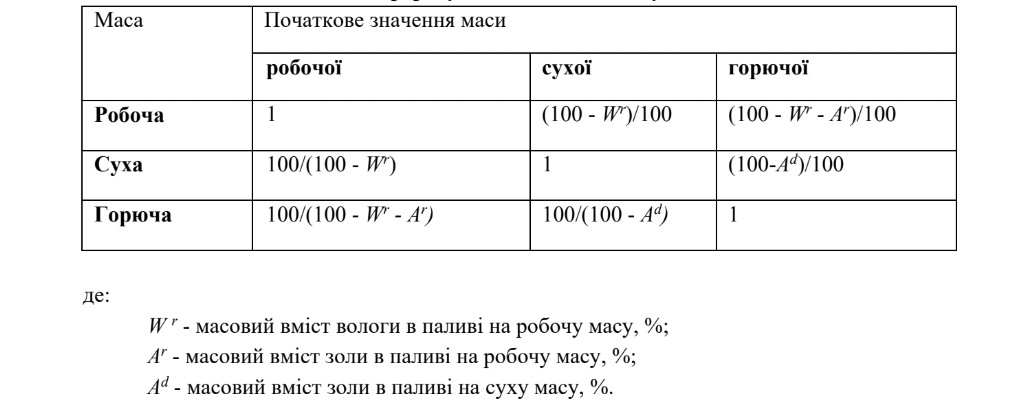
Написати мобільний калькулятор для перерахунку елементарного складу та нижчої теплоти згоряння мазуту на робочу масу для складу горючої маси мазуту, що задається наступними параметрами: вуглець, %; водень, %; кисень, %; сірка, %; нижча теплота згоряння горючої маси мазуту, МДж/кг; вологість робочої маси палива, %; зольність сухої маси, %; вміст

ванадію (V), мг/кг.

**Хід виконання:**

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

Для перерахунку масового вмісту складових палива:



Для нижчої теплоти згорання за формулою Мендєлєєва:



Для перерахунку теплоти згоряння:

A white sheet with black text and numbers

Description automatically generated

Завдання 1

Код реалізації завдання 1 знаходиться у файлі MainActivity.kt у класі Task1Fragment. 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:

calculateButton.setOnClickListener **{** try {  
 val H\_p = inputFields[0].*text*.toString().*toDouble*()  
 val C\_p = inputFields[1].*text*.toString().*toDouble*()  
 val S\_p = inputFields[2].*text*.toString().*toDouble*()  
 val N\_p = inputFields[3].*text*.toString().*toDouble*()  
 val O\_p = inputFields[4].*text*.toString().*toDouble*()  
 val W\_p = inputFields[5].*text*.toString().*toDouble*()  
 val A\_p = inputFields[6].*text*.toString().*toDouble*()  
  
 val K\_PC = 100 / (100 - W\_p)  
 val K\_PT = 100 / (100 - W\_p - A\_p)  
  
 val H\_C = H\_p \* K\_PC  
 val C\_C = C\_p \* K\_PC  
 val S\_C = S\_p \* K\_PC  
 val N\_C = N\_p \* K\_PC  
 val O\_C = O\_p \* K\_PC  
 val A\_C = A\_p \* K\_PC  
  
 val H\_T = H\_p \* K\_PT  
 val C\_T = C\_p \* K\_PT  
 val S\_T = S\_p \* K\_PT  
 val N\_T = N\_p \* K\_PT  
 val O\_T = O\_p \* K\_PT  
  
 val Q\_P\_H = 339 \* C\_p + 1030 \* H\_p - 108.8 \* (O\_p - S\_p) - 25 \* W\_p  
 val Q\_C\_H = (Q\_P\_H + 0.025 \* W\_p) \* (100 / (100 - W\_p))  
 val Q\_T\_H = (Q\_P\_H + 0.025 \* W\_p - A\_p) \* (100 / (100 - W\_p - A\_p))  
  
 val df = DecimalFormat("#.##")  
 df.*roundingMode* = RoundingMode.*HALF\_UP*

і після цього показуютья результати користувачу:

resultsTextView.*text* = """  
 Коефіцієнти переходу:  
 K\_PC: ${df.format(K\_PC)}  
 K\_PT: ${df.format(K\_PT)}  
  
