Finance Tracker

Requirements and Test Document

Pablo Bandera Lopez

04/04/2025

CS 225, Spring 25

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

This document will detail the various requirements and test cases needed to have a fully functional application. This program will be a personal monthly finance tracker where the user will be able to create entries, edit entries throughout the month and view all entries that have been entered. Further details can be found in the Software Design Document.

The requirements and test cases that follow break down the application into testable units. Some requirements and test cases are designed to test program reliability and adequate function of the back-end code. Other requirements and test cases are designed to test the user interface and its ability to interact with the back-end program. Both of these types of test cases requirements are necessary for the application to function.

BACKGROUND INFORMATION:

As previously stated, further information about the specific categories, sub-categories, accounts, types, etc. Is available in the software design document. Refer to tables 1-6 to see the list of available selections and when they should become available.

REQUIREMENTS:

Table 1: Requirement Specifications

ID	Requirement Specification
1	As a tester I want to be able to create an instance of Entry and set all its required information: dollarAmount, type, category, account, comment.
	1.1: A new instance can only be created when a valid entryID, entryID>0, is passed in the constructor. 1.2: A new instance of Entry shall be initialized by setting all its values to –1 and amount to 0.0. 1.3: The dollar amount, type, category, subcategory1-4, account, attributes shall only be settable to a value greater than 0 after being initialized. An InvalidEntryException shall be thrown if attempted.
2	As a tester I want to be able to create and instance of Entries to then add and remove entries from.
	2.1: A new Entry may only be added if the entry is valid. 2.2: An Entry shall be able to be removed based on and entryNumber. 2.3: An Entry shall be able to be found by entryNumber.
3	As a tester want to be able to Create a new instance of EntryFrame with or without an Entry.

	3.1: The program shall only display selection menus for dollarAmount, type, account, an area for a comment, and the submit or cancel buttons if instantiated with a null entry. 3.2: The program shall load the Entry data if instantiated and display all the previously selected and entered values for the user to change. 3.3: The program shall add the Entry when the submit button is pressed only if the Entry contains valid information.
4	As a programmer I want to be able to read and write from a file containing data in the following format int,double,int,int,int,int,int,int,int,String.
	4.1: The program shall name a new file Month.txt and it should be located in the directory ./Files/YYYY/Month.txt where YYYY is the current numerical year, and Month is the current month's name. 4.2: The program shall be able to print a series of entries onto the file. 4.3: The program shall be able to read the entries in a file, parse them into Entry objects and return an Entries object with all the file Entry objects in it.
5	As a user I want to be able to read the data from the file in a textual, decoded way.
	5.1: The program can take a String read from a file and decode it into a textual representation. 5.2: The program will determine what account was charged and represent it in a text version. 5.3: The program will only take the relevant information from an Entry.
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put the user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>
User story ID	<Put user story here, verbatim from the ATS, Table 1.>
	<Put the requirement here, with a requirement ID.>

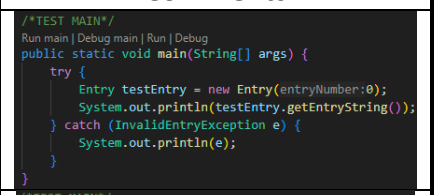
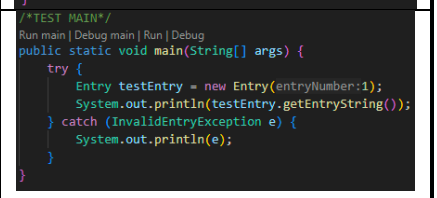
TEST CASES:

Table 1: Test Case Summary

User Story ID	Requirement ID	Test Case ID	Date	Status Pass/Fail/Pending
1	1.1	1.1	03.28.25	PASS
1	1.2	1.2	03.28.25	PASS
1	1.3	1.3	03.28.25	PASS
2	2.1	2.1	03.28.25	PASS
2	2.2	2.2	03.28.25	PASS

2	2.3	2.3	03.28.25	PASS
3	3.1	3.1	03.28.25	PASS
3	3.2	3.2	03.28.25	PASS
3	3.3	3.3	03.28.25	PASS
4	4.1	4.1	03.31.25	PASS
4	4.2	4.2	03.31.25	PASS
4	4.3	4.3	03.31.25	PASS
5	5.1	5.1	03.31.25	PASS
5	5.2	5.2	03.31.25	PASS
5	5.3	5.3	03.31.25	PASS

Table 2: Test Case Template and Results

Test Case ID: 1.1	**Current Status: PASS**		**Date:** 03.28.25
Req. ID: 1.1 A new instance can only be created when a valid entryID, entryID>0, is passed in the constructor.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 0.	Program should throw an InvalidEntryException with message: Invalid Entry Number.	
2	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1.	Program should print the new entry instance in the format: id,0.0,-1,-1,-1,-1,-1,-1,-1,-1, This format represents an empty entry.	

Screenshots:

```
159     /*TEST MAIN*/
        Run main | Debug main | Run | Debug
160     public static void main(String[] args) {
161         try {
162             Entry testEntry = new Entry(entryNumber:0);
163             System.out.println(testEntry.getEntryString());
164         } catch (InvalidEntryException e) {
165             System.out.println(e);
166         }
167     }
168
169 }
170
171
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
InvalidEntryException: Invalid Entry Number
```

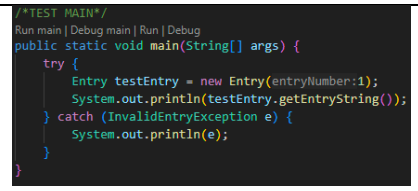
```
159     /*TEST MAIN*/
        Run main | Debug main | Run | Debug
160     public static void main(String[] args) {
161         try {
162             Entry testEntry = new Entry(entryNumber:1);
163             System.out.println(testEntry.getEntryString());
164         } catch (InvalidEntryException e) {
165             System.out.println(e);
166         }
167     }
168
169 }
170
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
1,0.0,-1,-1,-1,-1,-1,-1,-1,
```

Test Case ID: 1.2	**Current Status: pending**	**Date:** 03.28.25

Req. ID: A new instance of Entry shall be initialized by setting all its values to –1 and amount to 0.0.

