Team: Zachary Hager, Charles Kimmel, & Matthew White

October 12, 2022

CMSC 495 (CLASS 6381)

Overview of Capstone Team Project. This program is designed to  
load and retrieve aircraft flight time data (i.e. sorties).

GROUP 4 FINAL

SORTIE LOGGER

Table of Contents

Overview2

Introduction2

Roles and responsibilities2

Software Management2

Project Plan (Initial)3

Development Platform3

Basic Functionality3

UML Diagram3

System Requirements & Specification4

Minimum Hardware System Requirements4

User's Guide5

Start-up using Source Code5

Using the program5

Test Plan and Results8

Test Criteria8

Test Table8

Results10

Design and Alternate designs13

Operations Diagram13

Development History14

Code Revision History14

Comprehensive Development14

Conclusions15

Final Thoughts15

Ways to Improve15

Overview

**Introduction**

This program is an application that will add and retrieve records of aircraft sortie times (i.e., record take-off and land times of individual aircraft). At the axiom flights will take a “line” which is annotated by a 3-digit number (ex. “Line 151”). An aircraft will be designated for that line and annotated by it’s one letter and 4-digit tail number (ex. Aircraft “A0113”). These lines will have “scheduled” take-off and land times that will be annotated by Julian date and 24-hour time (ex. August 31, 2022 09:00 P.M. = 22243 2100). Next, there will be a text box to fill the “actual” take-off and land time of each sortie as sorties in real life will rarely follow the schedule perfectly. Lastly, an exception will be thrown requesting a statement from the user if the actual times are +/- 30 minutes from the scheduled time.

**Roles and responsibilities**

**Hager: Full Stack Development**

-Front end development by setting up initial classes.

-Team communications

-On going back-end refinements

**Kimmel: Back End Development/ Document Control**

-*Project proposal/lead*

-Documentation

-User guide & Test Planning

-Back-end refinements

**White: Back End Development/ Database Management**

-Trouble shooting

-Programing sharing/archiving

-Data base management

-Back-end refinements

**Software Management**

* **GitHub**: Used to share program files, and documentation
* **Discord**: For communication and team meetings.
* **IntelliJ IDEA & Eclipse**: IDE’s used between team members.

Project Plan (Initial)

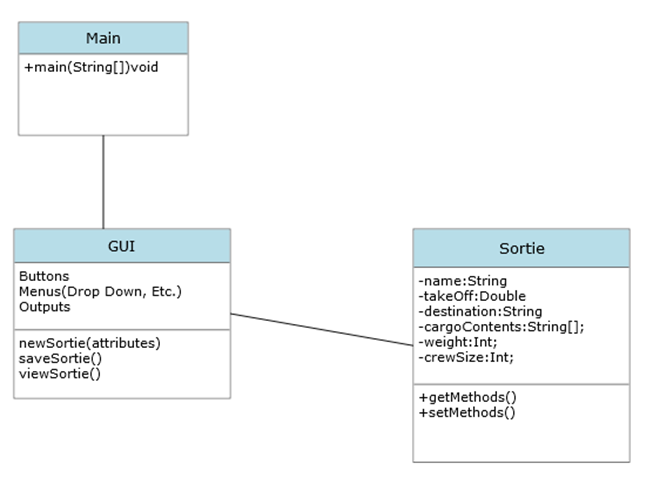
**Development Platform**

Our program was created with Java utilizing the Eclipse (Hager) and IntelliJ IDEA IDE’s (Kimmel & White) to write and edit the program. Each of us is utilizing the Java SE 8 JDK or newer to maximize compatibility, and because of its long-term support through 2030.

**Basic Functionality**

* GUI for user to input:
  + Aircraft- dropdown (Later Changed to text field)
  + Scheduled Time (text field)
  + Actual Takeoff/land time (text field)
  + Schedule Takeoff/land time (text field)
* Warning (pop up) for a time that is outside of 30 minutes from scheduled time
* Storage solution
  + Write and save a file
  + Write to a database
* Upgrades (in no particular order)-
  + Aesthetics
  + Database security
  + Cloud options

**UML Diagram**



System Requirements & Specification

**Minimum Hardware System Requirements**

* RAM: 256 MB
* Disk space: 256 MB Total: (124 MB for JRE; 2 MB for Java Update, 128 MB for program)
* Processor: Minimum Pentium 2 266 MHz processor
* Browsers: Internet Explorer 9 and above, Microsoft Edge, Firefox, Chrome

User’s Guide

**Start-up using Source Code**

Package: Sorties

Files (.java): Main, Window GUI, Sortie, DataHandler

1. Download Group4\_Final.zip folder.

2. Extract the Sorties folder.

3. (On Windows OS) right-click the extracted “Sorties” folder.

