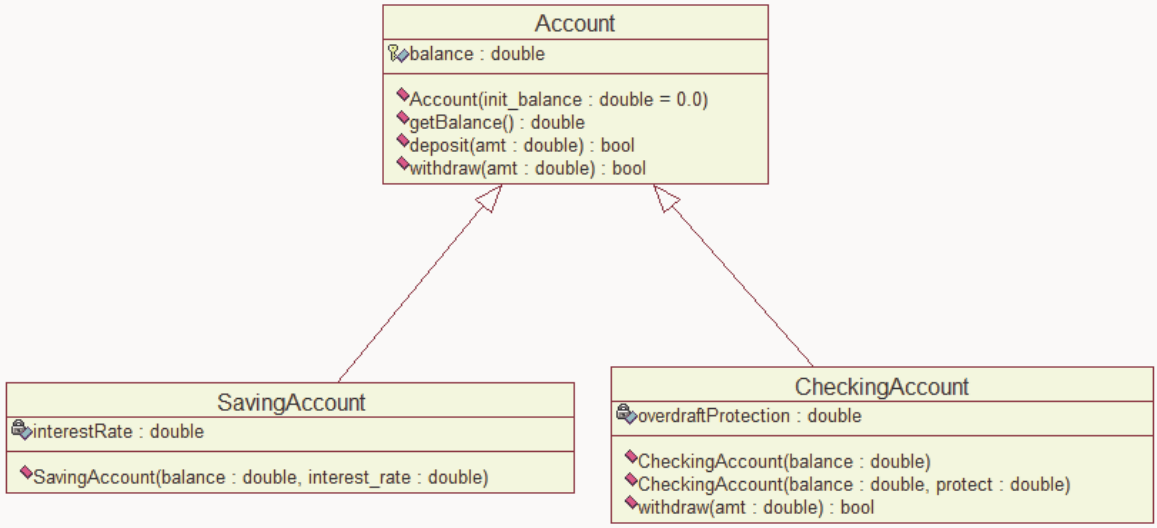
BLG332E
Object Oriented Programming

Practice Session 5

Exercise 8

In this exercise, you will create two subclasses of the Account class in the Banking project that we have studied in the previous Practice Session. These classes are SavingAccount and CheckingAccount.

Task 1: Change your working directory to `chap06/exercise1`



Task 2: Copy the previous project files (Bank.cpp, Bank.h, Customer.cpp, Customer.h, Account.h, Account.cpp) to the working directory.

Task 3: In the UML diagram, the `Account` class has changed. The `balance` attribute is now `protected` (indicated by the `#` character or the icon). Change the access mode of the `balance` attribute to `protected`.

Task 4: The `SavingAccount` class must inherit the `Account` class.

Task 5: It must include an `interestRate` attribute with type `double`.

Task 6: It must include a public constructor that takes two parameters: `balance` and `interest_rate`. This constructor must pass the `balance` parameter to the parent.

Task 7: The `CheckingAccount` class must inherit he Account class.

Task 8: It must include an `overdraftProtection` attribute with type double.

Task 9: It must include one public constructor that takes one parameter: `balance`. This constructor must pass the `balance` parameter to the parent constructor.

Task 10: It must include another public constructor that takes two parameters: balance and protect. This constructor must pass the balance parameter to the parent constructor.

Task 11: The CheckingAccount class must override the withdraw method. It must perform the following check: If the current balance is adequate to cover the amount to withdraw, then proceed as usual. If not and if there is overdraft protection, then attempt to cover the difference (balance-amount) by value of the overdraftProtection. If the amount needed to cover the overdraft is greater than the current level of protection, then the whole transaction must fail with checking balance unaffected.

Task 12: In the main Exercise1 directory, compile and run the TestBanking program.

The output should be

```
Creating the customer Jane Smith.
Creating her Savings Account with a 500.00 balance and 3% interest.
Creating the customer Owen Bryant.
Creating his Checking Account with a 500.00 balance and no overdraft protection.
Creating the customer Tim Soley.
Creating his Checking Account with a 500.00 balance and 500.00 in overdraft protection.
Creating the customer Maria Soley.
Maria shares her Checking Account with her husband Tim.

Retrieving the customer Jane Smith with her savings account->
Withdraw 150.00: false
Deposit 22.50: true
Withdraw 47.62: false
Withdraw 400.00: false
Customer [Soley, Maria] has a balance of 22.5

Retrieving the customer Owen Bryant with his checking account with no overdraft protection.
Withdraw 150.00: false
Deposit 22.50: true
Withdraw 47.62: false
Withdraw 400.00: false
Customer [Soley, Maria] has a balance of 22.5

Retrieving the customer Tim Soley with his checking account that has overdraft protection.
Withdraw 150.00: false
Deposit 22.50: true
Withdraw 47.62: false
Withdraw 400.00: false
Customer [Soley, Maria] has a balance of 22.5

Retrieving the customer Maria Soley with her joint checking account with husband Tim.
Deposit 150.00: true
Withdraw 750.00: false
Customer [Soley, Maria] has a balance of 150
```

Note that Jane's saving account and Owen's checking account fundamentally behave as a plain-old bank account. But Tim and Maria's joint checking account has 500.0 worth of overdraft protection. Tim's transactions dip into that protection and therefore his ending balance is 0.0. His account's overdraft protection level is 424.88. Finally Maria deposits 150.0 into this joint account; raising the balance from 0.0 to 150.0. Then she tries to withdraw 1000.0, which fails because neither the balance nor the overdraft protection can cover the requested amount.