Step #	Operator Action	Expected Results	Comments
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int > 0.	Program should print the new entry instance in the format: id,0.0,-1,-1,-1,-1,-1,-1,-1, This format represents an empty entry.	

```
/*TEST MAIN*/
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    try {
        Entry testEntry = new Entry(entryNumber:1);
        System.out.println(testEntry.getEntryString());
    } catch (InvalidEntryException e) {
        System.out.println(e);
    }
}
```

Screenshots:

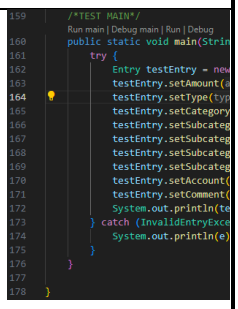
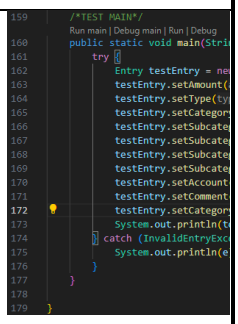
```
159     /*TEST MAIN*/
        Run main | Debug main | Run | Debug
160     public static void main(String[] args) {
161         try {
162             Entry testEntry = new Entry(entryNumber:1);
163             System.out.println(testEntry.getEntryString());
164         } catch (InvalidEntryException e) {
165             System.out.println(e);
166         }
167     }
168
169 }
170
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
1,0.0,-1,-1,-1,-1,-1,-1,-1,
```

Test Case ID: 1.3	**Current Status: PASS**	**Date:** 03.28.25

Req. ID: 1.3 The dollar amount, type, category, subcategory1-4, account, attributes shall only be settable to a value greater than 0 after being initialized. An InvalidEntryException shall be thrown if attempted.

Step #	**Operator Action**	**Expected Results**	**Comments**
1	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1. Use the setter methods to set all parameters to a value greater than 0 or non-empty string.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,accout,comment	
2	In Entry Class /*Test Main*/ instantiate a new Entry object and pass it int 1. Use the setter methods to set all parameters to a value greater than 0 or non-empty string. Using one of the setter	Program should throw and InvalidEntryException and print the corresponding error message. Example: InvalidEntryException: No Category Selected	

	methods set a value of 0 for any of the elements. Example: testEntry.setCategory(0);		

Screenshots:

```java
/*TEST MAIN*/
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    try {
        Entry testEntry = new Entry(entryNumber:1);
        testEntry.setAmount(amount:12.22);
        testEntry.setType(type:2);
        testEntry.setCategory(category:2);
        testEntry.setSubcategory(subcategory:1);
        testEntry.setSubcategory2(subcategory2:2);
        testEntry.setSubcategory3(subcategory3:1);
        testEntry.setSubcategory4(subcategory4:3);
        testEntry.setAccount(account:2);
        testEntry.setComment(comment:"Test Comment");
        System.out.println(testEntry.getEntryString());
    } catch (InvalidEntryException e) {
        System.out.println(e);
    }
}

}
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

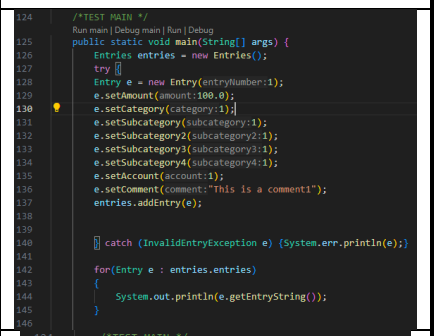
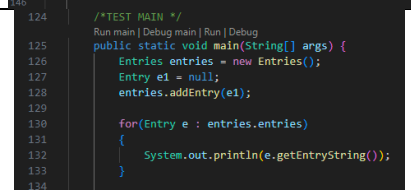
```
1,12.22,2,2,1,2,1,3,2,Test Comment
```

```java
/*TEST MAIN*/
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    try {
        Entry testEntry = new Entry(entryNumber:1);
        testEntry.setAmount(amount:12.22);
        testEntry.setType(type:2);
        testEntry.setCategory(category:2);
        testEntry.setSubcategory(subcategory:1);
        testEntry.setSubcategory2(subcategory2:2);
        testEntry.setSubcategory3(subcategory3:1);
        testEntry.setSubcategory4(subcategory4:3);
        testEntry.setAccount(account:2);
        testEntry.setComment(comment:"Test Comment");
        testEntry.setCategory(category:0);
        System.out.println(testEntry.getEntryString());
    } catch (InvalidEntryException e) {
        System.out.println(e);
    }
}

}
```

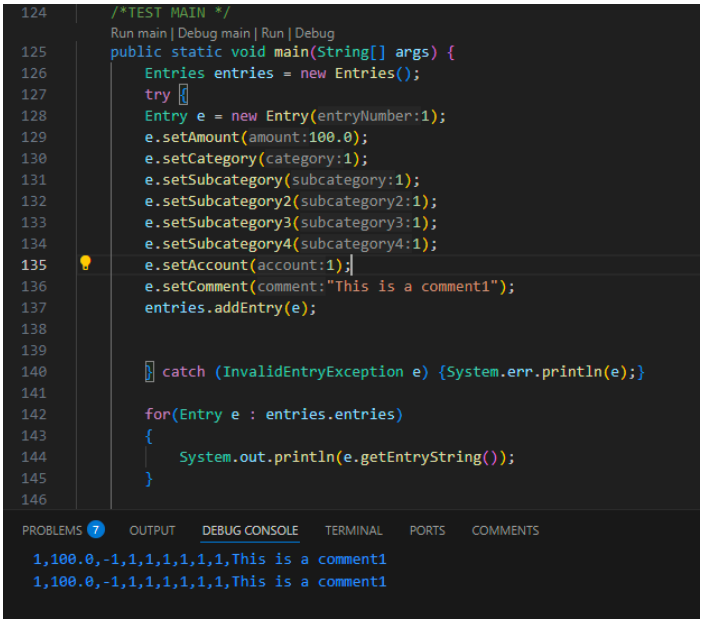
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
InvalidEntryException: No Category Selected
```

Test Case ID: 2.1	**Current Status: PASS**	**Date:** 03.28.25
Req. ID: 2.1 A new Entry may only be added if the entry is valid. Valid is not null.		

Step #	**Operator Action**	**Expected Results**	**Comments**
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	The Entry object e1 should be added to the entries attribute.	
2	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set it equal to null.	The Entry object e1 should be rejected from the entries.	