4. Click “Open Folder as [Preferred/Installed IDE] Project”

5. Once IDE opens, compile and run the package

6. Once GUI appears click any of the three tabs to interact with the program.

\*The displayed test cases were made using IntelliJ IDEA. Eclipse was also used in development. This Guide is not all encompassing. This is merely amongst the easiest ways to run our program across any Windows PC, and IDE.

**Using the Program**

**Starting UI:**

Program opens with three tabs and a close button:

|  |  |  |
| --- | --- | --- |
| Numeration | Tabs | Function |
| 1 | Add Sortie (Default open screen) | Transitions/opens screen that allows user to create flight record. |
| 2 | View Sortie | Transitions/opens screen that allows user to recover previously saved flight records |
| 3 | Finalize Times | Adds statement from user for time discrepancies. |

**Add Sortie UI:**

Screen that appears if “Add Sortie” is selected (also default open screen):

|  |  |  |
| --- | --- | --- |
| Type | Input | Function |
| Text Field | Create Line (int) | User must assign a 3-digit line number to the flight. |
| Text Field | Aircraft Tail Number  (string) | User must enter the tail number of the aircraft. |
| Text Field | Scheduled Take-off (int) | User must enter scheduled take-off time in Julian date and military time format: (YYDDD HHMM) |
| Text Field | Scheduled Land (int) | User must enter scheduled land time in Julian date and military time format: (YYDDD HHMM) |
| Text Field | Actual Take-off (int) | User must enter actual take-off time in Julian date and military time format: (YYDDD HHMM) |
| Text Field | Actual Land (int) | User must enter actual land time in Julian date and military time format: (YYDDD HHMM) |
| Conditional Text Field | Additional Information  (string) | User must enter a short explanation if the “Actual” times deviate 30 minutes over or under the scheduled time. |
| Button | Save Record | Transposes the information the user entered into a document and saves it to the program/database. |
| Button | Clear All Fields | Clears all text fields |

**View Sortie UI:**

Screen that appears if “View Sortie” is selected:

|  |  |  |
| --- | --- | --- |
| Type | Input | Function |
| Query (Text to Search) | Enter Line number to search flight data (int) | User enters the line number of the sortie they wish to view. |
| Drop Down (Select a Flight) | Displayed list of all flight documents | User can scroll through and select which flight record they wish to open. |
| Button | Search | Fills empty fields on screen with the flight data you have selected |

**Finalize Times UI:**

Screen that appears if “View Sortie” is selected:

|  |  |  |
| --- | --- | --- |
| Type | Input | Function |
| Drop Down (Select a Flight) | Displayed list of all flight documents | User enters the line number of the sortie they wish to view. |
| Text Field (Takeoff discrepancy) | Longer string | User types reason for time deviations over or under 30 minutes |
| Text Field (Land discrepancy) | Longer string | User types reason for time deviations over or under 30 minutes |

