Bank Account Management--Project.py - D:/Python Full Stack Developer/Bank Account Management--Project.py (3.12.4)

File Edit Format Run Options Window Help

```
class BankAccount:
    def __init__(self,account_number,owner,balance=0):
        self.account_number=account_number
        self.owner=owner
        self._balance=balance #protected members, only subclasses can access this variable

    def deposit(self,amount):

        if amount>0:
            self._balance=self._balance+amount
            print(f" Amount Deposited: ${amount} into account {self.account_number}.")

    def withdraw(self,amount):
        if amount > self._balance:
            print("Insufficient funds.")
        elif amount > 0:
            self._balance = self._balance - amount
            print(f"Withdrawn: ${amount} from account {self.account_number}.")
        else:
            print("Invalid withdrawal amount.")

    def get_balance(self):

        return self._balance

    def display_details(self):
        print("Account Number: ",self.account_number)
        print("Owner: ",self.owner)
        print("Total Available Balance: ",self._balance)

class SavingsAccount(BankAccount):

    def __init__(self, account_number, owner, balance=0, interest_rate=0.02):
        super().__init__(account_number, owner, balance)
        self.interest_rate = interest_rate

    def apply_interest(self):

        interest=self._balance * self.interest_rate
        self_balance=self._balance+interest

        print("Applied Interest: ", self._balance)

class CurrentAccount(BankAccount):

    def __init__(self, account_number, owner, balance=0, overdraft_limit=500):
        super().__init__(account_number, owner, balance)
```

Ln: 24 Col: 28

Bank Account Management--Project.py - D:/Python Full Stack Developer/Bank Account Management--Project.py (3.12.4)

File Edit Format Run Options Window Help

```
        print("Applied Interest: ", self._balance)

class CurrentAccount(BankAccount):

    def __init__(self, account_number, owner, balance=0, overdraft_limit=500):
        super().__init__(account_number, owner, balance)
        self.overdraft_limit = overdraft_limit

    def withdraw(self, amount):
        """Override withdraw method to include overdraft limit."""
        if amount > self._balance + self.overdraft_limit:
            print("Insufficient funds, even with overdraft.")
        elif amount > 0:
            self._balance -= amount
            print(f"Withdrawn: ${amount} from account {self.account_number}. New balance: ${self._balance}")
        else:
            print("Invalid withdrawal amount.")

#Abstract class - blue print of all the classe
class Bank:

    def __init__(self):
        self.accounts={}

    def add_account(self,account):
        self.accounts[account.account_number]=account
        print(f"Account {account.account_number} added for {account.owner}.")

    def get_account(self, account_number):
        """Retrieve an account by account number."""
        return self.accounts.get(account_number, None)

    def display_all_accounts(self):
        """Display details for all accounts."""
        for account in self.accounts.values():
            account.display_details()

bank = Bank()

savings_account = SavingsAccount(account_number=101, owner="Alice", balance=1000)
current_account = CurrentAccount(account_number=102, owner="Bob", balance=500)
savings_account1 = SavingsAccount(account_number=103, owner="George", balance=2000)

bank.add_account(savings_account)
bank.add_account(current_account)
bank.add_account(savings_account1)
```

Ln: 24 Col: 28

Bank Account Management--Project.py - D:/Python Full Stack Developer/Bank Account Management--Project.py (3.12.4)

File Edit Format Run Options Window Help

```
#Abstract class - blue print of all the classe
class Bank:

    def __init__(self):
        self.accounts={}

    def add_account(self,account):
        self.accounts[account.account_number]=account
        print(f"Account {account.account_number} added for {account.owner}.")

    def get_account(self, account_number):
        """Retrieve an account by account number."""
        return self.accounts.get(account_number, None)

    def display_all_accounts(self):
        """Display details for all accounts."""
        for account in self.accounts.values():
            account.display_details()

bank = Bank()

savings_account = SavingsAccount(account_number=101, owner="Alice", balance=1000)
current_account = CurrentAccount(account_number=102, owner="Bob", balance=500)
savings_account1 = SavingsAccount(account_number=103, owner="George", balance=2000)

bank.add_account(savings_account)
bank.add_account(current_account)
bank.add_account(savings_account1)

print("\nAll bank accounts:")
bank.display_all_accounts()

print("\nPerforming operations:")
savings_account.deposit(500)
savings_account.apply_interest()
savings_account.withdraw(200)


current_account.deposit(300)
current_account.withdraw(1000)  # This should consider overdraft

# Display balances after operations
print("\nBalances after operations:")
print(f"Savings Account Balance: ${savings_account.get_balance()}")
print(f"Current Account Balance: ${current_account.get_balance()}")
```

Ln: 24 Col: 28

Bank Account Management--Project.py - D:/Python Full Stack Developer/Bank Account Management--Project.py (3.12.4)

File Edit Format Run Options Window Help

```python
#Abstract class - blue print of all the classe
class Bank:

    def __init__(self):
        self.accounts={}

    def add_account(self,account):
        self.accounts[account.account_number]=account
        print(f"Account {account.account_number} added for {account.owner}.")

    def get_account(self, account_number):
        """Retrieve an account by account number."""
        return self.accounts.get(account_number, None)

    def display_all_accounts(self):
        """Display details for all accounts."""
        for account in self.accounts.values():
            account.display_details()

bank = Bank()

savings_account = SavingsAccount(account_number=101, owner="Alice", balance=
current_account = CurrentAccount(account_number=102, owner="Bob", balance=5(
savings_account1 = SavingsAccount(account_number=103, owner="George", balanc

bank.add_account(savings_account)
bank.add_account(current_account)
bank.add_account(savings_account1)

print("\nAll bank accounts:")
bank.display_all_accounts()

print("\nPerforming operations:")
savings_account.deposit(500)
savings_account.apply_interest()
savings_account.withdraw(200)


current_account.deposit(300)
current_account.withdraw(1000)  # This should consider overdraft

# Display balances after operations
print("\nBalances after operations:")
print(f"Savings Account Balance: ${savings_account.get_balance()}")
print(f"Current Account Balance: ${current_account.get_balance()}")
```

IDLE Shell 3.12.4

File Edit Shell Debug Options Window Help

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: D:/Python Full Stack Developer/Bank Account Management--Project.py =
Account 101 added for Alice.
Account 102 added for Bob.
Account 103 added for George.

All bank accounts:
Account Number:  101
Owner:  Alice
Total Available Balance:  1000
Account Number:  102
Owner:  Bob
Total Available Balance:  500
Account Number:  103
Owner:  George
Total Available Balance:  2000

Performing operations:
 Amount Deposited: $500 into account 101.
Applied Interest:  1500
Withdrawn: $200 from account 101.
 Amount Deposited: $300 into account 102.
Withdrawn: $1000 from account 102. New balance: $-200

Balances after operations:
Savings Account Balance: $1300
Current Account Balance: $-200
>>>
```

Col: 28

Type here to search

30°C Mostly cloudy ENG 15:20 16-11-2024