Jonathan Sánchez
VBA Project Write-up

Debt Elimination Exel-eration

Executive Summary

I volunteer teach personal finance at libraries for a hobby, and I've created my own program to help others become debt free. My philosophy with debt elimination is that it is a mental game. There needs to be motivating factors in place in order to stick to the program and become debt free. One great motivating factor is simply the elimination of a debt. While teaching, I've noticed people are overwhelmed with many different forms of debt (frequently 10+ debts per person) and sitting down and prioritizing the debts in an elimination plan is a daunting task. I've decided, therefore, to automate this process using VBA.

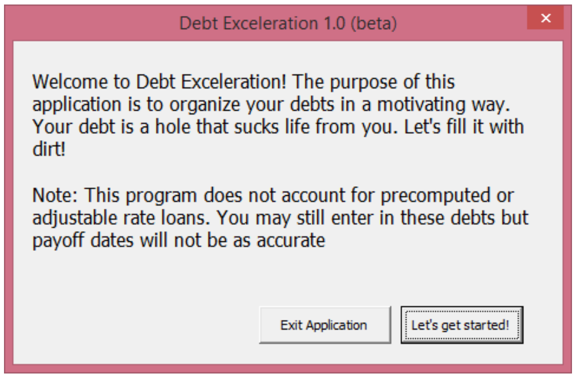
My 'Debt Exceleration' VBA project uses an algorithm that prioritizes debts in a debt-snowball schedule based primarily on the size of the balance (90% weight), and secondarily on the interest rate (10% weight). The goal is to motivate while also saving on interest expense where possible. I plan to distribute this project to past, current, and future students.

Implementation

I will discuss the nature of this project in three sections: ***Input, Calculate,*** and ***Modify***.

Input

Upon opening the program for the first time the user will immediately see a user form with some basic information:



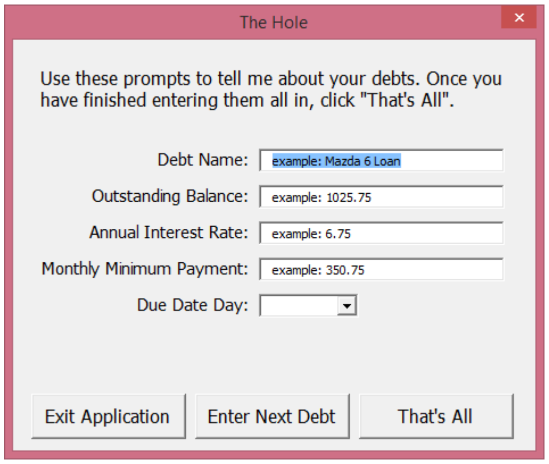
Welcome to Debt Exceleration! The purpose of this application is to organize your debts in a motivating way. Your debt is a hole that sucks life from you. Let's fill it with dirt!

Note: This program does not account for precomputed or adjustable rate loans. You may still enter in these debts but payoff dates will not be as accurate

Debt Exceleration 1.0 (beta)
This prompt only pops-up if it is the user's first time opening the application. The prompt contains two buttons: 'Exit Application' & 'Let's get started'.

<u>Exit Application</u>: In order to control the user's experience, I require them to go through the prompts. Therefore, the top right 'X' on the input user prompt is disabled. This is why the 'Exit Application' option is available. When the user selects 'Exit Application' the program checks if this is the only excel workbook open, if it is then it closes Excel. If other workbooks are open, than it only closes this workbook. The workbook is intentionally not saved upon closing.

<u>Let's get started</u>: Pushing this button simply moves the user to the next prompt.



Use these prompts to tell me about your debts. Once you have finished entering them all in, click "That's All".

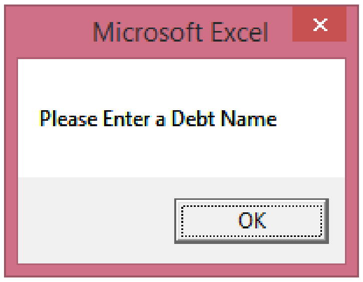
Debt Name:	example: Mazda 6 Loan
Outstanding Balance:	example: 1025.75
Annual Interest Rate:	example: 6.75
Monthly Minimum Payment:	example: 350.75
Due Date Day:	

The Hole
I teach that debt is a gaping hole and you need a shovel to fill it in with dirt. The size of the hole and shovel can vary. Thus, you will see this terminology throughout the program.

This user form takes in the debt information from the users. The fields are pre-populated with examples of values that the user can input.

