***Test Case Specification***

***For***

***Team Coffee House***

***April 2, 2019***

******

Prepared by:

David Clendenning

& Jon Lewis

***Table of Contents:***

***1* INTRODUCTION**……………………………………………………4

***2* TEST CASES: INVENTORY SYSTEM**…………………………4

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1 | 4/2/2019 | David Clendenning | Initial Document |

**1 INTRODUCTION**

This document provides the test cases to be carried out for the DTCC Toner Inventory System. Each module of the system is to be represented by an individual test case (excluding access methods). Each case details the input and expected outputs.

**2 Test Cases: Toner Inventory System**

|  |  |
| --- | --- |
| Test ID | 3.1.1 |
| Title | Read Item With Incorrect Size Of File |
| Feature | Reads .csv file into itemList array for an instance of the data access object |
| Objective | Confirms that the .csv file is unable to be read |
| Setup | Access to JDK and java virtual machine environment to run application  Computer must have file “printersTestOutOfRange.csv” |
| Test Data | - Csv file = “printersTestOutOfRange.csv”  - Data access object |
| Test Actions | 1. create data access object  2. store “printersTestOutOfRange.csv” with itemCSV method in global variable itemFilePath within data access object  3. confirm that the NumberFormatException occurs |
| Expected Results | NumberFormatException should be thrown and test case passes |

|  |  |
| --- | --- |
| Test ID | 3.1.2 |
| Title | Read Item NonCSV |
| Feature | Reads .csv file into itemList array for an instance of the data access object |
| Objective | Confirms that the .csv file is unable to be read |
| Setup | Access to JDK and java virtual machine environment to run application  Computer must have file “Wilmington Toner Database(1).xlsx” |
| Test Data | - Csv file = “Wilmington Toner Database(1).xlsx”  - Data access object |
| Test Actions | 1.create data access object  2. store itemCSV in global variable itemFilePath within data access object  3. confirm that the ArrayIndexOutOfBoundsException occurs |
| Expected Results | ArrayIndexOutOfBoundsException should be thrown and test case passes |

|  |  |
| --- | --- |
| Test ID | 3.1.3 |
| Title | Read Items |
| Feature | Reads .csv file into itemList array for an instance of the data access object |
| Objective | Confirm that when readItemCSV is called the file is readable and is called on the data access object such that there are no errors or popup windows indicating a user input |
| Setup | Access to JDK and java virtual machine environment to run application  Computer must have file “Wilmington Toner Database.csv” |
| Test Data | - create data access object  - store Wilmington Toner Database.csv” with itemCSV method in global variable itemFilePath within data access object |
| Test Actions | 1.Calls function to store itemCSV in data access object  2.Calls function to read the stored itemCSV in data access object  3.Confirms that the input function was not called and no exceptions occured |
| Expected Results | The test case should not prompt for input nor throw an exception. Data access objects boolean needInput should evaluate to false. The test case should pass. |

|  |  |
| --- | --- |
| Test ID | 3.1.4 |
| Title | Read Printer Incorrect Size Of File |
| Feature | Reads .csv file into printerList array for an instance of the data access object |
| Objective | Confirms that the .csv file is not readable |
| Setup | Access to JDK and java virtual machine environment to run application  Terminal must contain the file, “printersTestOutOfRange.csv” |
| Test Data | - CSV file = “printersTestOutOfRange”  - Data access object printerList |
| Test Actions | 1. Create data access object  2. Store “printersTestOurOfRange.csv” into data access objects global variable printerFilePath  2. Confirm that ArrayIndexOutOfBoundsException is thrown when the readPrinterCSV method is called via data access object |
| Expected Results | Throws a ArrayIndexOutOfBoundsException and passes test case |

|  |  |
| --- | --- |
| Test ID | 3.1.5 |
| Title | Read Printer NonCSV |
| Feature | Reads .csv file into printerList array for an instance of the data access object |
| Objective | Confirm that the file is not readable |
| Setup | Access to JDK and java virtual machine environment to run application  Terminal must contain the file, “printers(1).xlsx” |
| Test Data | - XLSX file = “printers(1).xlsx”  - Data access object |
| Test Actions | 1. Create data access object  2. Store “printersTestOurOfRange.csv” into data access objects global variable printerFilePath  3. Confirm that NumberFormatException is thrown when the readPrinterCSV method is called via data access object |
| Expected Results | Throws a NumberFormatException and passes the test case |