Test Plan and Results

**Test Criteria:**

***Pass:*** Component being tested met or exceed expectations

***Fail:*** Component being test in met not expectation, or undermined other parts of the program

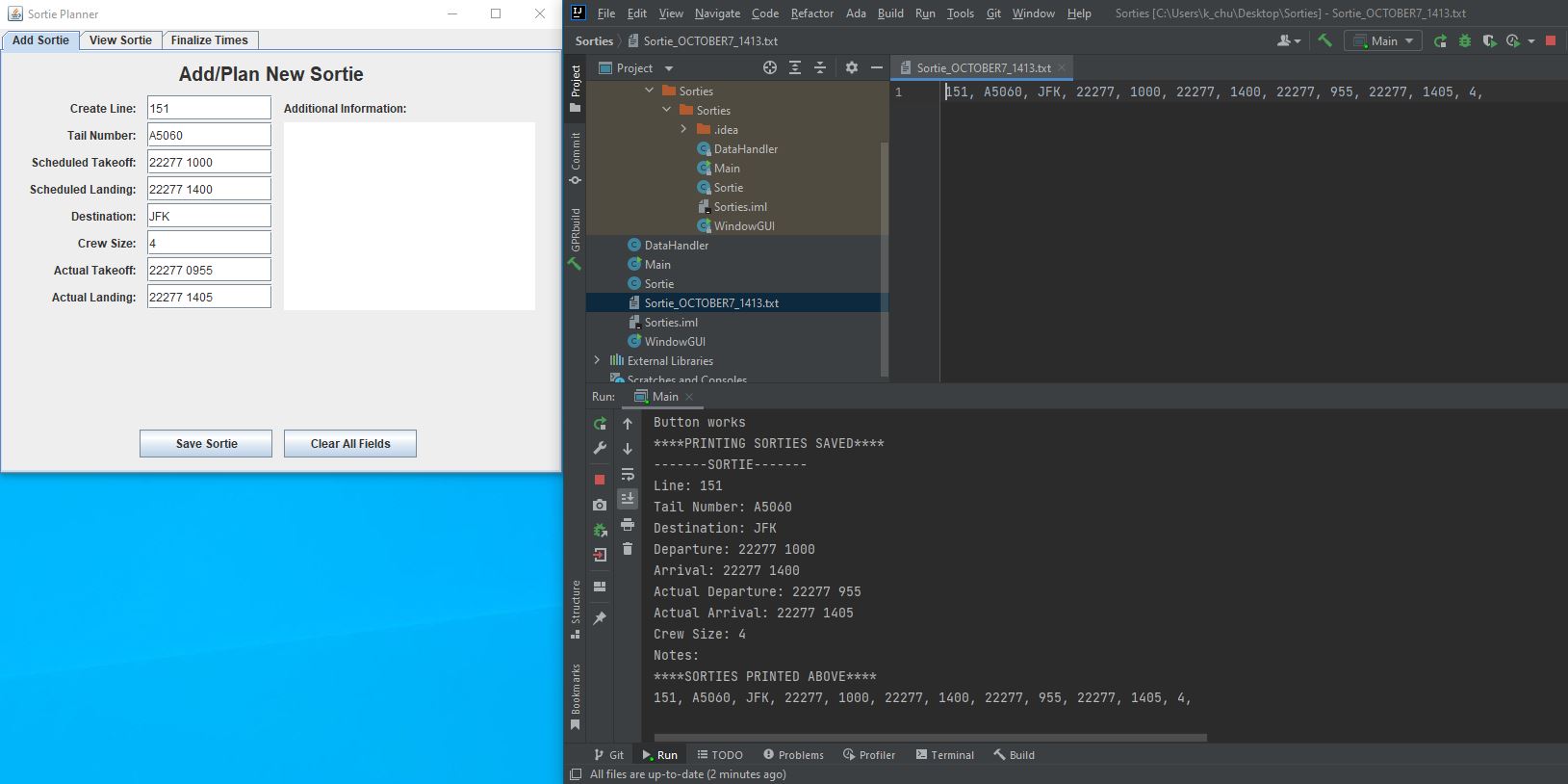
***Partial:*** Met some but not all requirements, and did undermine other parts of the program