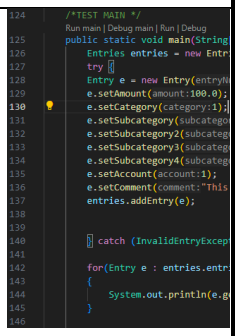
Screenshots:



```
124     /*TEST MAIN */
        Run main | Debug main | Run | Debug
125     public static void main(String[] args) {
126         Entries entries = new Entries();
127         try {
128         Entry e = new Entry(entryNumber:1);
129         e.setAmount(amount:100.0);
130         e.setCategory(category:1);
131         e.setSubcategory(subcategory:1);
132         e.setSubcategory2(subcategory2:1);
133         e.setSubcategory3(subcategory3:1);
134         e.setSubcategory4(subcategory4:1);
135         e.setAccount(account:1);
136         e.setComment(comment:"This is a comment1");
137         entries.addEntry(e);
138
139
140         } catch (InvalidEntryException e) {System.err.println(e);}
141
142         for(Entry e : entries.entries)
143         {
144             System.out.println(e.getEntryString());
145         }
146
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
1,100.0,-1,1,1,1,1,1,1,This is a comment1
1,100.0,-1,1,1,1,1,1,1,This is a comment1
```

Test Case ID: 2.2	**Current Status: pending**		**Date:** 03.28.25
Req. ID: 2.2 An Entry shall be able to be removed based on and entryNumber.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,accout,comment	
2	Use the removeEntryByNumber() function to remove the entry by ID.	Program should print the rest of the entries in the array in this case nothing should be printed.	
Screenshots:			

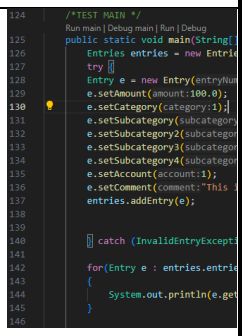
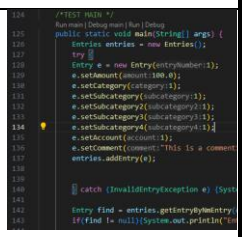
```java
/*TEST MAIN */
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    Entries entries = new Entries();
    try {
    Entry e = new Entry(entryNumber:1);
    e.setAmount(amount:100.0);
    e.setCategory(category:1);
    e.setSubcategory(subcategory:1);
    e.setSubcategory2(subcategory2:1);
    e.setSubcategory3(subcategory3:1);
    e.setSubcategory4(subcategory4:1);
    e.setAccount(account:1);
    e.setComment(comment:"This is a comment1");
    entries.addEntry(e);


    } catch (InvalidEntryException e) {System.err.println(e);}

    for(Entry e : entries.entries)
    {
        System.out.println(e.getEntryString());
    }

    entries.removeEntry(entryNumber:1);
    System.out.println(x:"After removing entry 1:");
    for(Entry e : entries.entries)
    {
        System.out.println(e.getEntryString());
    }

    //Entry e2 = new Entry(2);
    // e2.setAmount(200.0);
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
1,100.0,-1,1,1,1,1,1,1,1,This is a comment1
1,100.0,-1,1,1,1,1,1,1,1,This is a comment1
After removing entry 1:
```

Test Case ID: 2.3	**Current Status: PASS**	**Date:** 03.28.25

Req. ID: An Entry shall be able to be found by entryNumber.

Step #	**Operator Action**	**Expected Results**	**Comments**
1	In Entries Class /*TEST MAIN*/ instantiate a new Entries object. Instantiate a new Entry object: e1 and set its data. Add e1 to entries.	Program should print the new entry instance in the format: id,dollarAmount,type,category,subcategory,subcategory2,subcategory3,subcategory4,accout,comment	
2	Use the getEntryByNmEntry function with parameter 1.	Program should return the entry and print it.	

Screenshots:

