Brief and Time Planning:

I've been asked by the IT company I work for, to write a place and create a working program for the Ranui family that meets the specifications for a clothing allowance app. This app will allow the parents to keep track of their three teenage children's (Nikau, Hana, Tia) annual clothing allowance. The details of their allowance are as follows:

- Each child has an annual clothing allowance of $300.
- Each child is not allowed to overspend their allowance.
- If any child has more than $50 left in their clothing allowance at the end of the year, they will be allocated a bonus.
- The bonus is an activity of the child's choice.

The program I plan to create is for the IT company I work for and should work for people of all races, gender, and religion (the only requirement being that they can read English as the program will be written in English). This will allow the parents and other caregivers to more effectively, monitor their three daughters' clothing allowance, and look at how much allowance should be given, if their children are wise with their money, or if they need some lessons regarding good money spending /saving techniques, etc. Thus, further allowing them to help their children create good spending habits.
The program constructed will be based on the specifications outlined on request:

- Access certain files
- Asking for the file the user wants to use
- Show the names of each of the three children
- Show the amount of money left in their clothing allowance
- Show whether they are still on target to get the bonus
- A way of entering an amount spent on an item of clothing
- A way of selecting the child whose allowance the money will be deducted from.
- Use object-oriented programming
    - clearly define a class for the family
    - clearly define an object for each child of the family

After all these specifications are covered, I will then create the program/app to complete the required specifications and carry out the purpose of the program (record the three children's financial status and allow them to make withdrawals and deposits to purchase clothes/accessories and shows whether they are eligible for a bonus at the end of the year).

I plan to split my project into 5 time periods (each around a week-long)

In my first period, I would like to finish the planning process of the program, creating flowcharts/pseudo codes, finish the timeline, plan, and prepare inputs and outputs names, purposes that I plan to use in my program, their types, scope and how they will be tested against the specification. However, as they're only 2 working periods in the first week I will be slightly flexible with the finishing date for the first period (by giving myself the weekend to finish up and go over the work done so far if need be). **So, this period should be finished by 29/07/2022.**

In my second period, I will more comprehensively confirm program functions and plan out definitions, and individual parts to the code, before cross-referencing it with the Te Kura modules. Then I will begin to write the program based on the planning done in the first period. I'll use the list of inputs and outputs to create definitions for each module of the code and use the data provided to me to read the file and use it to carry out the process.
After completing small segments of the code (e.g. reading the file, determining the money in the account, working whether) I will get a peer within my PAD class to check whether the outline of the code makes sense to them (so that I'm not the only one who understands its)

