Lab 3 – Bank OOP

Objective

The goal of this lab is to use object-oriented programming (OOP) in Python to model a simple banking system.
You will design well-structured classes, use properties, operator overloading, error handling, and test code.
You should plan before coding and then implement your design in Python.

Task 1

Create the following classes in separate .py files:
- Account
- SavingsAccount
- Customer

You may also add additional helper classes if you want to use inheritance or composition.

Class: Account
- Attributes: account_number, balance, owner
- Read-only property: balance
- Methods: deposit(), withdraw(), transfer()
- Comparison operators based on balance (<, >, <=, >=)
- Equality (==) based on account_number
- __repr__ and __str__ implemented
- Raise TypeError/ValueError for invalid operations

Class: SavingsAccount (inherits from Account)
- Attribute: interest_rate
- Method: apply_interest() increases balance

Class: Customer
- Attributes: personal_number, name, accounts (list of Account)
- Methods: add_account(), total_balance()
- Comparison operators based on total balance
- Equality based on personal_number
- __repr__ and __str__ implemented

Bonus (Task 2): Bank class
- Attributes: name, customers (list of Customer)
- Methods: add_customer(), find_customer(), total_assets()
- Comparison operators based on total assets
- __repr__ and __str__ implemented

Hand-in

- Link to your public GitHub repository
- .py files for each class
- .ipynb notebook for manual tests
- (Optional) Unit test files

Assessment

Pass:
- Working solution that meets requirements
- Uses type hints, docstrings, and clear variable names
- Handles invalid input
- Git commits are meaningful

Pass with Distinction:
- Code is easy to follow and well structured
- Uses inheritance and composition effectively
- Reuses code (DRY principle)
- Includes Bank class and unit tests