**Test Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case # | Test Name | Test Description | Expected Result | Actual Result | Pass/Fail |
| 1 | Start Page | Program opens with two options to proceed to adding a sortie, or retrieving a record. | Test page is present with 2 options | Test page is present with 2 options | **PASS** |
| 2 | Add Sortie | After selecting the ‘Add’ button a page to fill the details of a sortie appear. | Page to add a sortie appears. | Page to add a sortie appears. | **PASS** |
| 3 | Create Line | Acting as the sortie’s axiom, the user gives the sortie a 3-digit designation | Program accepts 3-digit line number. | Program exclusively accepts 3-digit string. | **PASS** |
| 4 | Catching Line error | Makes the user reenter a line number if a character other than a number is entered | Prompt appears notifying the user of their error, user retries. | Prompt appears notifying the user of their error, user retries. | **PASS** |
| 5 | Tail Number | Drop down appears requesting the user select a preloaded aircraft tail number. | User selects tail number, and program proceeds. | Shifted to manual entry/ Program accepts Tail Number | **PASS** |
| 6 | Tail number error | Instructs user to select a tail number (aircraft) if the try to proceed without doing so. | Prompt appears notifying the user of their error, user retries. | Abandoned concept. Left it to user to enter proper tail number of aircraft. | **FAIL** |
| 7 | Scheduled take-off | User enters scheduled take-off date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts users’ entry. | **PASS** |
| 8 | Scheduled land time | User enters scheduled land date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts users’ entry. | **PASS** |
| 9 | Actual take-off time | User enters actual take-off date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts users’ entry. | **PASS** |
| 10 | Actual land time | User enters actual land date and time in Julian date notation and military time. (YYDDD HHMM) | Program accepts users’ entry. | Program accepts users’ entry. | **PASS** |
| 11 | Time deviation over 30 minutes | Prompts user to enter a short explanation on why the aircraft took off late. | User enter string, program accepts string | Prompted to enter string, program cannot store it. | **PARTIAL** |
| 12 | Time deviation under 30 minutes | Prompts user to enter a short explanation on why the aircraft took off early. | User enter string, program accepts string | Prompted to enter string, program cannot store it. | **PARTIAL** |
| 13 | Date or time annotation error | User enters the date or time in the wrong format, or with wrong characters. | Prompt appears notifying the user of their error, user retries. | Prompt appears notifying the user of their error, user retries. | **PASS** |
| 14 | Save added sortie | User saves all previous information entered. | Sortie data is saved to its own file in the program/data base. | Stores to a unique .txt file | **PASS** |
| 15 | Data check before saving | User erroneously attempts to save data while one of the boxes is not filled. | Prompt appears notifying the user of their error, user retries without clearing the other fields. | Prompt appears notifying the user of their error, user retries without clearing the other fields. | **PASS** |
| 16 | Retrieve Data | After selecting the ‘View Sortie’ button a page appears to retrieve sortie records. | Page appears to fill information to retrieve records. | Page appears to fill information to retrieve records. | **PASS** |
| 17 | Open archived data | User opens archived files. | User view’s record of previously added files. | Can only receive saved .txt files | **PARTIAL** |

