Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Лабораторна робота № 1

з курсу: «*Кросплатформна розробка мобільних застосунків*»

**Виконав:**  
студент 4-го курсу,  
групи ТВ-01  
Школенко Денис Ігорович

**Перевірив:**

Недашківський О.Л.

Київ 2024

Лабораторна робота № 1

**Завдання:**

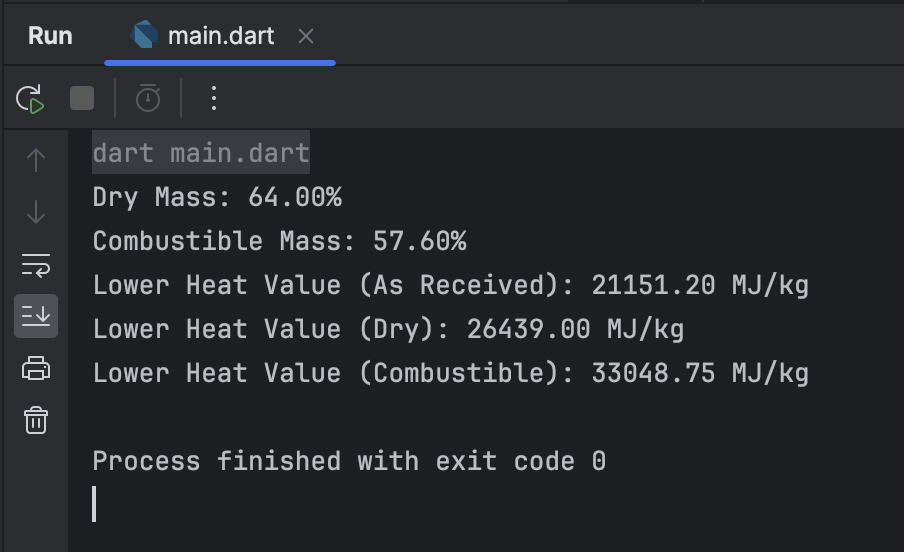
Створіть "Програмний калькулятор для розрахунку складу сухої та горючої маси палива та нижчої теплоти згоряння для робочої, сухої та горючої маси за заданим варіантом".

**Хід виконання:**

1. Аналіз завдання: Розглянули вимоги до калькулятора для обчислення складу палива та його теплоти згоряння.
2. Структурування коду: Вирішили структуру програми, включаючи класи для представлення даних палива (FuelData) та обчислення (методи calculateDryMass, calculateCombustibleMass, calculateLowerHeatCombustion).
3. Реалізація логіки обчислень: Написали логіку для обчислення сухої маси, горючої маси та теплоти згоряння.
4. Забезпечення надійності даних: Включили перевірку на коректність введених даних.
5. Тестування програми: Провели перевірку роботи програми з заданими вхідними даними.
6. Оформлення виводу: Налаштували вивід результатів з округленням до двох знаків після коми.

*/// Represents the composition and combustion calculations for a specific fuel.  
/// This class is immutable and provides methods to calculate dry mass, combustible mass,  
/// and lower heat value of combustion in different states.*class FuelData {  
 final double hydrogen;  
 final double carbon;  
 final double sulfur;  
 final double nitrogen;  
 final double oxygen;  
 final double water;  
 final double ash;  
  
 const FuelData({  
 required this.hydrogen,  
 required this.carbon,  
 required this.sulfur,  
 required this.nitrogen,  
 required this.oxygen,  
 required this.water,  
 required this.ash,  
 }) : assert(hydrogen + carbon + sulfur + nitrogen + oxygen + water + ash <= 100.0,  
 'The sum of components must not exceed 100%.');  
  
 */// Calculates the dry mass percentage.* double calculateDryMass() => 100 - water - ash;  
  
 */// Calculates the combustible mass percentage.* double calculateCombustibleMass() => calculateDryMass() - (oxygen + nitrogen + sulfur);  
  
 */// Calculates lower heat of combustion for the fuel in different mass states.* Map<String, double> calculateLowerHeatCombustion() {  
 const double cH = 339; // Carbon to Heat value constant  
 const double hH = 1030; // Hydrogen to Heat value constant  
 const double oH = 108; // Oxygen to Heat value constant  
 const double sH = 25; // Sulphur to Heat value constant  
  
 double lhvAsReceived = cH \* carbon + hH \* hydrogen - oH \* (oxygen - sulfur) - sH \* water;  
 double lhvDry = lhvAsReceived / (1 - water / 100);  
 double lhvCombustible = lhvAsReceived / ((1 - water / 100) - (ash / 100));  
  
 return {  
 'asReceived': lhvAsReceived,  
 'dry': lhvDry,  
 'combustible': lhvCombustible,  
 };  
 }  
}  
  
void main() {  
 const fuelData = FuelData(  
 hydrogen: 3.2,  
 carbon: 54.4,  
 sulfur: 2.3,  
 nitrogen: 1.0,  
 oxygen: 3.1,  
 water: 20.0,  
 ash: 16.0,  
 );  
  
 var combustionValues = fuelData.calculateLowerHeatCombustion();  
  
 print('Dry Mass: ${fuelData.calculateDryMass().toStringAsFixed(2)}%');  
 print('Combustible Mass: ${fuelData.calculateCombustibleMass().toStringAsFixed(2)}%');  
 print('Lower Heat Value (As Received): ${combustionValues['asReceived']?.toStringAsFixed(2) ?? 'N/A'} MJ/kg');  
 print('Lower Heat Value (Dry): ${combustionValues['dry']?.toStringAsFixed(2) ?? 'N/A'} MJ/kg');  
 print('Lower Heat Value (Combustible): ${combustionValues['combustible']?.toStringAsFixed(2) ?? 'N/A'} MJ/kg');  
}

**Результат роботи програми:**

****

**Висновок**

У результаті виконання лабораторної роботи №1 було розроблено програмний калькулятор на мові Dart, який успішно розраховує склад сухої та горючої маси палива та нижчу теплоту згоряння. Програма демонструє важливість структурного підходу до розробки програмного забезпечення, а також підкреслює значення надійності та точності в обчисленнях.

При порівнянні з мовою Kotlin, Dart виявився досить гнучким та ефективним у контексті даної задачі, надаючи чітку типізацію та простоту синтаксису. Незважаючи на те, що Kotlin може використовуватися для подібних завдань, особливо у розробці Android-додатків, Dart показав себе як потужний інструмент для швидкого розгортання та легкості тестування у рамках цієї лабораторної роботи. Вибір між Dart та Kotlin може залежати від конкретного використання та особистих переваг розробника.