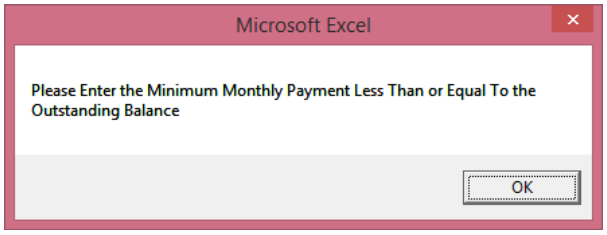
'The Hole' user form has three buttons: 'Exit Application', 'Enter Next Debt' and 'That's All'.

Exit Application: This button has the same functionality as the 'Exit Application' on the first user form. Therefore, you may refer to the explication in the first user form. The 'x' on 'The Hole' has also been disabled.



Please Enter a Debt Name

Enter Next Debt: The program checks to make sure the prepopulated fields have changed before allowing the user to move to the next prompt. If the fields are still the same, it tells the user exactly where the problem is (see prompt on the left). Also, I limit the keyboard to only allow numbers and decimals as input in the 'Outstanding Balance', 'Annual Interest Rate', and 'Monthly Minimum Payment'. The 'Due Date Day' only allows numbers by also excluding decimals. Any other attempted keyboard press will result in a 'beep' warning.

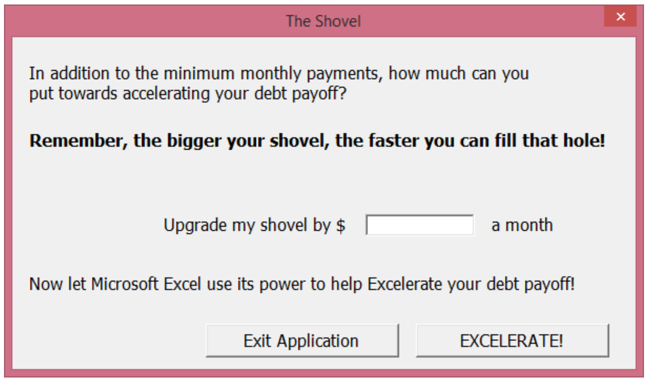


Please Enter the Minimum Monthly Payment Less Than or Equal To the Outstanding Balance

'The Hole' user form also has other forms of data validation. For example, a minimum payment greater than the balance would not make sense. The user would therefore receive a prompt letting them know of the error (left).

After the data has been validated, the program checks if dynamic arrays have already been defined. If they have not than dynamic arrays are defined for each of the values with one position in each array storing inputted debt information. Then, because the user selected 'Enter Next Debt', the user form is called again to gather information on the next debt. With each subsequent debt the dynamic arrays are redefined with one more slot and the debt information is stored.

That's All: By pressing this button, the data is validated one last time and the final debt is added to the dynamic arrays. The user form closes and the next user form is displayed.



In addition to the minimum monthly payments, how much can you put towards accelerating your debt payoff?

Remember, the bigger your shovel, the faster you can fill that hole!

Upgrade my shovel by $ a month

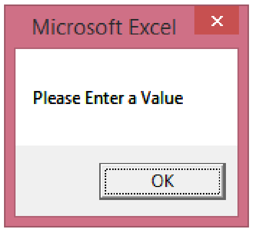
Now let Microsoft Excel use its power to help Excelerate your debt payoff!

The Shovel

This final prompt takes input on the "extra" amount of money the user plans to put towards paying off debt. I call this amount the "Shovel".

This user form has two buttons: 'Exit Application' and 'EXCELERATE!'.

Exit Application: This button has the same functionality as it did in previous prompts. The 'x' is once again disabled on this form.



Please Enter a Value

EXCELERATE!: This button validates that the user entered a value. If a valid value is not entered the user receives a prompt (left). Also, the program will only allow number and decimals to be entered and will 'beep' if other keys such as letters are attempted.

After storing the 'Shovel' value. The program calls a sub procedure to do all the hard work.

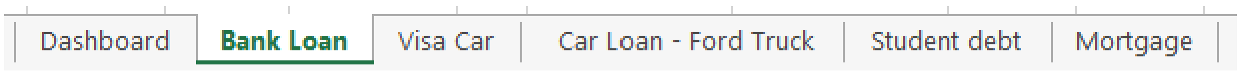
Calculate

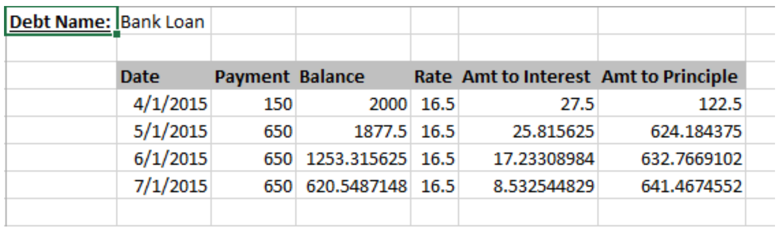
The ‘Calculate’ portion operates in three main steps: **rank debts, create debt schedules, create dashboard**.