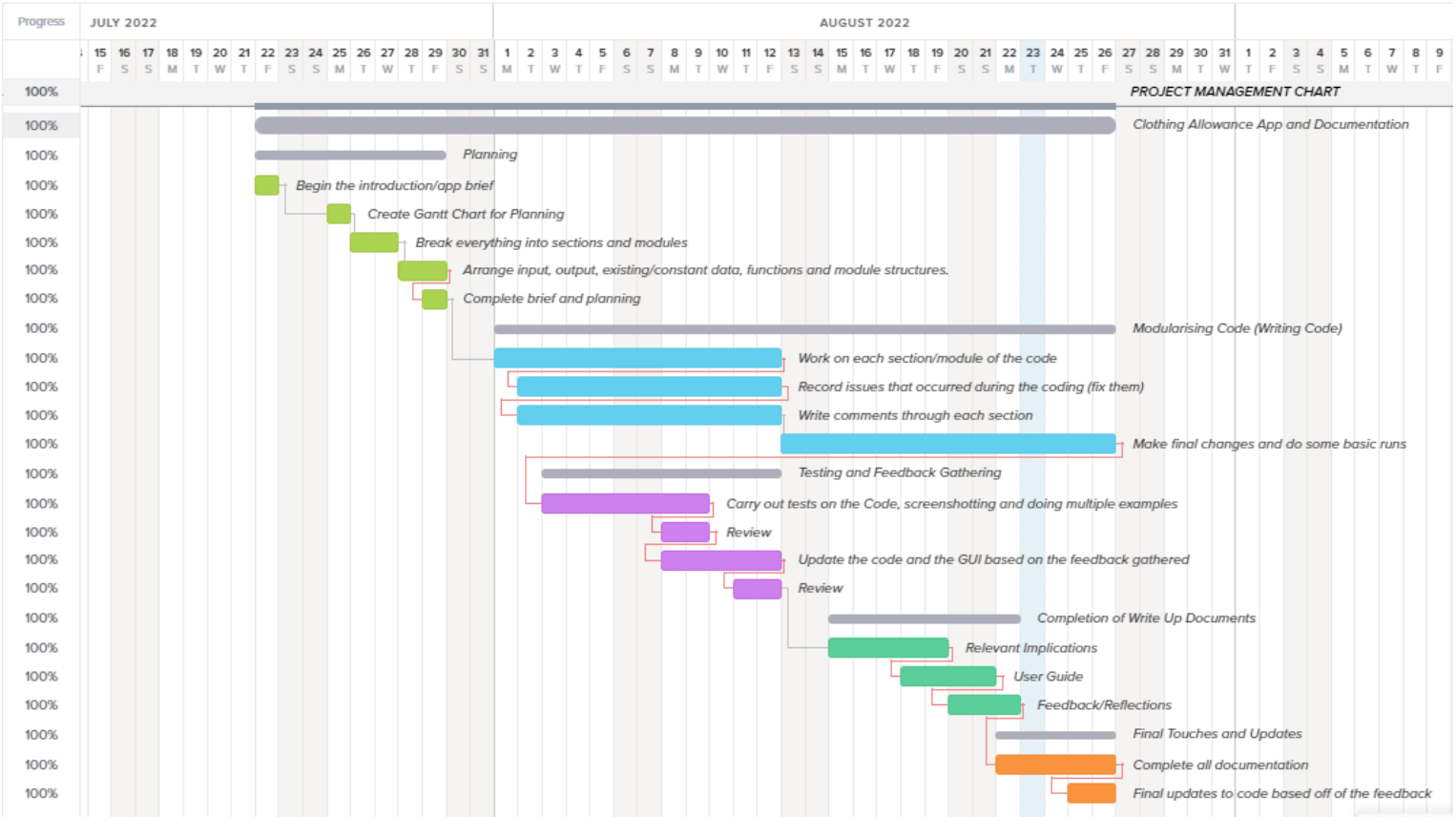
before asking them to run the code and check for errors. **So, this period should be finished by 05/08/22.**

In my third period, I will run test runs with my clients to ask them if there is anything missing that they'd like or any changes or additions they'd like to make to the program. I'll also ask people with no programming experience such as members in my family to see whether the program isn't overly complicated and can be handled well by those with limited (or no) understanding of programming (this is important as the family may not be too good with such technology, so if the program is too difficult to understand without knowing how to program it will not be an effective program). **So, this period should be finished by 12/08/22.**

In my fourth period, I will mainly work on testing, debugging, and making my program/code more user-friendly, to make the program run as smoothly as possible and carried out its respective tasks. I will use any of the recommendations given by my peers, client, and family to improve my program. I will make sure to include the updates I've made in my testing document. **So, this period should be finished by 19/08/22.**

My final period will be making any last finishing touches, writing the user guide for the clients (and their operators) to use, and any changes/updates to my code or documents. I will get feedback from my teacher as we all as my Te Kura tutor and following their suggestion before, finally rerunning all my tests to make sure the appropriate results come out and try to crash the code, before finally submitting the code on **26/08/22.**

Gantt Project Management Chart:



Resources: For this process, I will use the following the school computer, LucidChart, Python 3.7. Wing IDE (on both my personal laptop and school computer), Tekura for more specific feedback and help, my time, my family, peers, and client (to make suggestions, etc),

I will also use word documents to present the testing, relevant implications, user guide, feedback/reflections, and planning/writeups.

