ESC Week 10
Name: Parwani Jayati
Student ID: 1005622

Unit Tests

Use Case 1- Receive CSV files
Input space- file upload

Tests in code:

Rationale:
It is important to test out which files format are uploaded by the user.
Invalid formats or empty or no files upload should fail the test case, while only a valid csv should pass the test case.

Equivalence Class – Partition	Tests function in code	Input Value-Type	Expected Outcome
No file uploaded (only 1 file uploaded)	`nofileType()`	Boundary Value	Invalid
Wrong format file uploaded (pdf, png etc)	`fileInvalidType()`	Middle Value	Invalid
File doesn't exist in path	`invalid_arguments()`	Boundary Value	Invalid (throws exception)
Empty csv file	`emptyfileType()`	Boundary Value	Valid
2 csv files	`csvValid()`	Middle Value	Valid
2 csv files but file name has an extra space such as: `"homework/sample_file_ 3.csv"`	`incorrect_arguments()()`	Boundary Value	Valid

Use case 2- Validate both csv files
Input Space: 5 column headers in a csv: [Customer ID#, Account No., Currency, Type, Balance, Transaction Date]
Rationale:
Content of the Csv files must be validated before csv comparison, so equivalence class input space has to be the valid header of the csv files uploaded as given above in blue.

Equivalence Class – Partition	Tests function in code	Input Value-Type	Expected Outcome
Csv file with missing column headers	`missing_headers()`	Boundary Value [Customer ID#, Account No]	Invalid

Csv file with 5 columns but wrong column headers	`allincorrect_headers()`	Middle Value [Customer ID#, Account No., Currency, Type, Date]	Invalid
CSV with 1 different column header than expected	`invalid_headers()`	Boundary Value [Customer ID#, Account Type., Currency, Type, Balance, Transaction Date]	Invalid
Csv file with 5 column headers as expected in order: Customer ID#, Account No., Currency, Type, Balance	`valid_headers()`	Middle Value [Customer ID#, Account No., Currency, Type, Balance]	Valid
Csv file with 5 correct column headers in a different order	`jumbled_headers()`	Boundary Value [Customer ID#, Account No., Currency, Balance, Type]	Invalid

System Testing

Use Case 3- Compare csv files row by row
Input Space: **2 corresponding array lists , one from each csv**

Rationale:
Row by row comparison is the main aim of the code and hence, its partitioning involves various cases which can be simulated. Exception represent the mismatch, while match cases are those rows are identical and must even account for jumbled order.

Equivalence Class – Partition	Tests function in code	Input Value-Type	Expected Outcome
some value in the array lists is **different** such as the customer id	`writemiddlefiletest()`	Middle Value Csv1: [**ID1**, BOS963211, USD, SAVINGS, 962510] Csv2: [**ID2**, BOS963211, USD, SAVINGS, 962510]	Mismatch-Exception -
All rows are different in both arrays	`writedoublefiletest()`	Boundary Value Csv1: [ID1, BOS9630233, USD, CURRENT, 932510]	Mismatch-Exception – output.csv has a

		Csv2: [ID2, BOS963211, SGD, SAVINGS, 962510]	length of the sum of both input csv-2 (subtract 2 to account for header)
All values are exactly same and so is their order	`writeemptyfiletest()`	Middle Value Csv1: [**ID1**, BOS963211, USD, SAVINGS, 962510] Csv2: [**ID1**, BOS963211, USD, SAVINGS, 962510]	Match – output.csv file is empty