1. **Data Engineering Bootcamp - Object-Oriented Programming (OOP) Assignment**

**Objective** :

The objective of this assignment is to implement a Bank Management System using Object-Oriented Programming principles, focusing on Inheritance and Abstraction concepts. The system will handle both Conventional and Sharia-compliant Savings Account types, include a login mechanism for account holders, and provide the functionality to export transaction history.

**Requirements**:

* 1. Create an abstract class **SavingsAccount** that will serve as the base class for both Conventional and Sharia-compliant Savings Account classes.
* The **SavingsAccount** class should have the following attributes:
  + account\_number: A unique identifier for the account.
  + account\_holder: The name of the account holder.
  + balance: The current account balance.
  + interest\_rate: The annual interest rate for the account.
  + transaction\_history: A list to store the transaction history for the account.
* The **SavingsAccount** class should have the following methods:
  + deposit(amount): Add the given amount to the account balance and record the transaction in the history.
  + withdraw(amount): Deduct the given amount from the account balance if sufficient funds are available and record the transaction in the history.
  + calculate\_interest(): Calculate and return the interest earned based on the current balance and interest rate.
  + display\_account\_info(): Display the account information, including the account number, account holder's name, and current balance.
* 2. Create two classes that inherit from **SavingsAccount**: **ConventionalSavings** and **ShariaSavings**.
* The **ConventionalSavings** class should have an additional attribute:
  + min\_balance: The minimum balance required to keep the account active. If the balance goes below this limit, a penalty will be applied.
* The **ShariaSavings** class should override the calculate\_interest() method to implement Sharia-compliant interest calculation. (interest\_rate = 0%)
* 3. Implement appropriate constructors for all classes to initialize the attributes.
* Write a **Bank** class that will manage multiple accounts. The **Bank** class should have the following methods:
  + register\_account(account): Register a new account in the bank.
  + login(account\_number): Verify the account number for login and return the corresponding account object.

**—(BONUS)---**

* + **export\_transaction\_history(account): Export the transaction history for a given account to local .xlsx and a SPREADSHEET file.**

[**Day2: OOP - Google Sheets**](https://docs.google.com/spreadsheets/d/1ojN5IITl16HRFvXFrKuEJQOzpjr5L-vu2dZQ_lKTVJg/edit#gid=7814554)

—-----------------

* Write a program to demonstrate the functionality of the Bank Management System:
  + Create instances of both ConventionalSavings and ShariaSavings accounts.
  + Register these accounts with the Bank.
  + Implement a login mechanism for account holders to access their respective accounts. (just simple login, user: S0001, pass: 12345. No need for hashing)
  + Prefix C for Conventional Account Prefix S for Sharia
  + Perform deposits and withdrawals on the logged-in accounts.
  + Display the account information and the interest earned for both accounts.

**—(BONUS)---**

* + Run the app and user interaction in terminal.
  + Export the transaction history for each account to local .xlsx and spreadsheets file using the export\_transaction\_history method of the Bank class.

[Day2: OOP - Google Sheets](https://docs.google.com/spreadsheets/d/1ojN5IITl16HRFvXFrKuEJQOzpjr5L-vu2dZQ_lKTVJg/edit#gid=7814554)

—-------------

Guidelines:

* Use proper naming conventions for variables, functions, and classes.
* Apply appropriate access modifiers for class attributes and methods.
* Handle edge cases, such as insufficient balance during withdrawals or invalid inputs.
* Make use of inheritance, abstraction, and override principles effectively.
* Utilize CSV file handling, pandas, gspread etc for transaction history export.

Submission Guidelines:

* Your submission should include the following:
  + Source code for both the single-threaded and multithreaded data processing functions.
  + Any additional utility functions or helper code used in the implementation.
  + Clear instructions on how to run the code and any specific dependencies required.
* Organize your code neatly and use meaningful variable names and comments.

—--

1. **Data Engineering Bootcamp - Multithreading Assignment**

**Assignment Overview**: In this assignment, you will explore the concept of computing parallelism in the context of data engineering. Parallelism is the technique of breaking down a large task into smaller sub-tasks and processing them simultaneously, utilizing multiple computing resources to achieve faster and more efficient data processing. You will work on building a data processing pipeline that leverages parallel computing to analyze a large dataset.

**Dataset**:

silahkan unduh seluruh pdf yang ada di sini url BATAM.xlsx.

**Assignment Tasks**:

* Read and download pdf:
  + Load the dataset from the provided CSV file into memory.
  + Preprocess the data if necessary (e.g., handle missing values).
* Single-threaded Data Processing:
  + Implement a single-threaded data processing.
  + Temukan peraturan mana saja yang mengandung kata “sanksi”, dan “berlaku sampai dengan”
  + Ex:  
    | kata | list peraturan |

| berlaku sejak | peraturan 1, peraturan 2, … |

| sanksi | peraturan 51, peraturan 52, … |

* + Calculate and print the following statistics for the entire dataset:
    - Total number of temperature readings.
    - Minimum, maximum, and average temperature values.
    - Calculate the average temperature for each location.
* Multithreaded Data Processing:
  + Modify the data processing function to support multithreading.
  + Divide the dataset into equal chunks and process each chunk concurrently using multiple threads.
  + Calculate the same statistics as in Task 2 for the entire dataset, but using multithreading this time.
* Performance Comparison:
  + Measure and compare the execution time of the single-threaded and multithreaded data processing functions.
  + Discuss your observations and findings regarding the performance improvement achieved using multithreading.

Submission Guidelines:

* Your submission should include the following:
  + Source code for both the single-threaded and multithreaded data processing functions.
  + Any additional utility functions or helper code used in the implementation.
  + A brief report documenting your findings and performance comparison results.
  + Clear instructions on how to run the code and any specific dependencies required.
* Organize your code neatly and use meaningful variable names and comments.