Input Information:

Name of Inputted Variable:	Data Types:	What is it used for:
amount	DoubleVar	This is the of money the user would like to withdraw or deposit from their specific account, the variable is a float value.

Functions:

Name of Inputted Variable:	Data Types:	What is it used for:
Account	Class	The class is the outline used to create the Object 'Account', the Account class stores the details of each account and has methods to deposit and withdraw money from their account,
__init__	def	This function is always executed when the class is being initiated. This function is used to assign values to object properties or other operations that are necessary to do when the object is being created, using self, name, and balance as its variables.
deposit	def	Is the method used to add 'amount' to the balance
withdraw	def	Is the method used to subtract 'amount' from the balance
get_data	def	Checks whether the file exists and if so, reads the file and separates it to find the accounts and their balance, by reading the lines and separating by commas. If the file doesn't exist, then a new file is created with the name clothing_allowance_app_file.txt, with the accounts and balances preset in the file, before rerunning the definition.
create_name_list	def	A function to get the account names and put them into an empty list 'name_list' by running through the account_list and appending each account to the the name_list.
bonus_check	def	This definition checks whether the account has enough money in the account to receive a bonus at the end of the year, if it does then a bonus message is displayed on the app stating they will receive the bonus else a savings message will be displayed on the app urging them to save some more money to receive the bonus at the end of the year.
update_balance	def	This function updates the balance by opening the file and then going through each account in the account_list before writing into the file the balances for each specific account, before saving to the file and closing it.
deposit_money	def	Displays messages in the app regarding the transaction that has taken place, and to which account it had occurred or an error message stating that an invalid value has been entered
withdraw_money	def	Displays messages in the app regarding the transaction that has taken place, and to which account it had occurred with or an error message stating that an invalid value has been entered or there are insufficient funds in the account to make the withdrawal.

manage_action	def	This definition manages all the functions running through the accounts in order, checking whether the chosen_action is Deposit or Withdraw and then selecting a certain definition depending on the answer, before updating the balance, checking whether the account is able to receive the allocated bonus, etc. This definition sets most of the other definitions in action.
account_list	list	Account list uses the file to create a list of accounts (as objects), where each object is an 'account', which is then used throughout the GUI and other definitions, to carry out certain actions (i.e. selecting an account to carry out an action and selecting a particular amount).
name_list	list	Uses the definition create_names _list to create a list of account names within the file and stores them as a string.

Output Information:

Names of Outputs:	**Output Format:**	**Other Information:**	**Data type/structure printed:**
clothing_allowance_app_file with changes to bank balance	The file contains the three accounts (Nikau's Account, Hana's Account, and Tia's Account) as well as the bank balances of each of these accounts.	Can be accessed through the folder with the app to show the files separately, if the file doesn't exist a new file is created under the same name.	File
balance	As amounts are deposited or withdrawn the balance will change and be shown in the app at the bottom frame (for all three accounts), and this will also be saved to the file.	Initially starts at 300 dollars for the year, and changes as money are deposited and withdrawn.	Float
The GUI/app itself and all its features	The GUI will be formed as the program is started, **messages** will be displayed as well as **accounts**, **balances**, **actions** choices, **submit** buttons, etc	Is written in the code, and will show up as soon as the program is initiated.	Graphical User Interface

Constant and Existing Data:

Constants, available data	**Scope:**	**Values Set:**

clothing_allowance_app_file	It is a pre-existing data/file that is created automatically or already set up by the parents, that contains the number of accounts and the balances of each account	Nikau's Account, 300 Hana's Account, 300 Tia's Account, 300
Accounts (name)	Another pre-existing data, that is in the file, already set for the three children's account's names	Nikau's Account Hana's Account Tia's Account
Balance	Preset data, set by the parents for the children's annual clothing allowance of 300.	300 300 300

Module Structure:

The code will be separated into modules. I will write the definition for each function that is in my code (as stated above). So, there will be one for selecting the file in the folder and splitting the lines, one for saving this information into a file, another for turning them into accounts with certain balances, one for depositing money into the account, and one for withdrawing money from an account, one for selecting a certain action, etc. So, the program will allow the user to select a file, open a test file, and make changes to this file (withdrawals/deposits or no changes at all) all of which will be displayed on a screen using GUI and allowing for these changes to be saved into the file and accessed through an easily accessible 'clothing allowance app'.

