Application Development I
Milestone 1 - Documenting Budget Files

Worth: Depending on the final number of milestones, this milestone will be worth 6.5-8% (All milestones together will comprise 65% of your final grade, there will be 7-10 milestones)

Due: **Date as per LEA,** 8am the following morning.

Late Penalty: No lates without prior permission.

Cheating: Please remember that showing or giving your code to a classmate is cheating. If you want to help a friend that is stuck, try guiding them to think of things differently. Give yourself the time to work through roadblocks. If you are stuck, please MIO your teacher or ask during class time.

Plagiarism: You should not be pasting code from anywhere without acknowledging it. If you do bring in some code with the acknowledged source, it should be a small proportion of your code.

Submit: submitted onto LEA in the "Assignments" Section, a .zip file containing the Visual Studio solution. This should include:

- All the .cs files with the appropriate documentation added to the code
- The generated website containing the BudgetCode API documentation

---

Learning Objectives:

- Learn to read someone else's code
- Learn to document public APIs
- Learn to generate API documentation through tools such as docfx console

Requirements:

1) Use XML documentation tags to create comments for ALL public classes, properties and methods, respecting the documentation guidelines (see distributed document in Lea).
    - All parameters for any given method *must* have a description
    - All methods *must* have a summary that describes what the method does
    - If a method throws *any* exceptions, it must be indicated
    - All return objects *must* have a description, not just the type of object returned
    - Additional comments may be necessary if the method is doing something unexpected or weird

2) **ALL** methods must have example code that uses the most common feature of that method.

    a) All the methods in HomeBudget.cs should have an extensive example with inputs and outputs shown. **See example of an example, at the end of this document.**

    b) All methods in the other classes should have an example, but it could be simple. For example, for Expenses.List(), this example code would be sufficient:

```
Expenses expenses = new Expenses();
expenses.Add(DateTime.Now, (int)Category.CategoryType.Expense,
            23.45, "textbook");
List<Expense> list = expenses.List();
foreach (Expense expense in list)
      Console.WriteLine(expense.Description);
```

3) For the starting point of the API, HomeBudget:
    - Include a small code example that demonstrates the typical usage of the class.
    - Include a link to Categories and Expenses in the description of the class.

NOTE: you do not have to replace any comments that are ‘within’ the method.

Marking Criteria:

- Proper spelling and grammar
- Clarity of comments (brief is ok, but must be readable)
- Thorough (no missing comments)
- Comments respect the guidelines - see distributed document.
- Proper code examples for each method and the HomeBudget class.

Application Development I
Milestone 1 - Documenting Budget Files

Example example

one example (for GetBudgetItems)

It is NOT complete, you have to complete it.

- The *balance* is not described… what is that number representing?
- Is the list sorted, if so, how?
- Missing examples (Show examples that answer the following questions):
    - What happens if the filter flag is set to true?
    - What is the Category ID for?
    - Is the Date range inclusive or exclusive (i.e. does it include items from the start date, or only after the start date)?

```
/// <example>
///
/// For all examples below, assume the budget file contains the
/// following elements:
///
/// <code>
/// Cat_ID  Expense_ID  Date                    Description                    Cost
///    10       1       1/10/2018 12:00:00 AM   Clothes hat (on credit)         10
///     9       2       1/11/2018 12:00:00 AM   Credit Card hat                -10
///    10       3       1/10/2019 12:00:00 AM   Clothes scarf(on credit)        15
///     9       4       1/10/2020 12:00:00 AM   Credit Card scarf              -15
///    14       5       1/11/2020 12:00:00 AM   Eating Out McDonalds            45
///    14       7       1/12/2020 12:00:00 AM   Eating Out Wendys               25
///    14      10       2/1/2020 12:00:00 AM    Eating Out Pizza                33.33
///     9      13       2/10/2020 12:00:00 AM   Credit Card mittens            -15
///     9      12       2/25/2020 12:00:00 AM   Credit Card Hat                -25
///    14      11       2/27/2020 12:00:00 AM   Eating Out Pizza                33.33
///    14       9       7/11/2020 12:00:00 AM   Eating Out Cafeteria            11.11
/// </code>
///
/// <b>Getting a list of ALL budget items.</b>
///
/// <code>
/// <![CDATA[
///  HomeBudget budget = new HomeBudget();
///  budget.ReadFromFile(filename);
///
///  // Get a list of all budget items
///  var budgetItems = budget.GetBudgetItems(null, null, false, 0);
///
///  // print important information
```

Application Development I
Milestone 1 - Documenting Budget Files

```
///  foreach (var bi in budgetItems)
///  {
///      Console.WRiteLine (
///          String.Format("{0} {1,-20}  {2,8:C} {3,12:C}",
///             bi.Date.ToString("yyyy/MMM/dd"),
///             bi.ShortDescription,
///             bi.Amount, bi.Balance)
///       );
///  }
///
/// ]]>
/// </code>
///
/// Sample output:
/// <code>
/// 2018/Jan/10 hat (on credit)       ($10.00)     ($10.00)
/// 2018/Jan/11 hat                     $10.00        $0.00
/// 2019/Jan/10 scarf(on credit)      ($15.00)     ($15.00)
/// 2020/Jan/10 scarf                   $15.00        $0.00
/// 2020/Jan/11 McDonalds             ($45.00)     ($45.00)
/// 2020/Jan/12 Wendys                ($25.00)     ($70.00)
/// 2020/Feb/01 Pizza                 ($33.33)    ($103.33)
/// 2020/Feb/10 mittens                 $15.00     ($88.33)
/// 2020/Feb/25 Hat                     $25.00     ($63.33)
/// 2020/Feb/27 Pizza                 ($33.33)     ($96.66)
/// 2020/Jul/11 Cafeteria             ($11.11)    ($107.77)
/// </code>
///
/// </example>
```