Rank Debts

The program runs my algorithm on each debt and creates a priority score which it stores in an array. All arrays are then resorted according to the scores. After this runs, the first priority debt will be stored in the first slot of all arrays, the second priority in the second slot, etc. This is a huge benefit to the user as I have found that prioritizing debts is one of the main reasons people fail at trying to eliminate debt with a debt snowball.

Create Debt Schedule





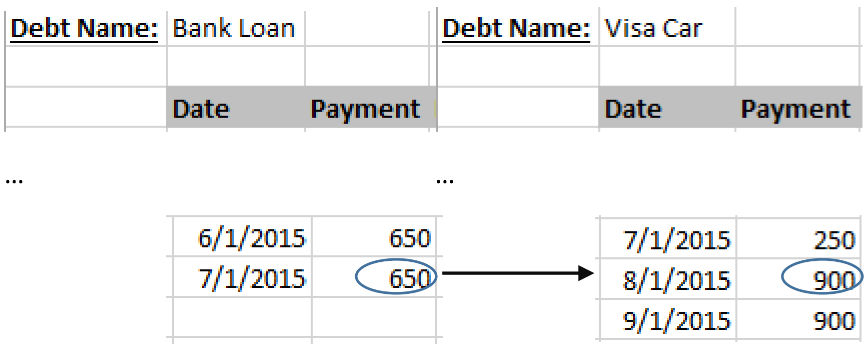
Debt Name: Bank Loan

Date	Payment	Balance	Rate	Amt to Interest	Amt to Principle
4/1/2015	150	2000	16.5	27.5	122.5
5/1/2015	650	1877.5	16.5	25.815625	624.184375
6/1/2015	650	1253.315625	16.5	17.23308984	632.7669102
7/1/2015	650	620.5487148	16.5	8.532544829	641.4674552

The program then creates and formats a new worksheet in order of priority for each debt. These worksheets display the debt schedules with columns for the following information: Date, Payment, Balance, Rate, Amount to Interest, and Amount to Principle. These worksheets are created from the information stored in the dynamic arrays created to store the debt information. All calculations and formatting take place within the VBA code.

The dates are done according to the minimum payoff date occuring on the payment date of each month.

As part of the calculation process, a new array is created to capture each debt’s estimated payoff date. A ‘snowball’ payment is calculated by adding the ‘shovel’ to the first priority debt minimum payment. It is immediately applied to the first priority debt in the ‘payment’ column. While the first priority debt is being paid down, all subsequent debts uses their minimum payment in the ‘payment’ column. The month following the first priority payoff date, the second priority debt then receives the snowball (first priority minimum payment plus shovel amount). The snowball continues through all debts until all debts are paid off.

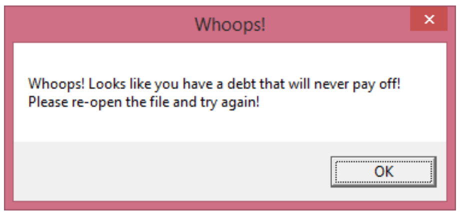


Debt Name: Bank Loan

Date	Payment
…	
6/1/2015	650
7/1/2015	650

Debt Name: Visa Car

Date	Payment
…	
7/1/2015	250
8/1/2015	900
9/1/2015	900



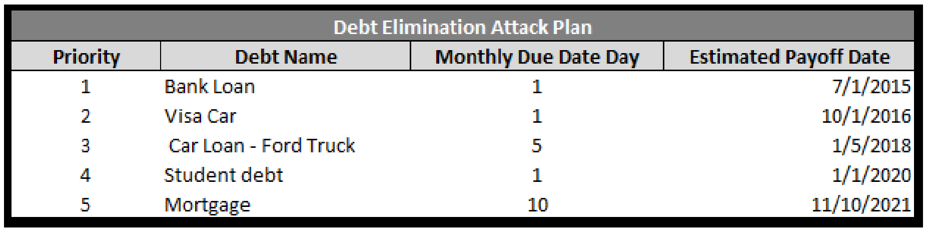
Whoops!

Whoops! Looks like you have a debt that will never pay off!
Please re-open the file and try again!

