30/40

## - From the Chapter 10 Video Notes:

In the 6th video (section 10.7), as discussed by the narrator, are the 4 properties for the Employee class grouped together physically in memory?

***No.***

In the 9th video (section 10.13), what does the author state is a good use of object initializers?

***When there is no other way to call the constructor with only a single value.***

I’m looking for “LINQ to create anonymous objects”

SavingsAccount class

- annualInterestRate was to have been a static variable (all accounts would share that rate)

- Technically the author stated to have methods CalculateMonthlyInterest amd ModifyAnnualInterestRate (he didn’t state to define properties as you did)

Here’s how I’d change your code to have it fit to what was asked for:

class SavingsAccount

{

//variables

// private decimal annualInterestRate = 0m;

private static decimal annualInterestRate = 0m;

private decimal savingsBalance = 0m;

public SavingsAccount()

:this(0m) //don’t set the rate

{

}

public SavingsAccount(decimal startingBalance)

{

// ModifyInterestRate = interest;

SavingsBalance = startingBalance;

// CalculateMonthlyInterest = startingBalance;

}

public decimal SavingsBalance

{

get

{

return savingsBalance;

}

set

{

if (value >= 0m)

savingsBalance = value;

else

throw new ArgumentOutOfRangeException("savingsBalance", value,

"savingsBalance can't be negative, 0 or greater please.");

}

}

public static void ModifyInterestRate(decimal rate)

{

if (rate >= 0m)

annualInterestRate = rate;

else

throw new ArgumentOutOfRangeException("annualInterestRate", rate,

"annualInterestRate can't be negative, 0 or greater please.");

}

public void CalculateMonthlyInterest()

{

savingsBalance += savingsBalance \* (annualInterestRate / 12m);

}

}

Test program:

When displaying dollar amounts, only display 2 decimal digits (make this a “standard” practice)

What you would now need:

SavingsAccount.ModifyInterestRate(.04m); //set the rate for all accounts

SavingsAccount saver1 = new SavingsAccount(2000m); //create an account

saver1.CalculateMonthlyInterest(); //calculate

Console.WriteLine("Balance for saver1 is {0:f2}", saver1.SavingsBalance);

SavingsAccount saver2 = new SavingsAccount(3000m);

saver2.CalculateMonthlyInterest();

Console.WriteLine("Balance for saver2 is {0:f2}", saver2.SavingsBalance);

SavingsAccount.ModifyInterestRate(.05m); //change the rate

saver1.CalculateMonthlyInterest();

Console.WriteLine("Balance for saver1 is {0:f2}", saver1.SavingsBalance);

saver2.CalculateMonthlyInterest();

Console.WriteLine("Balance for saver2 is {0:f2}", saver2.SavingsBalance);