|  |  |
| --- | --- |
| Test ID | 3.1.6 |
| Title | Read Printers |
| Feature | Reads .csv file into printerList array for an instance of the data access object |
| Objective | Confirm that when readPrinterCSV is called the file is readable and is called on the data access object such that there are no errors or popup windows indicating a user input |
| Setup | Access to JDK and java virtual machine environment to run application  Terminal must contain the file, “printers.csv” |
| Test Data | - Data access object  - CSV file = “printers.csv” |
| Test Actions | 1. Create data access object  2. Extract data from csv file  3. Confirm whether the user input function was called (storePrinterCSV()) such that user input is required |
| Expected Results | The input function should not have been called and the test case should not throw any exceptions. Data access objects boolean needInput should evaluate to false. Test case passes |

|  |  |
| --- | --- |
| Test ID | 3.3.1 |
| Title | Random Index Get Item |
| Feature | Return item from list via model number |
| Objective | Confirms that the data access objects getItem method is equal to the specified Item |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object  - testItem to be added to all indices of data access object  - Item with model number “c”  - Random int (range of 0 – itemList.length)  - Item – expectedResult |
| Test Actions | 1. create data access object  2. create empty item to be added to indices of itemList  3. creates random int within range(0 – itemList.length)  4. assigns itemList[random int] to the ‘expected result’ Item  5. compare the item ‘expected result’ to the return value of getItem called with parameter “c” (model number) |
| Expected Results | The ‘expected result’ item should be equal to the getItem function return value (getItem should return expectedResult |

|  |  |
| --- | --- |
| Test ID | 3.3.2 |
| Title | Test When Item Is Not In List |
| Feature | Return item from list via model number |
| Objective | Confirms that the data access objects getItem method is equal to null |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object (w/ itemList)  - getItem parameter – “c” |
| Test Actions | 1. create data access object  2. compare the null value to the method getItem(“c”) and return whether they are equal or not |
| Expected Results | The two arrays should not be equal and test case should pass |

|  |  |
| --- | --- |
| Test ID | 3.3.3 |
| Title | Get Printer |
| Feature | Return printer from data access objects printerList via asset tag |
| Objective | Confirm that the expected item is return from the getPrinter method |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object  - testPrinter -> empty printer object  - expectedResult (printer with asset tag 14321) |
| Test Actions | 1. Create data access object  2. Create random int value  3. Create testPrinter empty printer object  4. Fill the data access objects printerList with testPrinter at each index  5. Create expectedResult printer with asset tag of 14321  6. Compare expectedResult to result of getPrinter method with parameter of 14321 (integer asset tag) |
| Expected Results | Confirm that the printer placed in a random position in the printerList array is found by the method getPrinter() |

|  |  |
| --- | --- |
| Test ID | 3.3.4 |
| Title | Get Printer When List Is Empty |
| Feature | Return printer from data access objects printerList via asset tag |
| Objective | Confirms that the data access objects getPrinter method is equal to null |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object  - Data access object printerList |
| Test Actions | 1. Create data access object  2. Compare null value to method getPrinter() with parameter of (14321) and return whether they are equal or not |
| Expected Results | The method returns null because this printer with the specified asset tag is not in the data access objects printerList |

|  |  |
| --- | --- |
| Test ID | 3.4.1 |
| Title | Stock Deficit MinMax Equal |
| Feature | Returns if the specified Item has a deficit in stock |
| Objective | To confirm that the current stock minus the minimum stock is greater than zero. |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 0  - Current Stock = 0 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned boolean value matches the expected boolean value |
| Expected Results | The current stock and minimum stock are both zero which is not a deficit therefore the result should be false |