Program and GUI Structure and Modules:

• How will your program show the required information in the GUI and receive the user input?

The program will show all the necessary information in the GUI app. This includes a welcome message giving a brief explanation of the account at the top, the current balance, and the names of each of the accounts, all in the top frame (Account Details), in the bottom frame there will be two combo boxes allowing the user to choose the account they'd like to use and whether they'd like to make a deposit or a withdrawal, underneath which will be an entry label with the value they'd like to deposit or withdraw into/from the account, and finally the submit label to carry out the transaction. Upon pressing the submit button a feedback/error message will be displayed stating whether the transaction went through, and from which account and the amount, or an error message stating the value submitted is not a valid amount/there are insufficient funds in the account to make the transaction. As well as the feedback/error message is displayed, a bonus message will also be displayed stating whether the user is on track to receive a bonus at the end of the year or needs to save some more money in their account in order to get one.

• How will you store the data? – What variables will you require and what type of data will your variables store (e.g. text, numeric, Boolean)?

All data will be stored in the file ('clothing_allowance_app_file.txt'), within this file will be the three accounts of the children ('Nikau's Account' 'Hana's Account' and 'Tia's Account' strings) and the balances of each account (float). The app will allow the user to make withdrawals and deposits to the user's respective account balance. Everything else within the app will be pre-set meaning no input will be required from the user so no variables or datatypes will be required to be stored.

• How will you structure your program? – What procedural structure will your program require? – Will you create functions/procedures to improve the structure, flexibility, and robustness of your program? – What parameters and/or return values would be required?

The program will be split into multiple different definitions (as well as definitions embedded within the Class), these definitions will all work interchangeably and simultaneously to carry out each of their respective parts such as reading in the file, creating the list of accounts, checking the balances and printing statements based off these, etc. Most of the definitions

