ACIT4420 Mandatory assignment report

Deliverables

1. The main python code in the `bank_account.py` file.
2. This report.
3. Test cases in `test_cases.py`

How inheritance was used

Inheritance was used to reduce code duplication by allowing the derived classes (`SavingsAccount` and `CheckingAccount`) to extend generic functionality of the `BankAccount` class. The `BankAccount` class defines common behaviors such as depositing, withdrawing, and getting account info as well as defining the balance and account holder attributes. Both `SavingsAccount` and `CheckingAccount` inherit the methods and attributes of the base class which helps to centralize maintenance. This means that, if changes need to be made, the update would only need to be made in the `BankAccount` class.

Withdraw method

In the base class, the `withdraw()` method checks if there are sufficient funds before deducting the requested amount from the balance.
The `CheckingAccount` modifies this behavior to include a transaction fee for each withdrawal. Instead of replacing the withdrawal logic, it calls the super's (`BankAccount`)

withdrawal method, which was made to return a boolean, with a modified withdrawal amount that includes the transaction fee.
The use of `super().withdraw()` avoids code duplication in this case.

Optimizations and design decisions

Code reuse

Both derived classes call the parent constructor to initialize shared attributes, avoiding repetition.

Separation of concerns

`SavingsAccount` only extends functionality relevant to savings (interest handling). Meanwhile, `CheckingAccount` focuses on transaction fee management.

Error handling

Methods return a boolean to indicate success or failure. This enables more robust error handling (e.g. preventing overdrafts, rejecting invalid deposits).