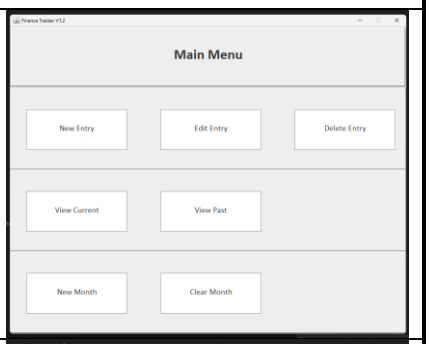
```java
/*TEST MAIN */
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    Entries entries = new Entries();
    try {
    Entry e = new Entry(entryNumber:1);
    e.setAmount(amount:100.0);
    e.setCategory(category:1);
    e.setSubcategory(subcategory:1);
    e.setSubcategory2(subcategory2:1);
    e.setSubcategory3(subcategory3:1);
    e.setSubcategory4(subcategory4:1);
    e.setAccount(account:1);
    e.setComment(comment:"This is a comment1");
    entries.addEntry(e);


    } catch (InvalidEntryException e) {System.err.println(e);}

    Entry find = entries.getEntryByNmEntry(entryNumber:1);
    if(find != null){System.out.println("Entry Found: "+find.getEntryString());}

    // for(Entry e : entries.entries)
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

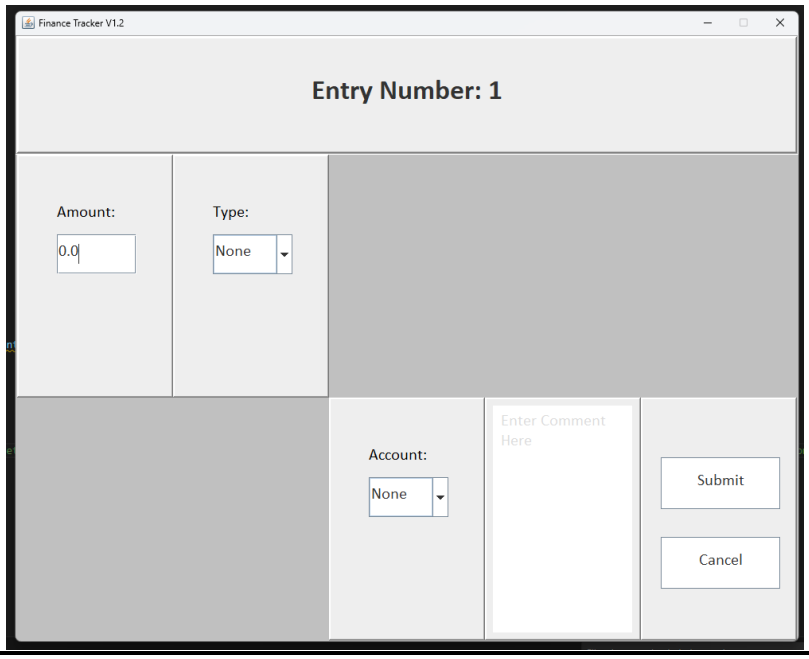
```
1,100.0,-1,1,1,1,1,1,1,1,This is a comment1
Entry Found: 1,100.0,-1,1,1,1,1,1,1,1,This is a comment1
```

Test Case ID: 3.1	**Current Status: PASS**	**Date:** 03.28.25

Req. ID: 3.1 The program shall only display selection menus for dollarAmount, type, account, an area for a comment, and the submit or cancel buttons if instantiated with a null entry.

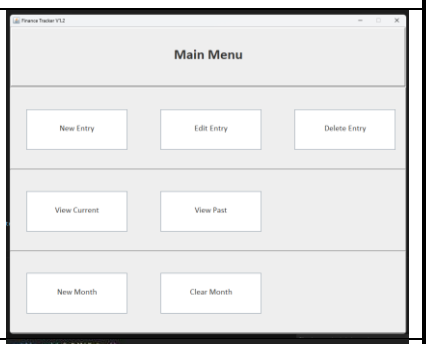
Step #	Operator Action	Expected Results	Comments
1	Run the main method of FinanceTracker.	The Main Menu Should Display.	