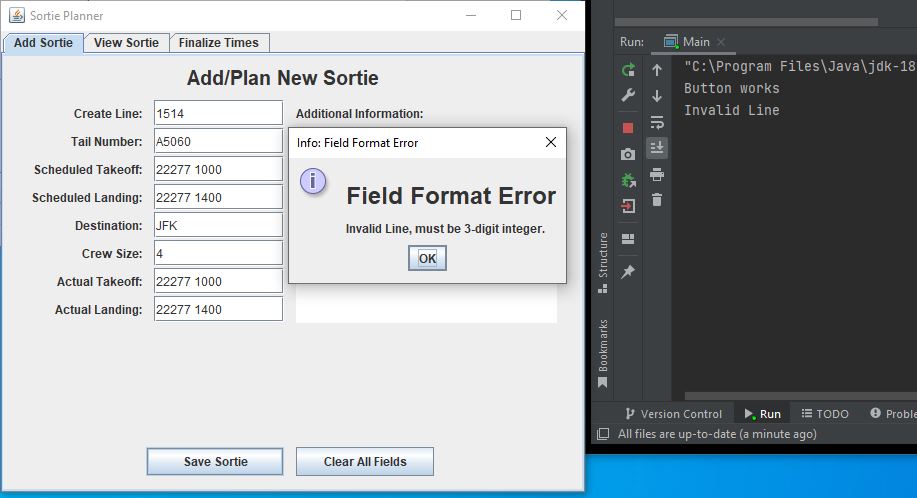
**Results:**

**Test Cases: 1, 2, 3, 5, 7, 8, 9, 10, 14, 17**

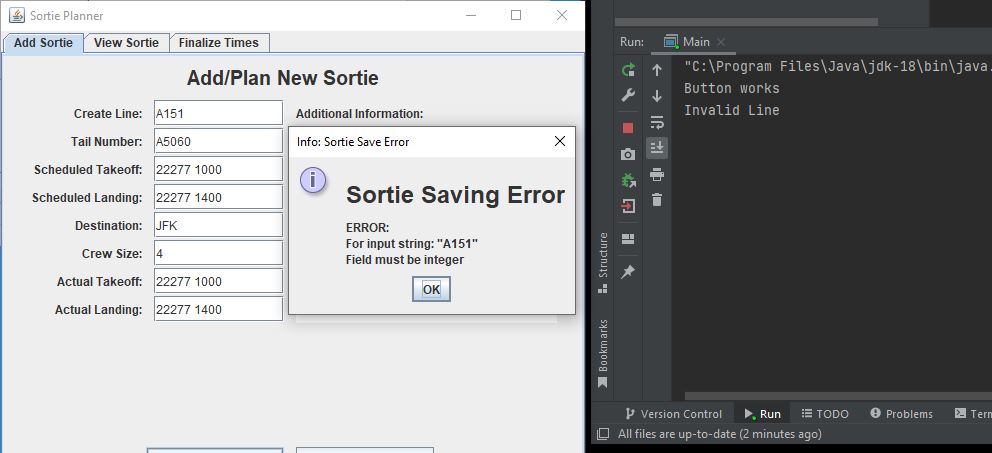


Program opens automatically to add sortie tab. Save and Clear function operate as expected and further more stores all data entered by the user as .txt file

