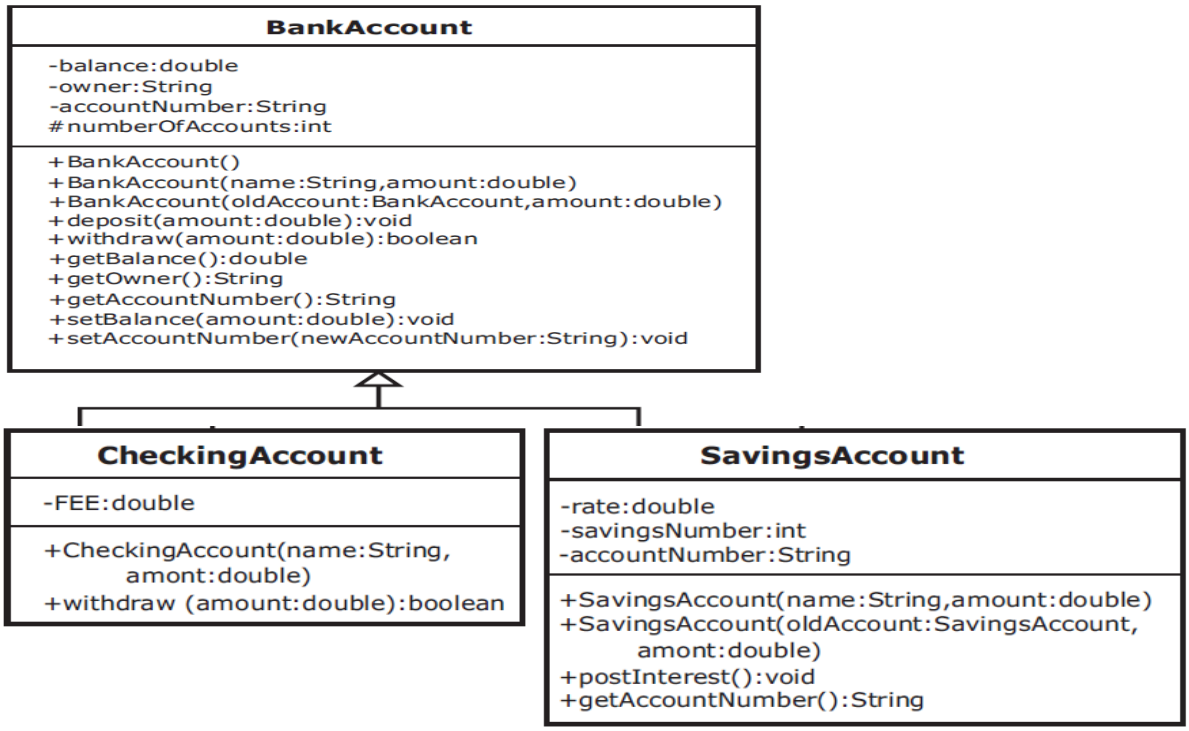
Lab Activity 6 - Inheritance

Objectives

• Be able to derive a class from an existing class
• Be able to define a class hierarchy in which methods are overridden and fields are hidden
• Be able to use derived-class objects
• Implement a copy constructor

Introduction

In this lab, you will be creating new classes that are derived from a class called BankAccount. A checking account ***is a*** bank account and a savings account ***is a*** bank account as well. This sets up a relationship called inheritance, where BankAccount is the superclass and CheckingAccount and SavingsAccount are subclasses. This relationship allows CheckingAccount to inherit attributes from BankAccount (like owner, balance, and accountNumber, but it can have new attributes that are specific to a checking account, like a fee for clearing a check. It also allows CheckingAccount to inherit methods from BankAccount, like deposit, that are universal for all bank accounts.You will write a withdraw method in CheckingAccount that overrides the withdrawmethod in BankAccount, in order to do something slightly different than the original withdraw method.You will use an instance variable called accountNumber in SavingsAccount to hide the accountNumber variable inherited from BankAccount.