2	Select New Entry.	A new instance of Entry Frame should be created with a null entry passed in the constructor.	

Screenshots:

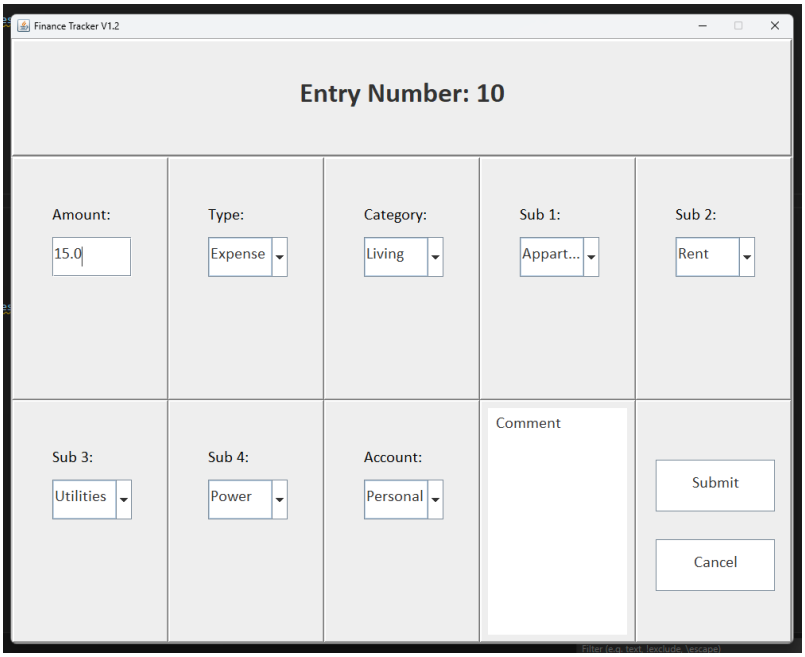


Test Case ID: 3.2	**Current Status: Pass**	**Date:** 03.28.25

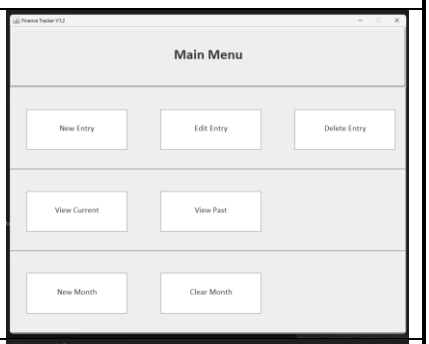
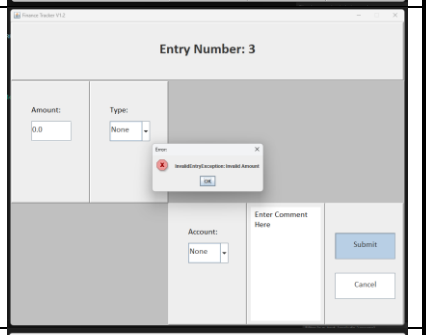
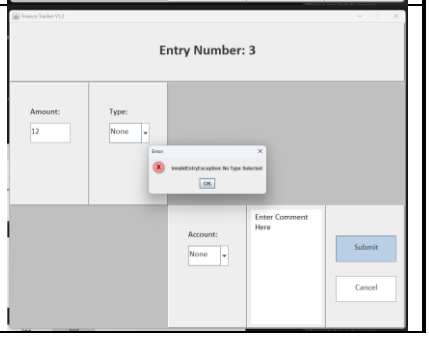
Req. ID: 3.2 The program shall load the Entry data if instantiated and display all the previously selected and entered values for the user to change.

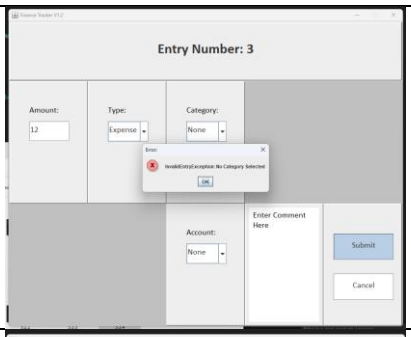
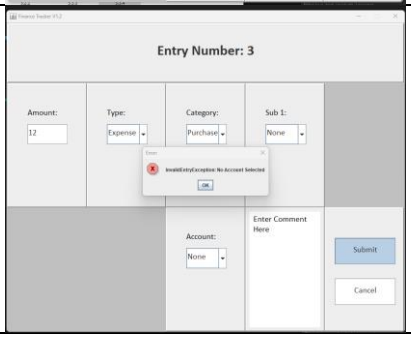
Step #	Operator Action	Expected Results	Comments
1	Run the main method of FinanceTracker.	The Main Menu Should Display.	
2	Select Edit Entry.	A new instance of Entry Frame should be created with a fabricated entry passed in the constructor.	

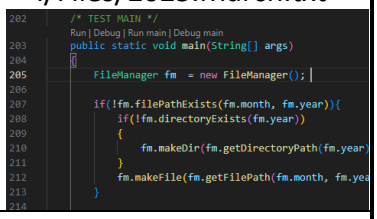
Screenshots:



Test Case ID: 3.3	**Current Status: pending**	**Date:** 03.28.25
Req. ID: 3.3 The program shall add the Entry when the submit button is pressed only if the Entry contains valid information.		

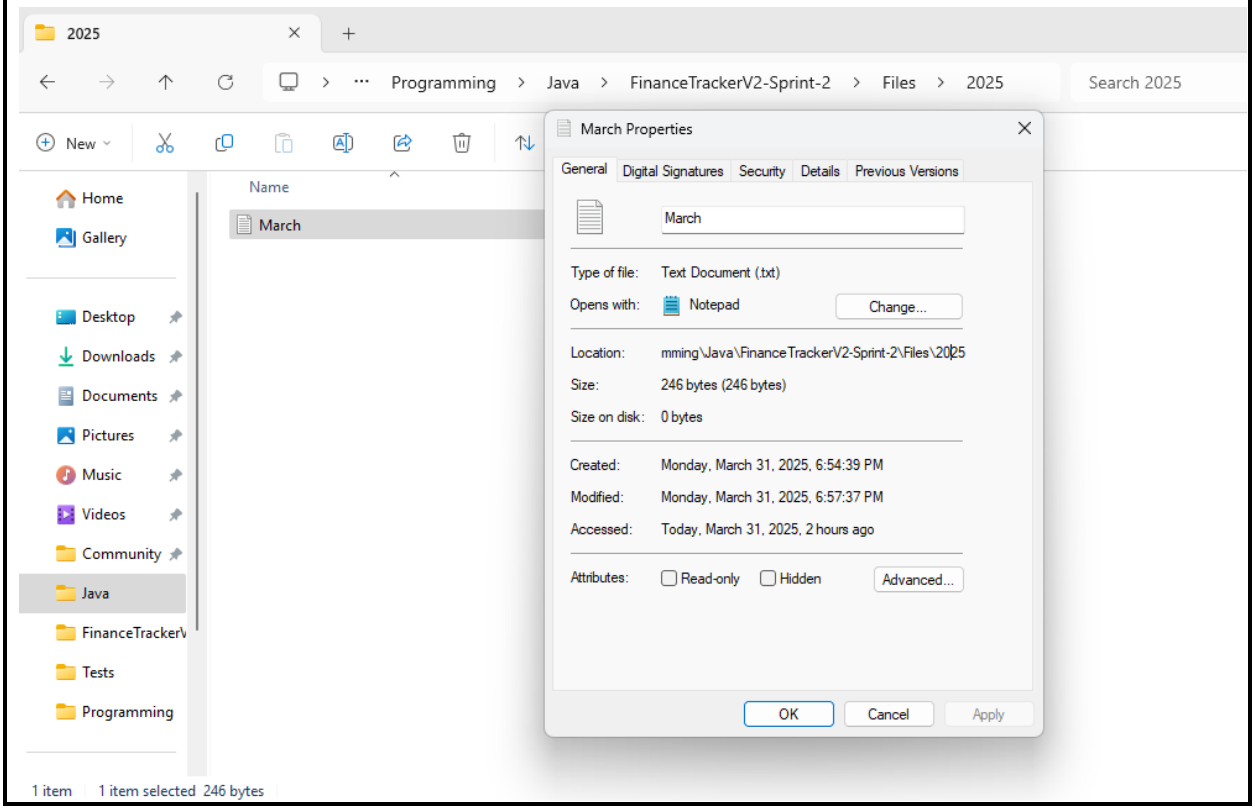
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Run the main method of FinanceTracker.	The Main Menu Should Display.	
2	Select New Entry.	A new instance of Entry Frame should be created with a null entry passed in the constructor.	
3	Click Submit	An Error message should appear, invalid amount.	
