**COP 3855 – Web Systems Development**

**C# Programming In-Class Activity**

**Real Estate Investment Program**

You should review the problem specification in below figure and make sure you understand the problem definition. Several values must be entered into the program. These values must be entered as string variables and then parsed into numeric fields, so that arithmetic can be performed.

Two separate classes are to be developed. Creating a separate class for the real estate object enables this class to be used by many different applications. One application is to produce a listing showing the cash flow from the investment. Other applications might include determining total investment dollar amounts or locations of investments. If the characteristics of real estate objects are abstracted out, many applications can reuse the class.

The problem specification for the Real Estate Investment example is displayed. The problem description is as follows. 
How much cash flow profit is a rental investment generating? Create an application to determine what the cash flow is for a real estate investment used as a rental. 
Design an object oriented solution. Use two classes.
For the real estate property class, characteristics such as the year the home was built, purchase price, and street address will help identity the current state of an object. The real estate object also has a monthly income amount from rent and a monthly expense characteristic. Include a method to determine what the monthly earning, or cash flow, is based on deducting the total monthly expenses from the monthly rental income.
In the second class, instantiate an object of the real estate property class. Call the constructor that creates an object using year built, purchase price, and street address. Allow the user to input the yearly taxes and insurance expenses. The monthly utilities costs should also be considered. In this application class, calculate a monthly expense based on the inputted values. Set the appropriate data field in the real estate property object. 
Write code in the application class to display the property address and the expected cash flow for a given month.

Below table lists the instance variables needed for the RealEstateInvestment problem.

|  |  |  |
| --- | --- | --- |
| **Data item description** | **Type** | **Identifier** |
| Year the home was built | int | yearBuilt |
| Location of the home | string | streetAddress |
| Original purchase price | double | purchasePrice |
| Total expenses for average month | double | monthlyExpense |
| Rental premium per month | double | incomeFromRent |

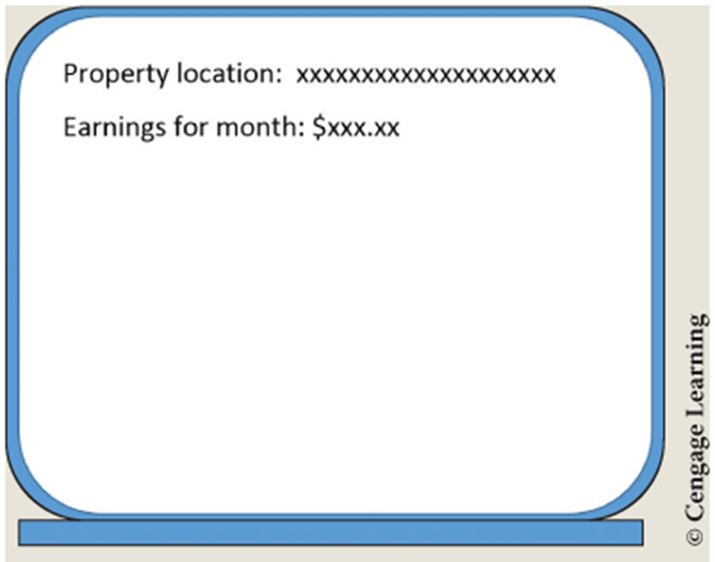
The class that is using the RealEstateInvestment class also needs data. As noted in the problem specification, the application class allows the user to enter values for expenses. Below table identifies some of the local variables needed by the application class.

|  |  |  |
| --- | --- | --- |
| **Data item description** | **Type** | **Identifier** |
| Cost of insurance per year | double | insurance |
| Amount of taxes per year | double | taxes |
| Estimated monthly utility costs | double | utilities |
| String value for inputting values | string | inValue |

To illustrate the use of constants, a default rental rate is set as a constant value. The identifier and preset constant value will be

RENTAL\_AMOUNT = 1000.00.

The desired output is to display the address of a property and the expected cash flow for a given month. Below figure shows a prototype of the desired final output. The xxx.xx is placed in the prototype to represent the location in which the calculated values should appear.



The object-oriented approach focuses more on the object. The real estate property has both data and behavior characteristics that can be identified. Class diagrams are used to help design and document these characteristics. Below figure shows the class diagrams for the RealEstateInvestment example.