OK

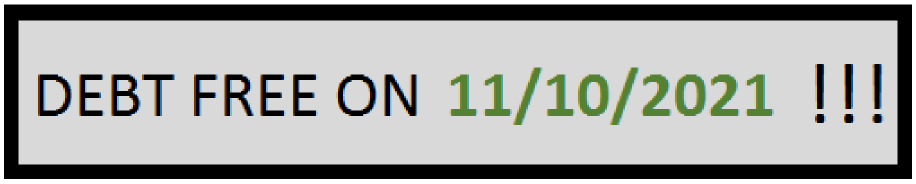
In the case where a user enters a debt that can never be paid off a new prompt is generated. Following this prompt, the application will close without saving, running the same sub procedure that the ‘Exit Application’ buttons use.

Create Dashboard



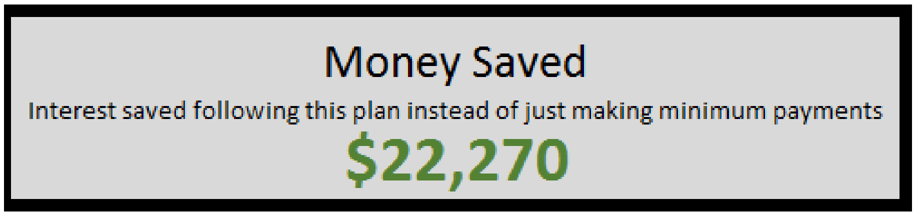
Debt Elimination Attack Plan			
Priority	Debt Name	Monthly Due Date Day	Estimated Payoff Date
1	Bank Loan	1	7/1/2015
2	Visa Car	1	10/1/2016
3	Car Loan - Ford Truck	5	1/5/2018
4	Student debt	1	1/1/2020
5	Mortgage	10	11/10/2021

The next phase is to create the dashboard in the first worksheet. The program first creates and displays the debt plan by listing all debts in order of priority.



DEBT FREE ON 11/10/2021 !!!

Next the estimated debt free date is displayed by choosing the last payoff date from the payoff date array.

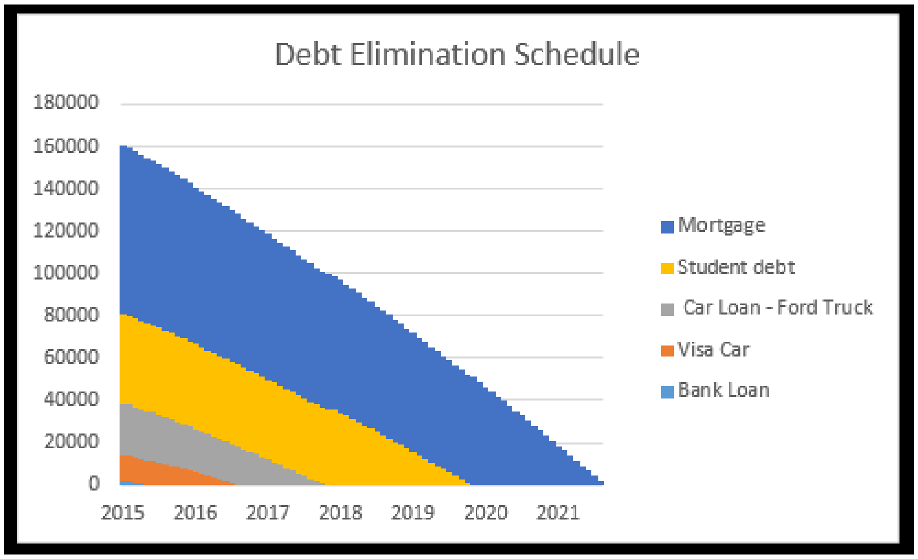


Money Saved

Interest saved following this plan instead of just making minimum payments

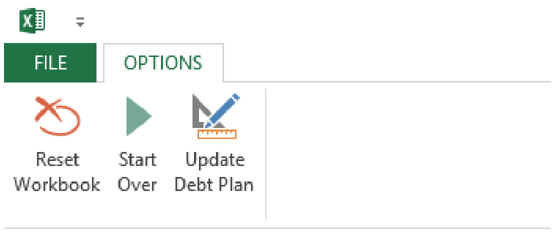
$22,270

The program then calculates the interest spent eliminating debt if the debt plan is followed and the interest that would have been spent illuminating debt if the user only made minimum payments for the life of the debts. These numbers are then compared to calculate the amount of money saved by following the Debt Exceleration plan as opposed to making minimum payments.



Finally, a debt elimination graph is created, formatted and displayed. This graph takes data from all the debt worksheets (which is a dynamic number depending on the number of debts the user has). Again, all the work in creating this graph is done in VBA.

