Assignment 5 (Part 1)

Design a banking system using object-oriented programming concepts in Python.

Requirements:

- Create an abstract class called BankAccount that defines the common attributes and behaviors of different types of bank accounts such as SavingsAccount, CheckingAccount, and CreditCardAccount.
- The BankAccount class should have the following attributes:
    - account_number (integer)
    - account_holder_name (string)
    - account_balance (float)
- The BankAccount class should have the following methods:
    - deposit(amount: float) -> None - add the specified amount to the account balance
    - withdraw(amount: float) -> bool - remove the specified amount from the account balance if the balance is sufficient and return True, otherwise return False
    - get_account_balance() -> float - return the current balance of the account
- Create a subclass called SavingsAccount that inherits from the BankAccount class and has the following additional attributes:
    - interest_rate (float)
- The SavingsAccount class should have the following methods:
    - add_interest() -> None - add interest to the account balance based on the current interest rate
- Create a subclass called CheckingAccount that inherits from the BankAccount class and has the following additional attributes:
    - overdraft_limit (float)
- The CheckingAccount class should have the following methods:
    - withdraw(amount: float) -> bool - remove the specified amount from the account balance if the balance plus the overdraft limit is sufficient and return True, otherwise return False
- Create a subclass called CreditCardAccount that inherits from the BankAccount class and has the following additional attributes:
    - credit_limit (float)
- The CreditCardAccount class should have the following methods:

    - make_purchase(amount: float) -> bool - remove the specified amount from the credit limit if the limit is not exceeded and return True, otherwise return False
    - make_payment(amount: float) -> None - add the specified amount to the account balance to reduce the credit card debt
- Create instances of each subclass and test the functionality of the implemented methods.
- Use abstraction to hide the implementation details of the banking system from the end-users.
- Use polymorphism to handle different types of bank transactions.

Deliverables:

- A well-formatted Python code.