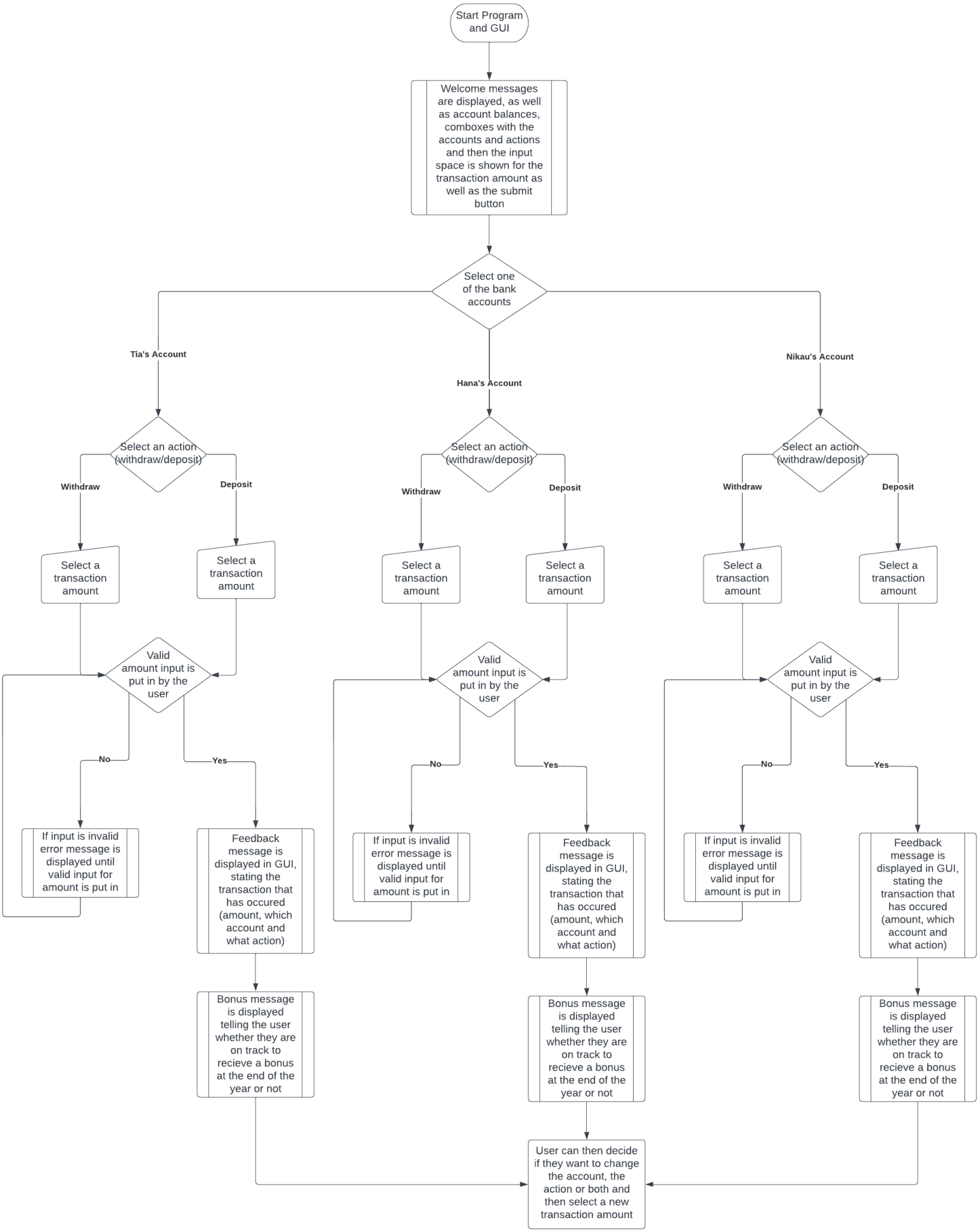
will either be run through the manage_action definition, with a couple of them such as creat_names_list and get_data being run outside of this definition, as well as the classes and the definitions within it being called into use with the GUI and other definitions. The imports will be kept at the top of the program, followed by the Classes and their sub-definitions, then the functions and setup, and finally the GUI code itself. The GUI code will contain all the code for all the labels, frames, buttons, combobox, messages, entry labels and messages that show in the app. The code being separated into these three sections will prevent overlap with the code and will make going over the code a lot easier and making changes to the code as well. This will also mean if other people need to check the code or go over it, it will be easy for them to see what is being done with the help of comments. The only variable that will have to be returned will be the name_list so it can be accessed outside of particular definitions and allocated to account_names (which is used in the GUI code to set up the values in the accounts combobox.

• How will you validate input and give feedback to the user? – What methods will you use to restrict and/or validate input?
The user will constantly be given feedback and messages to help direct them. When the user opens the app, they will be welcomed and given a quick message about the app's purpose and functionality, as well as this they will be shown the balances of the three children's accounts. They will then be able to select the account they wish to make the transaction from and what action they would like to take (deposit or withdraw). Upon selecting the account, they will be shown the account they've selected and the action they have chosen displayed in the combobox. They will then be asked to submit an amount for the transaction to carry out. If they do not select an appropriate value (i.e. they leave it blank or use anything that isn't a positive number greater than0) an error message will be shown asking the user to input an appropriate number/positive number. If the user wishes to make a withdrawal and attempts to withdraw more money than there is in the account, they will be told that they have insufficient funds in their account to make that transaction. Before finally being given a feedback message when an appropriate transaction has occurred stating the amount that has been transacted and from which account, as well as a message stating whether the user is illegible for a bonus.