|  |  |
| --- | --- |
| Test ID | 3.4.2 |
| Title | Stock Deficit Current Stock Larger |
| Feature | Returns if the specified Item has a deficit in stock |
| Objective | To confirm that the current stock minus the minimum stock is greater than zero. |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 5  - Current Stock = 6 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned boolean value matches the expected boolean value |
| Expected Results | The current stock is 6 and the minimum stock is 5, therefore the return value should be false |

|  |  |
| --- | --- |
| Test ID | 3.4.3 |
| Title | Stock Deficit Min Stock Larger |
| Feature | Returns if the specified Item has a deficit in stock |
| Objective | To confirm that the current stock minus the minimum stock is greater than zero. |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 5  - Current Stock = 2 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned boolean value matches the expected boolean value |
| Expected Results | The current stock is 2 and minimum stock is 5 therefore the result returned should be true, because there is a deficit |

|  |  |
| --- | --- |
| Test ID | 3.4.4 |
| Title | Stock Deficit Value MinMax Equal |
| Feature | Returns the deficit of specified Item if the item has a deficit in stock |
| Objective | To return the deficit amount that would need to be ordered |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 0  - Current Stock = 0 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned number matches the expected number |
| Expected Results | The current stock and minimum stock are both zero which is not a deficit therefore the result should be 0 |

|  |  |
| --- | --- |
| Test ID | 3.4.5 |
| Title | Stock Deficit Value Minimum Stock Larger |
| Feature | Returns the deficit of specified Item if the item has a deficit in stock |
| Objective | To return the deficit amount that would need to be ordered |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 5  - Current Stock = 0 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned number matches the expected number |
| Expected Results | The current stock is 0 and the minimum stock is 5 so the method should return 5 |

|  |  |
| --- | --- |
| Test ID | 3.4.6 |
| Title | Stock Deficit Value Current Stock Larger |
| Feature | Returns the deficit of specified Item if the item has a deficit in stock |
| Objective | To return the deficit amount that would need to be ordered |
| Setup | An item that has a specified minimum and current stock |
| Test Data | - Stock Information  - Minimum Stock = 5  - Current Stock = 0 |
| Test Actions | 1. Executes hasDeficit() method  2. Confirms whether returned number matches the expected number |
| Expected Results | The current stock is 1000 and the minimum stock is 5 so the method should return 5 |

|  |  |
| --- | --- |
| Test ID | 3.5.1 |
| Title | Add Item |
| Feature | Adds an item to the data access objects itemList |
| Objective | Test if the item has been successfully added  Catches any unwarranted exceptions |
| Setup | Data access object is created and is able to be used  Fill an empty slot in the data access objects itemList array with an item |
| Test Data | - Empty data access object of the CSV\_DBIMP class  - Empty ItemImp() object (“test1”) of the type Item superclass. |
| Test Actions | 1. Execute addItem() method  2. Confirms that an item has been placed inside the array by confirming the value of the object “test1” within the data access objects itemList array |
| Expected Results | The item should be added and no exceptions thrown |

|  |  |
| --- | --- |
| Test ID | 3.5.2 |
| Title | Add Item When List Is Full |
| Feature | Catches the ArrayIndexOutOfBoundsException in the case that the data access objects itemList array is full. |
| Objective | Throw an exception if an item cannot be added to the array |
| Setup | -Data access object is created and is able to be used  -Fill the data access objects itemList array with null ItemImp objects until array is full |
| Test Data | - Empty data access object of the CSV\_DBIMP class  - Empty ItemImp() object to be added. |
| Test Actions | 1. Fills the data access objects itemList array with empty ItemImp objects  2. Execute addItem() method  3. Confirms that an attempt was made to add item to the data access objects itemList array |
| Expected Results | Throws ArrayIndexOutOfBoundsException and Item is not added |

|  |  |
| --- | --- |
| Test ID | 3.5.3 |
| Title | Delete Item To First Index |
| Feature | Removes item from itemList array |
| Objective | Remove the selected item via model number |
| Setup | Add an item (with model number “HX756”) |
| Test Data | - testItem -> Empty Item to be tested of type Item  - dao -> data access object of type CSV\_DBIMP  - String -> “HX756” (model number) |
| Test Actions | 1. Add testItem to the array using addItem() method  2. Delete model number “HX756” from the array using deleteItem(“HX756”) method  3. Compares value of null, and item removed at that index (first item in list) |
| Expected Results | The index where the item was removed from is replaced with a null value. |

|  |  |
| --- | --- |
| Test ID | 3.5.4 |
| Title | Delete Item When List Is Empty |
| Feature | Catches error when ItemList is empty |
| Objective | Throw an ArrayIndexOutOfBoundsException if there is no item to be removed |
| Setup | Create a data access object (by default this objects itemList are all null values or ‘empty’) |
| Test Data | - dao -> data access object with empty itemList  - String -> “failDueToEmptyList” (model number) |
| Test Actions | 1. Execute deleteItem() method with parameters of “failDueToEmptyList” |
| Expected Results | Throw an ArrayIndexOutOfBoundsException that states that an item cannot be removed due to an empty list. |

|  |  |
| --- | --- |
| Test ID | 3.5.5 |
| Title | Delete Item In Random Index |
| Feature | Delete item according to specified model number, regardless of the position of the object within the data access objects itemList array |
| Objective | Delete an item that has been found via deleteItem parameters |
| Setup | - create random integer of the range (0 – itemList.length)  - create data access object of CSV\_DBIMP class  - create testItem of Item supertype |
| Test Data | - Data access object of type CSV\_DBIMP  - testItem of type Item  - random int (range 0 – itemList.length)  - ItemImp with model number “c” |
| Test Actions | 1. Create a new data access object  2. Populate data access objects itemList with new empty ItemImp objects  3. Remove Item with model number “c” from list |
| Expected Results | The space where the item was removed is replaced with a null value |

|  |  |
| --- | --- |
| Test ID | 3.6.1 |
| Title | Get All Items |
| Feature | Return all items within itemList in a data access object |
| Objective | Confirms that the data access objects itemList has the same objects within a new ArrayList created during test |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object (w/ itemList)  - ArrayList populated with all items from data access objects |
| Test Actions | 1. create data access object  2. create arrayList & populate with objects from itemList  3. compare the arrayList to the method getAllItems and return whether they are equal or not |
| Expected Results | The two arrays should be equal and test case should pass |

|  |  |
| --- | --- |
| Test ID | 3.6.2 |
| Title | Get All Items With Wrong Data |
| Feature | Return all items within itemList in a data access object |
| Objective | Confirms that the data access objects itemList is not equal to the arrayList created for test |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object (and data access objects itemList)  - ArrayList populated with all items from data access objects  - Random int (range of 0 – itemList.length) |
| Test Actions | 1. create data access object  2. create arrayList & populate with objects from itemList  3. creates random int within range(0 – itemList.length)  4. removes items at random index from data access objects itemList  5. compare the arrayList to the method getAllItems and return whether they are equal or not |
| Expected Results | The two arrays should not be equal and test case should pass |

|  |  |
| --- | --- |
| Test ID | 3.6.3 |
| Title | Get All Printers |
| Feature | Return all printers within printerList in a data access object |
| Objective | Confirms that the data access objects itemList is equal to the arrayList created for test |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object  - ArrayList populated with all printers from data access objects  - Data access object(s) (printerList) |
| Test Actions | 1. create data access object  2. create arrayList & populate with objects from printerList  3. compare the arrayList to the method getAllPrinters and return whether they are equal or not |
| Expected Results | The two arrays should be equal and test case should pass |

|  |  |
| --- | --- |
| Test ID | 3.6.4 |
| Title | Get All Printers With Wrong Data |
| Feature | Return all items within printerList in a data access object |
| Objective | Confirms that the data access objects printerList is not equal to the arrayList created for test |
| Setup | Access to JDK and java virtual machine environment to run application |
| Test Data | - Data access object  - ArrayList populated with all printers from data access objects  - Data access objects printerList  - Random int (range of 0 – itemList.length) |
| Test Actions | 1. create data access object  2. create arrayList & populate with objects from printerList  3. creates random int within range(0 – printerList.length)  4. removes items at random index from data access objects itemList  5. compare the arrayList to the method getAllItems and return whether they are equal or not |
| Expected Results | The two arrays should not be equal and test case should pass |