4	Enter Amount > 0 and click Submit.	An Error message should appear, invalid type.	

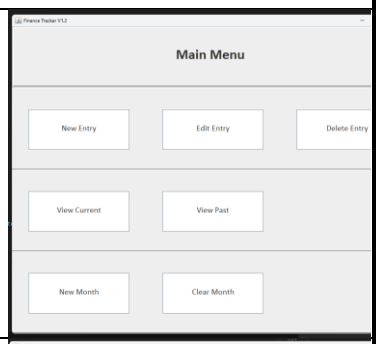
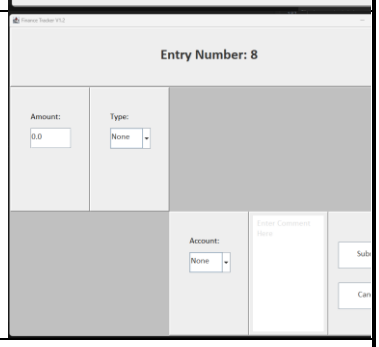
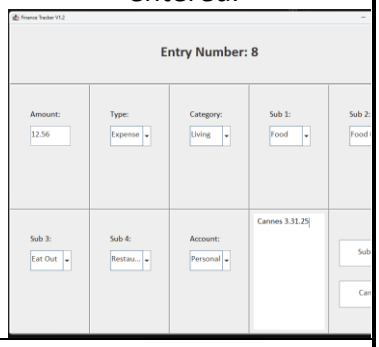
5	Select type expense and click submit.	An Error message should appear, invalid category.	
6	Select category personal and click submit.	An Error message should appear, no account selected.	
7	Select Account personal and click submit.	The frame should close and return to the main menu.	
Screenshots:			

Test Case ID: 4.1	Current Status: PASS		Date: 03.31.25
Req. ID: 4.1 The program shall name a new file Month.txt and it should be located in the directory ./Files/YYYY/Month.txt where YYYY is the current numerical year, and Month is the current month's name.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	In the FileManager class's /*TEST MAIN*/ create a new instance of FileManager.	A new instance shall be created.	FileManager fm = new FileManager();
2	Using the makeDir and makeFile functions create a directory and file with the required name and location.	A new Directory shall be created named after the numerical year and inside of it contains a text file named after the current month.	LOCATION: ./Files/YYYY/Month.txt Example: ./Files/2025.March.txt 

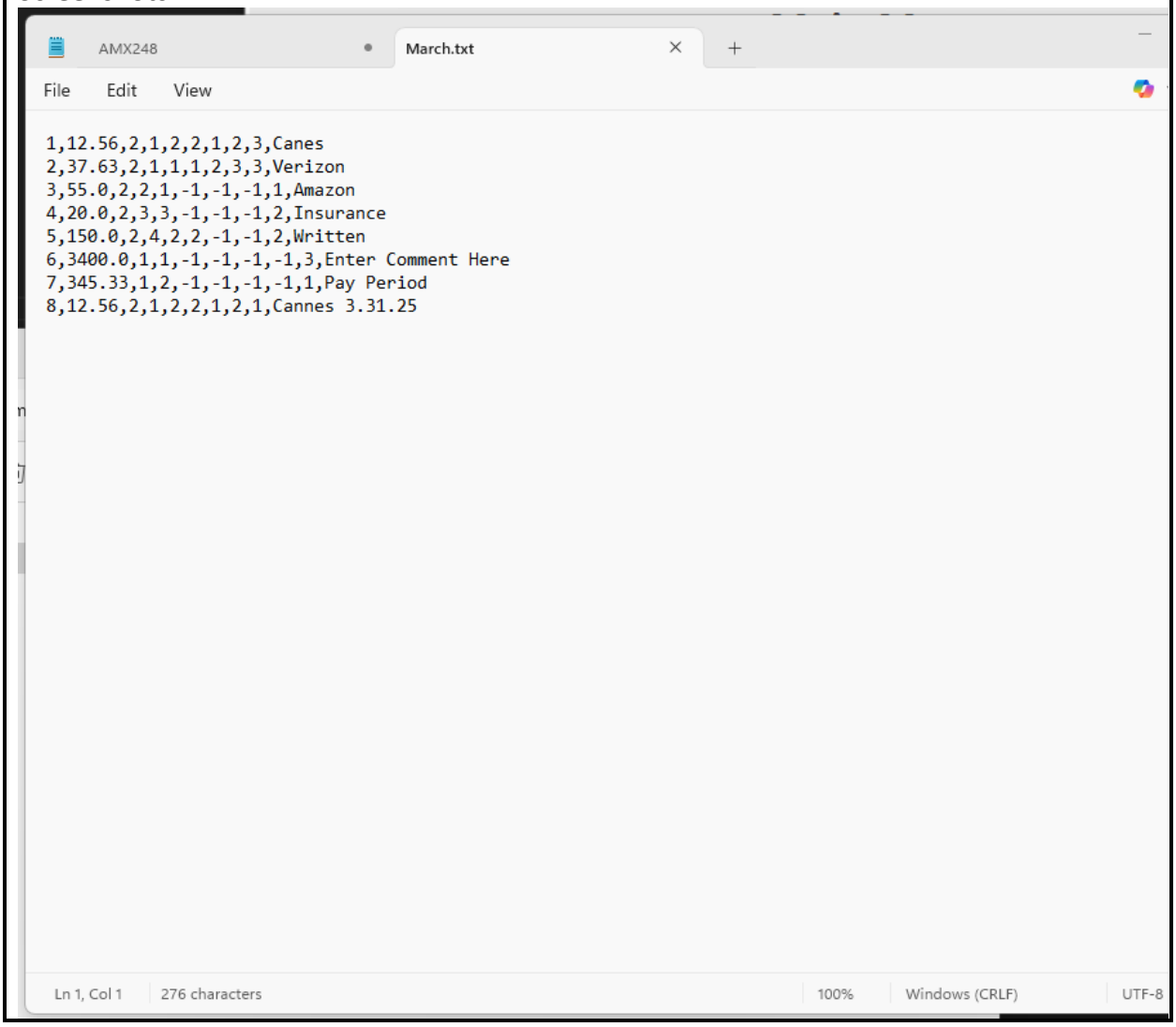
```
/* TEST MAIN */
Run | Debug | Run main | Debug main
public static void main(String[] args)
{
    FileManager fm  = new FileManager();

