

## BA272 Assignment 3

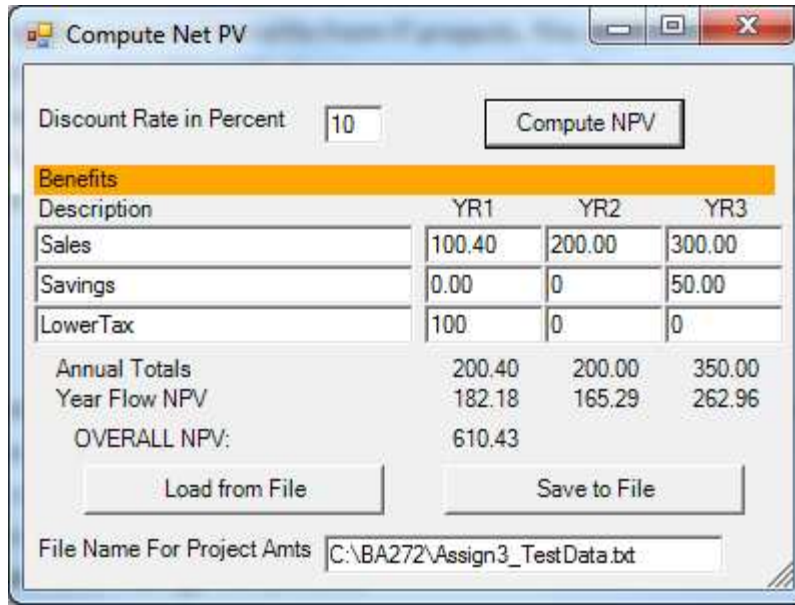
This assignment is designed to introduce graphical user interfaces created WITHOUT the Visual Studio IDE. It is important that you understand how forms are created manually if you are to understand what is really happening when you use the available “Power Tools.” Also, you will be asked to open files, catch exceptions, and use arrays.

- Creating simple forms and event handling
- Reading and writing files
- Catching exceptions
- Do some light processing of the text in the file (e.g., split, trim, and double.parse)
- Using arrays

Your coded result should require only a single .cs file (no other project files). Store that .cs file and the required test example text file in the `assign3\MyWork` folder. I recommend that you then try making a new project using that code and testing it. Also, please consider the grading criteria listed at the bottom of this page; your grade will be based on those criteria.

To get all that in, we need some fairly extensive directions. Here they are.

StapleMakers Inc. computes expected benefits from IT projects. This application computes the net present value of estimated future benefits. It will store projected benefits in and read a list of benefits from a simple text file. The application will use a simple Windows form with labels, textboxes, and buttons. A picture of a reasonable screen is shown below. The layout is up to you, but be sure the requirements are met. The “load file” button causes the program to open the named file and load its contents into the form. The “save file” button causes a new file to be created. Any time a number textbox is changed, the computations are to be cleared. When the “Compute” button is clicked, the present value computations should be performed.



**Inputs:**

- As many as three quantifiable benefits for the project with a title, and a projected cash flow to occur at the beginning of each of 3 years: year 1 (one year from now), year 2 (two years from now), and year 3 (3 years from now).
- A designated discount rate.
- A file name in which to store the inputs for future use.
- Buttons as described above to manage the process.

**Output:**

- Total expected cash flows for years 1, 2 and 3.
- The net present value (NPV) of each of those summarized flows based on the designated discount rate.
- An overall NPV.

Net Present Value of each flow is computed as:

Flow Amount \*  $1 / (1 + \text{discount rate})^{\text{Year}}$ . For example, a flow in the third year of \$50 based on a 10% discount rate would be \$37.57.

**Requirements:**

- Benefit details should be stored in a file in this format. An example file is provided – be sure your program can load the example file.

```
DiscountRateInPercent    10
Description  Year1      Year2      Year3
Sales        100.40    200.00    300.00
Savings      0.00      0         50.00
LowerTax     100       0         0
```

Note that a tab character separates the columns.

- Load values from the file into an array. *Hint: using the split command makes this very easy! Be sure you understand it.*
- Create and call a form with the required objects/components (p. 147-153).
- Create the needed event handling methods to facilitate program flow.
- Write out a file containing the data from the form.
- Compute net present value by calling a method you create.
- Account for errors that can occur while reading a file or accepting data from a text box. *Hint: use try/catch and tryparse.*

**Make sure you understand these issues in preparation for a quiz or exam:**

- What are events and how does event handling work? How are the events “thrown” by a form connected to handling methods?
- Basic text file handling includes opening and closing a stream and using ReadLine() and Peek.
- What is “passed” to an event handling routine?

**Evaluation Criteria (out of 100):**

- Documentation: (15)
  - o (5) Useful and appropriate comments are included.
  - o (5) Form layout is neat and understandably labeled.
  - o (5) Variable names indicate the type of variable and are meaningful.
- Components: (65) About half each for including the component and half if the component does what it should
  - o (10) A method defined in the program gives back NPV. *Hint: computing a power can be tricky!*
  - o (10) A form is created with:
    - buttons for all functions (2)
    - textboxes for benefit details (3)
    - textboxes for file name and discount rate (3)
    - several explanatory labels (2)
  - o (10) A try/catch block protects file access code from system errors.
  - o (10) Tryparse is used to avoid errors when accepting data from textboxes.
  - o (10) Data is read in from a file.
  - o (5) Data is written out into a file.
  - o (10) Data from the file is moved into an array.
- Function: (20)
  - o (10) The program compiles and executes.
  - o (10) The program produces correct results.

**Thoughts:**

- Identify some limitations of this program.

- Although this assignment demonstrates that you can make forms by executing code, Visual Studio allows you to “paint” forms – drawing a picture and setting attributes to make a form-based program. This can save a lot of time and debugging. Still, in the end, all that Visual Studio does when you draw a form is generate code that looks a lot like the code you used here. Hopefully, we will get a chance to see how it does it next week.
- There are different ways to perform looping. Be prepared to explain the relative advantages of `Do While` for accepting console input, `While` for reading a file, `For` for doing 12 months of estimates, and `Foreach` for processing values in an array.
- Handling events is a powerful programming paradigm. What are some issues that might arise from using them? Can you think of how you might benefit from making your own class throw its own event? (That last question is tough!)