Task #1 Extending BankAccount

1. Create the *BankAccount class* explained above using the sample code shown.
2. Create a new class called **CheckingAccount** that **extends BankAccount**.
3. It should contain a static constant **FEE** that represents the cost of clearing one check. Set it equal to 15 cents.

4. Write a constructor that takes a name and an initial amount as parameters. It should call the constructor for the superclass. It should initialize accountNumber to be the current value in accountNumber concatenated with –10 (All checking accounts at this bank are identified by the extension –10). There can be only one checking account for each account number. Remember since accountNumber is a private member in BankAccount, it must be changed through a mutator method.

5. Write a new instance method, **withdraw**, that overrides the withdraw method in the superclass. This method should take the amount to withdraw, add to it the fee for check clearing, and call the withdraw method from the superclass. Remember that to override the method, it must have the same method heading. Notice that the withdraw method from the superclass returns true or false depending if it was able to complete the withdrawal or not. The method that overrides it must also return the same true or false that was returned from the call to the withdraw method from the superclass.

6. Compile and debug this class.

Code Listing (BankAccount)

```
/**Defines any type of bank account*/
public class BankAccount
{
/**class variable so that each account has a unique
number*/
protected:
   static int numberOfAccounts = 100001;
/**current balance in the account*/
Private:
   double balance;
/** name on the account*/
  string owner;
/** number bank uses to identify account*/
  string accountNumber;
/**default constructor*/
public BankAccount()
{
balance = 0;
accountNumber = numberOfAccounts + "";
numberOfAccounts++;
}
```

```
public BankAccount(String name, double amount)
{
owner = name;
balance = amount;
accountNumber = numberOfAccounts + "";
numberOfAccounts++;
}

public BankAccount(BankAccount oldAccount, double amount)
{
owner = oldAccount.owner;
balance = amount;
accountNumber = oldAccount.accountNumber;
}
/**allows you to add money to the account
public void deposit(double amount)
{
balance = balance + amount;
}
/**allows you to remove money from the account if
public boolean withdraw(double amount)
{
boolean completed = true;
if (amount <= balance)
{
balance = balance - amount;
}
else
{
completed = false;
}
return completed;
}
/**accessor method to balance
public double getBalance()
{
return balance;
}
/**accessor method to owner
public String getOwner()
{
return owner;
}
/**accessor method to account number

public String getAccountNumber()
{
return accountNumber;
}
/**mutator method to change the balance
public void setBalance(double newBalance)
{
balance = newBalance;
}
/**mutator method to change the account number
@param newAccountNumber the new account number*/
public void setAccountNumber(String newAccountNumber)
```

```
{
accountNumber = newAccountNumber;
}
}
```

Task #2 Creating a Second Subclass

1. Create a new class called **SavingsAccount** that **extends BankAccount**.

2. It should contain an instance variable called rate that represents the annual interest rate. Set it equal to 2.5%.

3. It should also have an instance variable called savingsNumber, initialized to 0. In this bank, you have one account number, but can have several savings accounts with that same number. Each individual savings account is identified by the number following a dash. For example, 100001-0 is the first savings account you open, 100001-1 would be another savings account that is still part of your same account. This is so that you can keep some funds separate from the others, like an Eid account.

4. An instance variable called accountNumber that will hide the accountNumber from the superclass, should also be in this class.

5. Write a constructor that takes a name and an initial balance as parameters and calls the constructor for the superclass. It should initialize accountNumber to be the current value in the superclass accountNumber (the hidden instance variable) concatenated with a hyphen and then the savingsNumber.

6. Write a method called **postInterest** that has no parameters and returns no value. This method will calculate one month's worth of interest on the balance and deposit it into the account.

7. Write a method that overrides the **getAccountNumber** method in the superclass.

8. Write a copy constructor that creates another savings account for the same person. It should take the original savings account and an initial balance as parameters. It should call the copy constructor of the superclass, assign the savingsNumber to be one more than the savingsNumber of the original savings account. It should assign the accountNumber to be the accountNumber of the superclass concatenated with the hypen and the savingsNumber of the new account.

9. Compile and debug this class.

Task #3 Creating a Main Class

```
Create driver class and run the code
------------------ The End☺----------------------
```