 Склад сухої маси:  
 H\_C: ${df.format(H\_C)}  
 C\_C: ${df.format(C\_C)}  
 S\_C: ${df.format(S\_C)}  
 N\_C: ${df.format(N\_C)}  
 O\_C: ${df.format(O\_C)}  
 A\_C: ${df.format(A\_C)}  
  
 Склад горючої маси:  
 H\_T: ${df.format(H\_T)}  
 C\_T: ${df.format(C\_T)}  
 S\_T: ${df.format(S\_T)}  
 N\_T: ${df.format(N\_T)}  
 O\_T: ${df.format(O\_T)}  
  
 Нижча теплота згоряння:  
 Q\_P\_H: ${df.format(Q\_P\_H)} кДж/кг  
 Q\_C\_H: ${df.format(Q\_C\_H)} кДж/кг  
 Q\_T\_H: ${df.format(Q\_T\_H)} кДж/кг  
 """.*trimIndent*()

Сам зовнішній вигляд програми налащтовується у файлі “fragment\_task1”, який відповідає для UI для Завдання 1 і виглядає наступним чином:

A screenshot of a computer

Description automatically generated

При запуску контрольного прикладу виходять наступні результати:

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated

Та при запуску варіанту 7 з таблиці з варіантами наступні результати:

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated

Завдання 2

Код реалізації завдання 2 знаходиться у файлі MainActivity.kt у класі Task2Fragment. 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:

calculateButton.setOnClickListener **{** try {  
 val H\_g = inputFields[0].*text*.toString().*toDouble*()  
 val C\_g = inputFields[1].*text*.toString().*toDouble*()  
 val S\_g = inputFields[2].*text*.toString().*toDouble*()  
 val O\_g = inputFields[3].*text*.toString().*toDouble*()  
 val A\_g = inputFields[4].*text*.toString().*toDouble*()  
 val W\_p\_mazut = inputFields[5].*text*.toString().*toDouble*()  
 val Q\_g = inputFields[6].*text*.toString().*toDouble*()  
  
 val K\_GC = 100 / (100 - W\_p\_mazut - A\_g)  
  
 val H\_p = H\_g \* K\_GC  
 val C\_p = C\_g \* K\_GC  
 val S\_p = S\_g \* K\_GC  
 val O\_p = O\_g \* K\_GC  
 val A\_p = A\_g \* (100 / (100 - W\_p\_mazut))  
  
 val Q\_p = Q\_g \* (100 - W\_p\_mazut - A\_g) / 100  
  
 val df = DecimalFormat("#.##")  
 df.*roundingMode* = RoundingMode.*HALF\_UP*

і після цього показуютья результати користувачу:

resultsTextView.*text* = """  
 Коефіцієнт переходу до робочої маси:  
 K\_GC: ${df.format(K\_GC)}  
  
 Склад робочої маси:  
 H\_p: ${df.format(H\_p)}  
 C\_p: ${df.format(C\_p)}  
 S\_p: ${df.format(S\_p)}  
 O\_p: ${df.format(O\_p)}  
 A\_p: ${df.format(A\_p)}  
  
 Нижча теплота згоряння:  
 Q\_p: ${df.format(Q\_p)} МДж/кг  
""".*trimIndent*()

Сам зовнішній вигляд програми налащтовується у файлі “fragment\_task2”, який відповідає для UI для Завдання 1 і виглядає наступним чином:

Screens screenshot of a computer

Description automatically generated

При запуску контрольного прикладу виходять наступні результати:

A screenshot of a phone

Description automatically generated

Та при введенні довільних даних наступні результати:

A screenshot of a phone

Description automatically generated

**Висновок**

В результаті виконання практичної роботи №1 засвоїв основні арифметичні операції в мові Kotlin, особливості використання та створення функцій і класів. Навчився створювати та кастомізувати елементи у Android Studio та опрацьовувати введені користувачем значення.