REQUIREMENTS

Ok guys so this is going to be our requirements document to divide up and keep track of our tasks throughout the program. This is also due with the team sheets so make sure to keep this file updated…don't take it off the shared folder and if you copy it and make a change (i.e. adding a task or splitting a task into two) update it here so we have one consistent document with all of our changes. The Program has each person's responsibility listed under a section with the title.

FILE IO – Brian Olsen

• Allow the user to select an existing checkbook (file) or to create a new one.

• Use the filename suffix ".cbk" for the checkbook file.
at the start or after a transaction is complete.

• A completed transaction must be recorded to the checkbook file in a comma-delimited format. Each transaction must appear on its own line. The format for each transaction in the file is as follows:

- CASH:
  [*date*],CASH,[*amount*],CASH,[*memo*],[*new balance*]
- Check:
  [*date*],[*check #*],[*amount*],[*to field*],[*memo*],[*new balance*]
- DEPOSIT:
  [*date*],DEPOSIT,[*amount*],DEPOSIT,[*memo*],[*new balance*]

• Allow the user to select a box to view the completed check. If the box is checked, upon completion of a check transaction, display a completed check. Include check number, the "to" field, memo, amount, etc. – everything from the last assignment.

• You may use one additional file for program-specific data.

• Suggestion: Validate your results by importing your checkbook file into Excel and verifying correct entries and a correct balance.

J Unit Testing/ Documentation – Daren Hoots

• Implement J Unit Testing and documentation

All transactions should use the current date from the system.
• When the user selects to create a Check transaction, auto-increment the check number from the last used number. Check numbers for new checkbooks should start with 1001.

• Data must persist through multiple openings and closing of the program. In other words, the user must be capable of closing the program, starting it again, creating another check transaction, and having the program provide the next valid check number and provide a correct balance based upon all transactions in the checkbook.

GUI - Bryan Allen

• If a new or empty checkbook is selected, prompt the user to enter a beginning balance.
• If an existing checkbook is selected, always display the current balance
• Allow the user to enter three types of transactions – CASH, Check, and
DEPOSIT. A CASH transaction deducts money from the account with no associated check number. A Check deducts money from the account with an associated check number. Deposit adds money to the account.
• All transactions should allow the user to enter a memo.

• Prior to completing a transaction, provide a warning to the user if any transaction would create a negative balance. Allow the user to cancel this transaction and do not record it if it is cancelled.

• Allow the user to repeatedly enter multiple transactions without having to exit and restart the program. (Technically GUI but keep in Logic)

• Accept all required user input via graphical user interface.

• Provide all user feedback/output via graphical user interface.

General Requirments for everyone

• Using the PSP forms, estimate the effort *prior* to beginning your design or implementation and then carefully denote the time spent on the project so that you can compare to your estimates.

• Provide a class diagram of all Java classes that you created

  ○ You must **include** class relationships

• Provide a requirements list with your estimate Product Planning Sheet. The requirements List must also specify all functional (behaviors the program must have) and non-functional (project requirements that are not program behaviors) requirements that you perceive that are **not** explicitly stated in this assignment document. Be sure to separate the functional and non-functional requirements.