    if(!fm.filePathExists(fm.month, fm.year)){
        if(!fm.directoryExists(fm.year))
        {
            fm.makeDir(fm.getDirectoryPath(fm.year
        }
        fm.makeFile(fm.getFilePath(fm.month, fm.yea
    }
```

Screenshots:



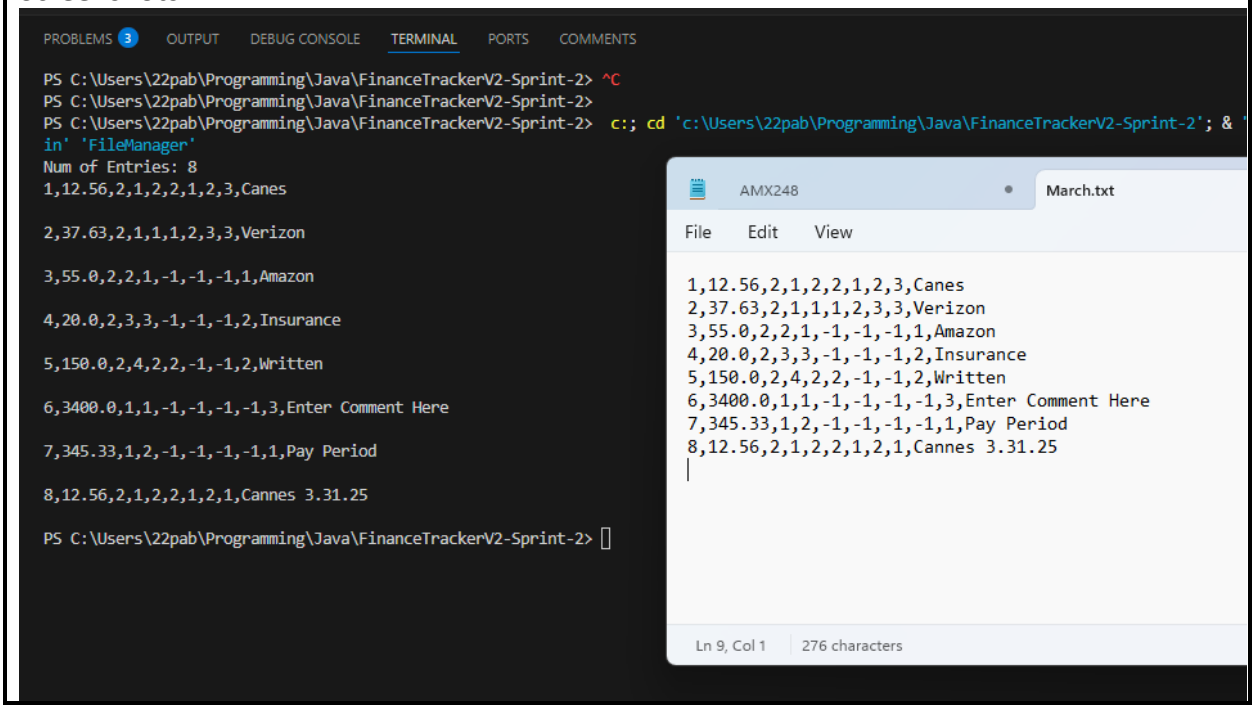
Test Case ID: 4.2	**Current Status: PASS**		**Date: 03.31.25**
Req. ID: 4.2 The program shall be able to print a series of entries onto the file.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Run the main funciton in FinanceTracker.java.	A new main menu frame should be visible.	
2	Select New Entry.	A new empty entry Frame should be visible.	
3	Fill out a new Entry with valid data and Click Submit.	A new Entry object should be created and added to the Entries ArrayList.	Valid data includes: amount > 0, type selected, category selected, a selection made for any subcategories that are visible, an account selected, a comment entered. 
4	Using File Explorer or equivalent, navigate to ./Files/2025/March.txt	Any Entry objects stored in the Entries object ArrayList should be printed in a semi-encoded string inside the file.	The numbers represent the selected index from the respective JComboBox.

Screenshots:



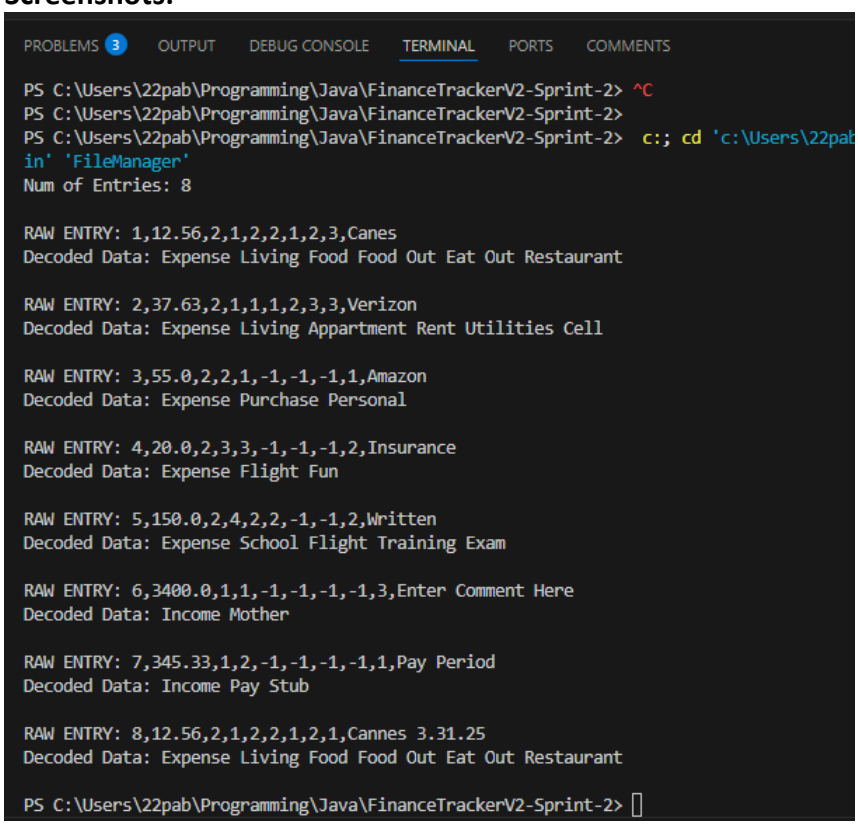
Test Case ID: 4.3	Current Status: PENDING		Date: 03.21.25
Req. ID: 4.3 The program shall be able to read the entries in a file, parse them into Entry objects and return an Entries object with all the file Entry objects in it.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	In the FileManager /*TEST MAIN*/ function: Create a new instance of File Manager.	A new instance of FileManager shoud be created.	
2	Create an new instance of Entries and utilize the readEntries function in combination with the FILE PATH functions to read from the file created int the previous test case 4.2.	All the entries stored in the file should be read into the the Entries object.	
3	Create a new instance of ArraList<Entry> and initialize it with the value if the Entries object array list using the getEntries method. Iterate through the entries and print them to the console.	All the entries stored in the file should be printed onto the console.	

Screenshots:



Test Case ID: 5.1	Current Status: PASS		Date: 03.31.25
Req. ID: 5.1 The program can take a String read from a file and decode it into a textual representation.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Using the /*TEST MAIN*/ found in FileManager.java, create a new instance of Decoder.	New null instance of decoder is created.	
2	Inside the for loop: initialize the new Decoder with the string of the current entry being read.	The current entry being read from the file, will be passed into Decoder.	
3	Us the getDecoded Method to print out the decoded equivalent of the data.	The console should display the decoded user selected data from the entry.	

Screenshots:



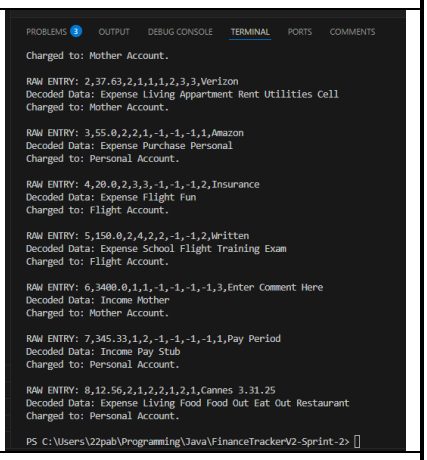
```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2> ^C
PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>
PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>  c:; cd 'c:\Users\22pab
in' 'FileManager'
Num of Entries: 8