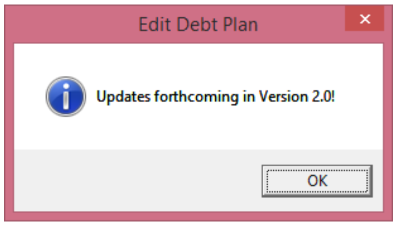
Modify



Once the program has created the dashboard and debt schedule it is ready for the user. I have limited the ribbon to only the functionalities I would like the user to have access to which is an options tab created specifically for this program. It has three options: Reset Workbook, Start Over, and Update Debt Plan.

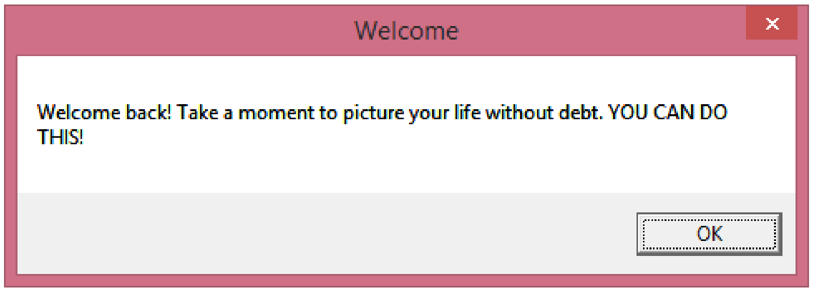
Reset Workbook: Clicking this button clears the dashboard and deletes all debt schedules. Essentially, it puts it back as a clean slate. This button does not do anything if the file has already been cleared.

Start Over: If the worksheet has been cleared, then this button begins the prompts from the start. Otherwise it does nothing (the ‘reset’ button must be selected prior to the ‘star over’ button).



Updates forthcoming in Version 2.0!

Update Debt Plan: This button will be used in version 2.0 to modify the ‘shovel’ size and make changes to the debts (i.e. car was sold, eliminating a debt). Currently, if pressed it displays the prompt to the left.



Welcome back! Take a moment to picture your life without debt. YOU CAN DO THIS!

Finally, the program can be saved and reopened for later use. Once the workbook has been opened for the first time, saved, and reopened the prompt to the left will be displayed upon opening instead of pushing the user through prompts again.

<u>Discussion of learning and Difficulties Encountered</u>

- This program uses dynamic arrays very heavily. I struggled with the loop that modifies various dynamic arrays to store the debt values (balance, rate, etc.). This loop would run when triggered by the button in the user form. However, since I reuse the user form for subsequent debts the loop is used for each debt. This poses a problem because the first time through the loop the arrays have not been defined yet and, therefore, could not be re dimmed. After struggling for a little while on how to approach this, I found on a module of code online with various programmed array functions such as IsArrayAllocated() that I could use in my logic to see if an array has already been defined. Problem solved.
- Another difficulty was in creating the graph used on the dashboard. This graph has to take input from an undefined number of worksheets and format it with the x and y ranges appropriate for the data. In other words, the graph is completely dynamic depending on the data the user enters. I was able to tackle this using various loops, properties, and arrays.
- One challenge that I was able to solve faster than I anticipated, but still took some problem solving, was rolling over the debt snowball from one debt to the next appropriately when creating the debt worksheet schedules. I was able to solve this by checking the previous debt in the payoff array and comparing it to the date column before populating the payment amount.
- I was proud of my ability to validate information in the user forms: only allowing the user to type certain characters into the fields, validating the inputs, and giving the appropriate prompts so the user can correct the inputs.
- Calculating the money saved was mathematically challenging. I use various time value of money excel functions in my VBA code along with arrays to assist with this.
- I enjoyed learning how to have code run based on events such as opening the workbook.
- Modifying the ribbon in Excel was new to me. I learned how to both hide and create new tabs and buttons.
- I tested my code thoroughly which resulted in discovering additional challenges, such as how to have my program avoid blowing up because a debt will never pay off.
- One challenge I did not fully resolve was creating a button in the ribbon that cleared the worksheet and reset all global variables AND THEN start up the prompts again. I ended up making this into two buttons with the second button needed to start the prompts up again. The difficulty was resetting all the variables without using “END” in my code.

The above represents a sample of my challenges and learnings in building this program. I could probably type another 3 pages on this topic.

<u>Assistance</u>

As I stated earlier, I loaded a module into my code in order to use IsArrayAllocated() to check if an array had already been defined. Other than this code, the only assistance I received was from small miscellaneous google searches that gave me direction in my own code.