**Test Cases: 4**

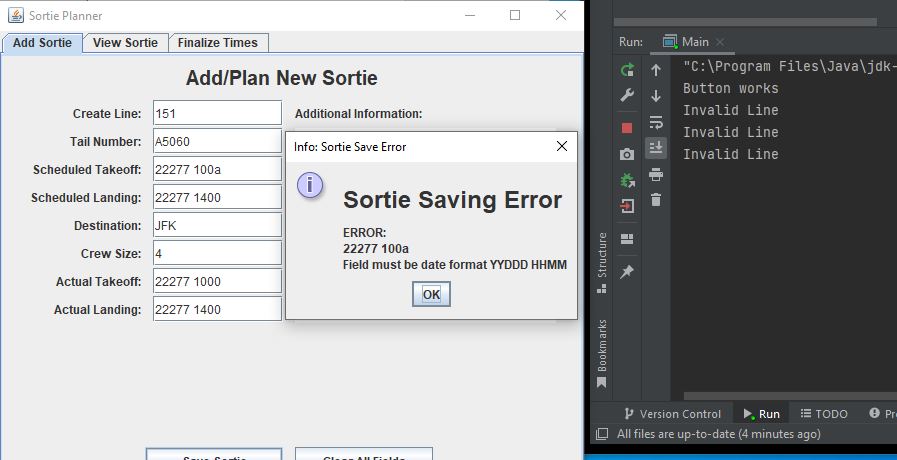


Catches Line entries over 3 digits

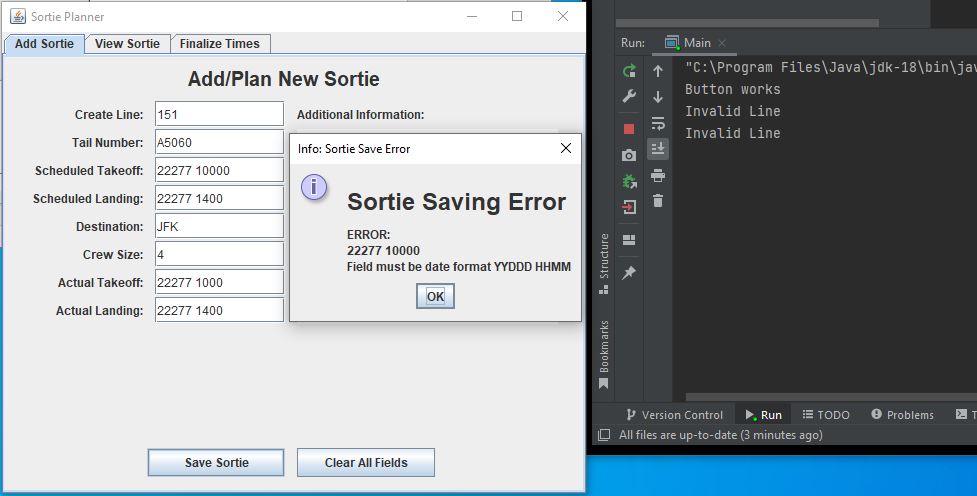


Catches Line if line has any charter other than an integer

**Test Cases: 13, 15**

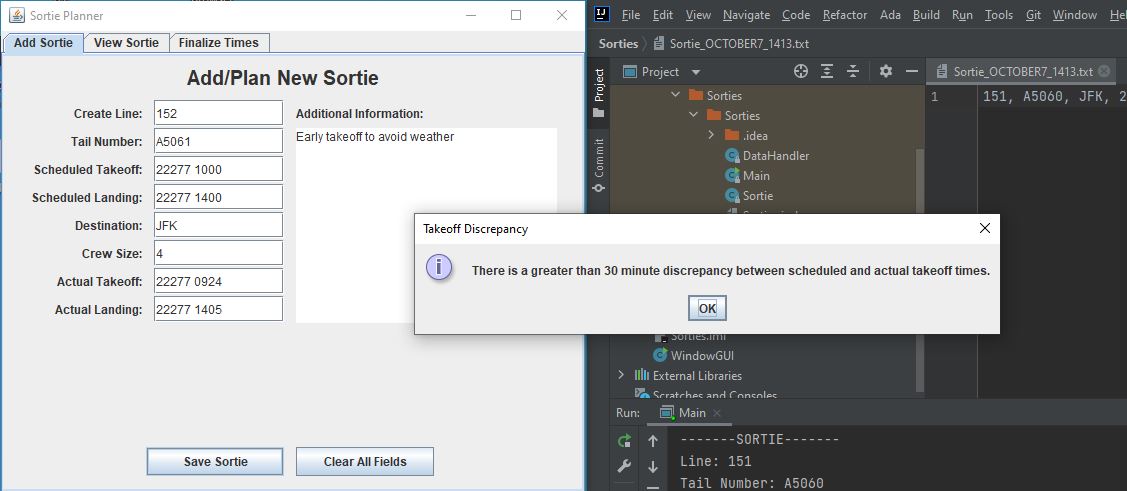


Catches time error that has a character other than an integer

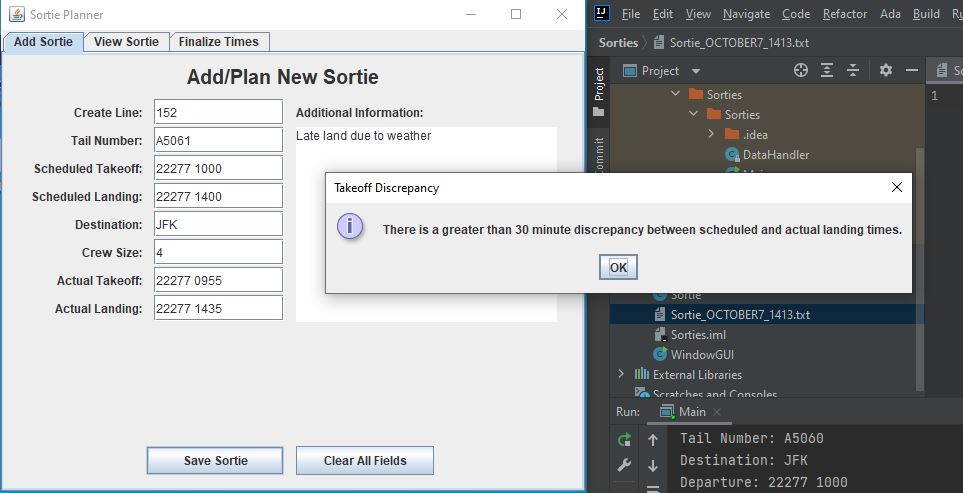


Catches time error that has too many digits

**Test Cases: 11, 12**



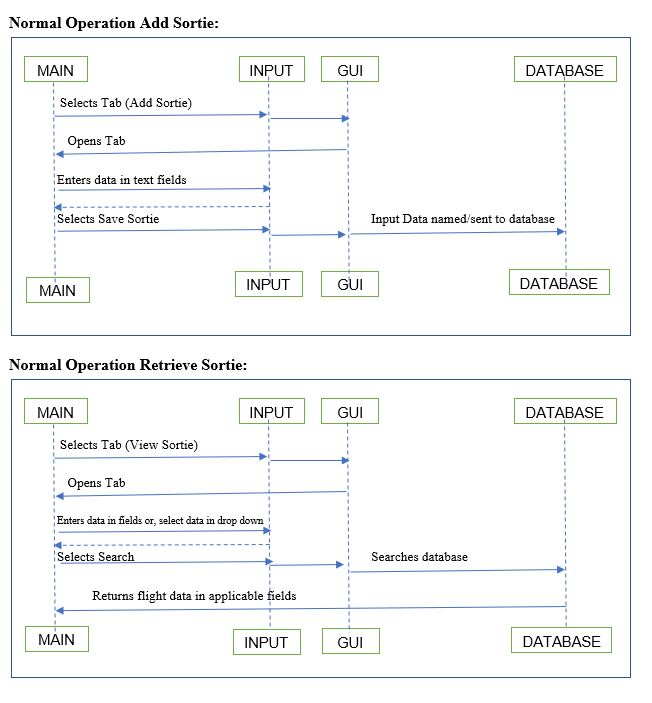
Take-off discrepancy catch, with added comment (more than 30 minutes early)



Land discrepancy catch, with added comment (more than 30 minutes late)

Design and Alternate designs

**Operation Diagrams**



Development History

**Code Revision History**

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Description** |
| Zachary Hager | 8/22/2022 | Initial GUI with drop downs and text boxes |
| Matthew White | 8/29/2022 | Refinements to critical components. Add data storing methods |
| Zachary Hager | 9/9/2022 | Migrated to tab-based interface & object “Sortie” coordination between 3 tabs (user)/1 class (code) |
| Zachary Hager | 9/25/2022 | Sortie class construction, exception handling in major fields, & save to IDE console |
| Charles Kimmel | 9/27/2022 | Consolidation of revised classes; program recompilable. |
| Zachary Hager | 10/3/2022 | Extensive revamp/Added error checking for all sortie fields when creating/saving a new sortie. |
| Zachary Hager | 10/5/2022 | Catch time discrepancies of 30 minutes |
| Matthew White | 10/6/2022 | Finalized method to store input data in to a .txt file |

**Comprehensive Development (Link)**

The entire development history of the program can be found at the GitHub link provided below. You will find all changes to the code as the program progressed, as well as our documentation along the way:

<https://github.com/platypus87/CMSC495FinalProject.git>

Conclusion