RAW ENTRY: 1,12.56,2,1,2,2,1,2,3,Canes
Decoded Data: Expense Living Food Food Out Eat Out Restaurant

RAW ENTRY: 2,37.63,2,1,1,1,2,3,3,Verizon
Decoded Data: Expense Living Appartment Rent Utilities Cell

RAW ENTRY: 3,55.0,2,2,1,-1,-1,-1,1,Amazon
Decoded Data: Expense Purchase Personal

RAW ENTRY: 4,20.0,2,3,3,-1,-1,-1,2,Insurance
Decoded Data: Expense Flight Fun

RAW ENTRY: 5,150.0,2,4,2,2,-1,-1,2,Written
Decoded Data: Expense School Flight Training Exam

RAW ENTRY: 6,3400.0,1,1,-1,-1,-1,-1,3,Enter Comment Here
Decoded Data: Income Mother

RAW ENTRY: 7,345.33,1,2,-1,-1,-1,-1,1,Pay Period
Decoded Data: Income Pay Stub

RAW ENTRY: 8,12.56,2,1,2,2,1,2,1,Cannes 3.31.25
Decoded Data: Expense Living Food Food Out Eat Out Restaurant

PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>
```

Test Case ID: 5.2	**Current Status: PASS**		**Date: 03.31.25**
Req. ID: 5.2 The program will determine what account was charged and represent it in a text version.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Using the /*TEST MAIN*/ found in FileManager.java, create a new instance of Decoder.	New null instance of decoder is created.	
2	Inside the for loop: initialize the new Decoder with the string of the current entry being read.	The current entry being read from the file, will be passed into Decoder.	
3	Use the decodeAccount method in the while loop to print out what account the entry was charged to.	The console should print what account the entries were charged to.	
Screenshots:			

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
Charged to: Mother Account.

RAW ENTRY: 2,37.63,2,1,1,1,2,3,3,Verizon
Decoded Data: Expense Living Appartment Rent Utilities Cell
Charged to: Mother Account.

RAW ENTRY: 3,55.0,2,2,1,-1,-1,-1,1,Amazon
Decoded Data: Expense Purchase Personal
Charged to: Personal Account.

RAW ENTRY: 4,20.0,2,3,3,-1,-1,-1,2,Insurance
Decoded Data: Expense Flight Fun
Charged to: Flight Account.

RAW ENTRY: 5,150.0,2,4,2,2,-1,-1,2,Written
Decoded Data: Expense School Flight Training Exam
Charged to: Flight Account.

RAW ENTRY: 6,3400.0,1,1,-1,-1,-1,-1,3,Enter Comment Here
Decoded Data: Income Mother
Charged to: Mother Account.

RAW ENTRY: 7,345.33,1,2,-1,-1,-1,-1,1,Pay Period
Decoded Data: Income Pay Stub
Charged to: Personal Account.

RAW ENTRY: 8,12.56,2,1,2,2,1,2,1,Cannes 3.31.25
Decoded Data: Expense Living Food Food Out Eat Out Restaurant
Charged to: Personal Account.

PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>
```

Test Case ID: 5.3	**Current Status: PASS**	**Date: 03.31.25**
Req. ID: 5.3 The program will only take the relevant information from an Entry.		

Step #	**Operator Action**	**Expected Results**	**Comments**
1	Run the /* Test Main*/ funciton in FileManager.java	A series of entries should be printed to the console.	
2	Verify printed values are those that are relevant.	When the relevant information is printed, the data from Type to SubCat4 should be converted into text form if not a –1. If a –1 is encountered nothing should be printed for that value.	The format of a Raw Entry is as follows: Entry Number, Ammount, Type, Category, SubCat1, SubCat2, SubCat3, SubCat4, Account, Comment If a piece of data is not relevant, it should be marked by a –1.

Screenshots:

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

Charged to: Mother Account.

RAW ENTRY: 2,37.63,2,1,1,1,2,3,3,Verizon
Decoded Data: Expense Living Appartment Rent Utilities Cell
Charged to: Mother Account.

RAW ENTRY: 3,55.0,2,2,1,-1,-1,-1,1,Amazon
Decoded Data: Expense Purchase Personal
Charged to: Personal Account.

RAW ENTRY: 4,20.0,2,3,3,-1,-1,-1,2,Insurance
Decoded Data: Expense Flight Fun
Charged to: Flight Account.

RAW ENTRY: 5,150.0,2,4,2,2,-1,-1,2,Written
Decoded Data: Expense School Flight Training Exam
Charged to: Flight Account.

RAW ENTRY: 6,3400.0,1,1,-1,-1,-1,-1,3,Enter Comment Here
Decoded Data: Income Mother
Charged to: Mother Account.

RAW ENTRY: 7,345.33,1,2,-1,-1,-1,-1,1,Pay Period
Decoded Data: Income Pay Stub
Charged to: Personal Account.

RAW ENTRY: 8,12.56,2,1,2,2,1,2,1,Cannes 3.31.25
Decoded Data: Expense Living Food Food Out Eat Out Restaurant
Charged to: Personal Account.

PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>
```

```
PROBLEMS 3    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

Charged to: Mother Account.

RAW ENTRY: 2,37.63,2,1,1,1,2,3,3,Verizon
Decoded Data: Expense Living Appartment Rent Utilities Cell
Charged to: Mother Account.

RAW ENTRY: 3,55.0,2,2,1,-1,-1,-1,1,Amazon
Decoded Data: Expense Purchase Personal
Charged to: Personal Account.

RAW ENTRY: 4,20.0,2,3,3,-1,-1,-1,2,Insurance
Decoded Data: Expense Flight Fun
Charged to: Flight Account.

RAW ENTRY: 5,150.0,2,4,2,2,-1,-1,2,Written
Decoded Data: Expense School Flight Training Exam
Charged to: Flight Account.

RAW ENTRY: 6,3400.0,1,1,-1,-1,-1,-1,3,Enter Comment Here
Decoded Data: Income Mother
Charged to: Mother Account.

RAW ENTRY: 7,345.33,1,2,-1,-1,-1,-1,1,Pay Period
Decoded Data: Income Pay Stub
Charged to: Personal Account.

RAW ENTRY: 8,12.56,2,1,2,2,1,2,1,Cannes 3.31.25
Decoded Data: Expense Living Food Food Out Eat Out Restaurant
Charged to: Personal Account.

PS C:\Users\22pab\Programming\Java\FinanceTrackerV2-Sprint-2>
```

Test Case ID: 5.3	Current Status: PENDING		Date: 03.31.25
Req. ID:			
Step #	**Operator Action**	**Expected Results**	**Comments**
1			
2			Any Printed
N			
Screenshots:			

*[Shall be **completed for user stories actively worked, and completed, in the current sprint**.]*

REFERENCES:

Oracle. (n.d.). *Trail: Creating a GUI with swing*. Trail: Creating a GUI With Swing (The JavaTM Tutorials). https://docs.oracle.com/javase/tutorial/uiswing/index.html

APPENDICES: