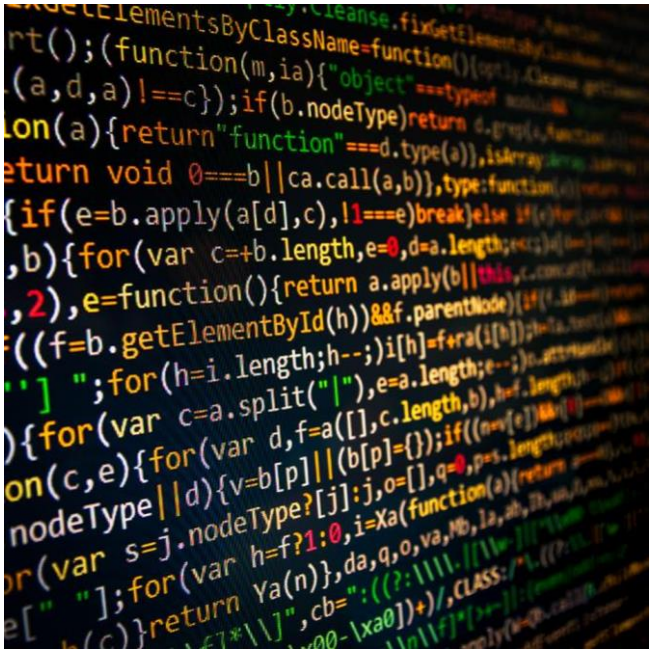


BUDGETING PROGRAM

Gabriel Da Silva 2017 (N0726678)

Software Engineering 1

Program Brief

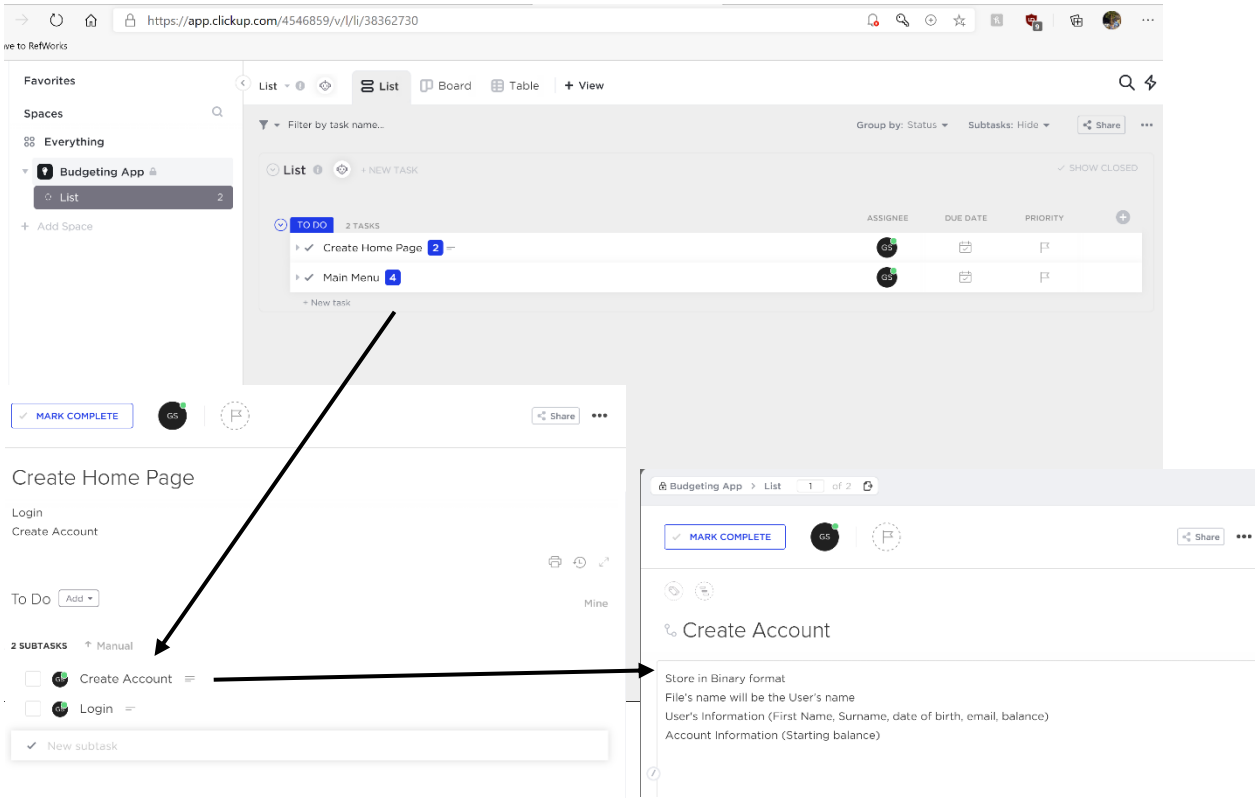
The program design is intended to be a basic budgeting application, where the user, would be able to create an account and insert their name, surname, date of birth, email, password, and the starting balance amount in British pounds. Once the account is created, the user would be able to login using their username “which is created automatically by the system” and their password.

After logging in, the user has four options, the first one to display their personal information. The second will allow them to check their balance with an additional option to see their statement of previous transactions. The third option will let them add a new transaction, and lastly the fourth option allows them to exit their account.

Software Approach

The first step of my development was to identify what classes I would want the compiler to have, which was a main page and the user account page. In the main page the user would have three options which was to create account, log in and exit the program. And in the user account the user would have four option as stated above.

Therefore, since I had an idea of the main functionalities, I initially approached the project using a pseudo-code. With the website ClickUp; I was able to design the pseudo-code as tasks, with sub-task of the more detailed problems of the project.



The main reason why I took this route is because, majority of my feedback received of previous work stated that I struggled to breakdown the problem, which would made difficult for me to solve the issue. Therefore, by using the pseudo code, it allowed me to understand what was intended in each part of the program, subsequently it became easier to design the syntax. However, the disadvantaged it was that it took me longer to finish the project, as I had to design the pseudo-code and to compile the code for the actual program. Additionally, the pseudo-code will not show where exception can happen or why it is happening (Bitesize, 2021). Therefore, to strike this issue, I used prototypes. Firstly, I compiled each part required, inserting all the fields, methods and functions required, and after I would test each time with different possibility which an user could insert either on purpose or by mistake, such as; inserting an empty string, or inserting a letter when an integer is required. Using this strategy allowed me to thoroughly check each step of the program, and making sure that that the program is running smoothly. However, this was very time consuming and the only reason I used this method was because, I was the only programmer involved in this project, so I knew all the code used, and because it was a small project with a few numbers of features. On the other hand, if there were more people involved, or the project was bigger, I would have only tested after

finishing each main part, e.g. (create account/ logging), this way it would save me a bit of time.

When the whole program was finished, I ran a final test creating my account and checking every feature of the program; by login in and adding new transactions. Once I finished, I asked a colleague to use the program, she created her account, and also logged in using her information, I was stood next to her to see if any exception would be thrown by the compiler when there were more than one account saved.

Problems Encountered

Login and password

The main problem which I had faced was to be able to consistently load the information of the same account. This was easily solved, passing the variable as argument to each method/function. I also had problem with the login system. I wanted to ask the user for the login username only once. I had tried many different possibilities, such as create a static field for the password, which then I realised every time I created a new account the password would change for every account. I also tried to do each method at the time. However, I was required to ask the username twice, first time to search and opened the file, and the second time to login. I then changed how each file would be named; instead of using the name of the user, I changed to the user's username, this way I would only ask the user once for their username. However, I also worked around with the order of the loadAccounts, accountLogin, and accountPassword methods. Firstly, the console would ask the user for the username; which would then be passed on to the loadAccount method in order to check if a file with that name existed; if the file exist it would open the file, and then check if the username string existed and matched with what was saved inside the user's account. Lastly, the console would ask the user for the password. The compiler would then check if what the user typed matched with what is saved in the file, by using

the accountPassword method which was used the accountLogin method. However, if the file does not exit the user would enter a loop asking the user to re-insert the username so he/she could proceed with the next step. The code was compiled this way to improve the code quality, I only made accountLogin method public and the other two is private, this way future programmers would not forget to use the other two methods if he/she would like to use the method in a different part of the program.

Checking Email

Another problem that I had faced, was with the emailChecker Method, in this method I make sure that the use is inserting a valid email. Because I did not know any class or any code that could verify the string. Therefore, I used stackoverflow to search for help. In order to solve my problem, I used the Net.Mail namespace, and the MailAddress class, whi allowed me to initialise a new instance with a valid email address, if the user insert an invalid email the console will throw an exception and return a Boolean variable with the value of false, the user would stay inside a loop; with the conditions to be looping if the Boolean variable was false.

Code Quality

In order to improve the code quality of this project, I have tried to encapsulate as much code as possible. By making the fields of the UserAccount class private, using constructors to easily set the value of the fields of each object, which in this case is each user account. Additionally, instead of creating multiple variables for each account, I have used a dictionary to create a collection of accounts, I used the username as the key and the constructor as value. This allowed me to manipulate the account in different methods by passing the dictionary and the username as arguments. Additionally, making sure that the only public methods were the essential ones, such as the ones which checks string, integers and decimals since these were used numerous of times. And, the methods that allows the user to create an account and to login. The advantage is that it made the main class very clear and organised, by hiding all the code that is involved with the account, while it all happens in the UserAccount class; such as, loading the file and verifying the password; all these functionalities happens inside the login method. However, if the program were to expand and create different classes, this could become an issue. Since the login, load the file and verify the password is all in one public method, it makes the program a bit rigid; if future programmers would only want to reuse the AccountPassword method in a

different class. I also approached the project using modular programming, by creating small reusable procedures, for asking the user for a value, and checking if the value is a correct string, int or decimal. This allowed me to save a lot of repeated code. Moreover, the configuration of both menus is written in a text file. This made the compiler a lot clear and it also made easier if; in the future, a business or anyone would like to add more functionalities to the program. However, they will also need to add more if and switch statements so the user can choose the new functions. Additionally, because each menu has different amount of if statements for functionality loaded from the file, I had to repeat the code to load both menus. But, the functions loading each menu has been placed in different classes to improve the quality. However, the file can not contain any white spaces, otherwise an exception will be throw, where the number and values in the array can not be read by the compiler.

References

Bitesize, 2021. *Pseudo-Code - Algorithms - Edexcel - GCSE Computer Science Revision - Edexcel - BBC Bitesize*. [online] BBC Bitesize. Available at: <https://www.bbc.co.uk/bitesize/guides/z7kkw6f/revision/2> [Accessed 4 January 2021].