The methods set in place to ensure valid inputs are that largely the user is given options to select rather than inputting themselves preventing them from putting invalid responses, furthermore, the user is provided a lot of messages to direct them and feedback messages depending on the actions they've done. Finally, with regards to the amount input, the program is made in such a way that any amount value that isn't a positive number will not be accepted and when the user is making a withdrawal amounts greater than the balance will not be accepted. This is set up in the code so that these values will not be accepted, and the user will be prompted to submit appropriate values.

• When will the program display output to the user?
The program will constantly display output to the user as soon as the program is initiated it displays the GUI, which contains the balances and account names. As the user interacts with the GUI, the app responds by providing options for the user to select and display messages and feedback/error messages depending on the inputs of the user. As amounts are deposited or withdrawn feedback messages will be displayed reiterating to the user what's happened and automatically updating the balance to the account and saving it to the file.

Flow Chart of the Program:


Start Program and GUI
Welcome messages are displayed, as well as account balances, comboxes with the accounts and actions and then the input space is shown for the transaction amount as well as the submit button
Select one of the bank accounts
Tia's Account
Hana's Account
Nikau's Account
Select an action (withdraw/deposit)
Withdraw
Deposit
Select a transaction amount
Valid amount input is put in by the user
No
Yes
If input is invalid error message is displayed until valid input for amount is put in
Feedback message is displayed in GUI, stating the transaction that has occured (amount, which account and what action)
Bonus message is displayed telling the user whether they are on track to recieve a bonus at the end of the year or not
User can then decide if they want to change the account, the action or both and then select a new transaction amount

Expected and Unexpected Input:

This will be used in reference to the code to see if I've got the planning steps and each condition correctly placed and ensuring that the expected outcome occurs in the actual code.

Possible Input (for Amount) Deposit/withdrawals for all three accounts	**Expected Outcome**
0	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
-1	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
-10	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
1.45	Amount is accepted transaction occurs, message recounting the transaction is displayed in the GUI and new bank balance is saved to the file and shown in the GUI
0.01	Amount is accepted transaction occurs, message recounting the transaction is displayed in the GUI and new bank balance is saved to the file and shown in the GUI
0.6794	Amount is accepted transaction occurs, message recounting the transaction is displayed in the GUI and new bank balance is saved to the file and shown in the GUI
10	Amount is accepted transaction occurs, message recounting the transaction is displayed in the GUI and new bank balance is saved to the file and shown in the GUI
194589	Amount isn't accepted transaction doesn't occur as the account doesn't have sufficient funds, error message is displayed in the GUI.
1000000	Amount isn't accepted transaction doesn't occur as the account doesn't have sufficient funds, error message is displayed in the GUI.
sdfh	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
DFDGD	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
One hundred	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
1 hundred	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
@#$%^&*()	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
Ao13	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
90g7	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)

12^&*	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
Asfk%&*	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)
12hsdf%^&*	Error message is displayed, value is not accepted, user is asked to input an appropriate value (positive number)

Testing for the bonus message (Nikau's Account):

Possible Input	**Expected Outcome**
Account has less than $50.00	A bonus message is displayed stating the account has less than $50.00 in it, so the user should save a bit more money to receive their allocated bonus at the end of the year.
Account has more than $50.00	A bonus message is displayed stating the account has more than $50.00 in it, so the user will be able to receive their allocated bonus as long as they keep at least $50.00 in their account until the end of the year.

Testing for when the clothing allowance app file isn't already existing:

Possible Input	**Expected Outcome**
The file doesn't exist (isn't created in the folder)	New file is created by the programme, and set up under the name clothing_allowance_app_file with the appropriate account information and balances
The file already exists (created in the folder)	The programme begins to run, reads the file and the GUI is displayed with the accounts, balances, actions, and other messages.