The class diagram for the Real Estate Investment example is shown. The name of the class, data members, and methods are shown for the classes Real Estate Investment and Property App.
Name of the class, Real Estate Investment; Data member, year Build; data type, i n t; Data member, purchase Price; Data type, double. Data member, street Address; Data type, string. Data member, monthly Expense; Data type, double. Data member, income From Rent; Data type, double. Method, Determine Monthly Earnings of; Return type, double. Method, Real Estate Investment of. Name of the class, Property App. Object, invest 1; Class, Real Estate Investment. Data member, RENTAL underscore AMOUNT; Data type, double; Value = 1000.00. Method, Get Expenses; Return type, double.

The class diagrams show neither the properties needed nor the local variables that might be needed by specific class methods. As you learned earlier, properties are new to C# and reduce the need to write mutators and accessors for the private instance variables. Below table lists the data members that will have properties defined and indicates whether both get and set are needed. The name of the property is also shown.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data member identifier** | **Property identifier** | **Set** | **Get** |
| yearBuilt | YearBuilt |  | ✓ |
| streetAddress | StreetAddress |  | ✓ |
| purchasePrice | PurchasePrice |  | ✓ |
| monthlyExpense | MonthlyExpense | ✓ | ✓ |
| incomeFromRent | IncomeFromRent | ✓ | ✓ |

During design, it is important to develop the algorithm showing the step-by-step process to solve the problem. Structured English, also called pseudocode, is suited for the object-oriented methodology. In addition to the Main( ) method, two additional methods—DetermineMonthlyEarnings( ) and GetExpenses( )—need to be designed. Below figure shows part of the Structured English design for the RealEstateInvestment example. Additional objects should be instantiated to test the other constructors, properties, and methods of the RealEstateInvestment class.

After the algorithm is developed, the design should be checked for correctness. When you desk check your algorithm, begin in Main( ). When you encounter method calls, keep your location in that method, go to the called method, and perform the statements. When you finish with the called method, return back to your saved location.

Test your algorithm with the following data:

Insurance: 650.00

Taxes: 1286.92

Utilities: 250.00

Use a calculator and write down the results you obtain. After you implement your design, you can compare these results with those obtained from your program output.

The data for the Real Estate Investment example in Structured English is displayed.
The main method from the Property App class is defined below.
Line 1. main, left parenthesis, right parenthesis.
Line 2, indented once. Construct Real Estate Investment object invest 1
Line 3, indented twice. year Built = 2004.
Line 4, indented twice. purchase Price = 150000.
Line 5, indented twice. street Address = open quotes, 65 t h Steet, close quotes.
Line 6, indented once. set invest 1, period, Monthly Expense = Get Expenses, left parenthesis, right parenthesis.
Line 7, indented once. set invest 1, period, Income From Rent = RENTAL underscore AMOUNT.
Line 8, indented once. Write, open quotes, Property Location, colon, close quotes + invest 1, period, Street Address.
Line 9, indented once. Write, open quotes, Earnings For Month, colon, close quotes, +, invest 1, period, Determine Monthly Earnings, left parenthesis, right parenthesis.
The Get Expenses method from the Property Appl class is defined below.
Line 1. Get Expenses, left parenthesis, right parenthesis.
Line 2, indented once. Write message to enter insurance rate.
Line 3, indented once. Read insurance rate into string in Value.
Line 4, indented once. insurance = parse, left parenthesis, in Value, right parenthesis.
Line 5, indented once. Write prompt to enter taxes.
Line 6, indented once. Read tax amount into string in Value.
Line 7, indented once. taxes = parse, left parenthesis, in Value, right parenthesis.
Line 8, indented once. Write prompt to enter utility charges.
Line 9, indented once. Read utility charges into string in Value.
Line 10, indented once. utilities = parse, left parenthesis, in Value, right parenthesis.
Line 11, indented once. return, left parenthesis, insurance, division slash, 12 + taxes, division slash, 12 + utilities.
The Detemine Monthly Eearnings method from the Real Estate Investment Class is defined below.
Line 1. Determine Monthly Earnings, left parenthesis, right parenthesis.
Line 2, indented once. return, left parenthesis, income From Rent minus monthly Expense, right parenthesis.