**Final thoughts:**

It was quite an excursion to get the program to this point over the last 8 weeks. Between being 1 under the maximum manning, time zone and schedule differences, and the fact our whole team has jobs outside of class it is quite amazing what we were able to pull off. We had wished the data retrieval was flusher, but we had spent too much time trying to work out a SQL online data base, that we had to readjust to just be able to create a file that can be retrieved. We could make these refinements given more time but overall, we are quite happy with what we have created. Being able to execute on each of our niches, and illuminating where some of our short comings were has been a fulfilling experience for all of us and we are happy that this program serves as a testament to that.

**Ways to Improve:**

**Data Retrieval**: The only means to pull the flight data is to open the .txt file that is created. We have the mechanisms within the program to display the data but they do not work. Future developers, can use the template we had laid out to build something more user friendly. Perhaps turn saves into .csv files and pull the data from there to display. With Java there are more built in mechanisms available to read .csv files versus .txt. We found it more important though to be able to get the data with incomplete component, over incomplete data with a slightly more refined method.

**Tail Number**: We had intended from the start to have a dropdown of different available aircraft tail numbers to select from (i.e. a limited number of aircraft to pull from), and add mechanisms to add more aircraft. We decided however to leave that to the user to fill as the user filling the proper entry is not very taxing on their end, and takes a lot of conditioning on the programing side. That being said it is a means by which one could improve the program by only having a fixed selection of aircraft to pull from.

**Storing Comments:** The program catches time deviations over or under 30 minutes requiring the user to leave a comment as too why, but does not store the comment. The building blocks were left in place for this to be built upon, but we were unable to reconcile this without undermining the rest of the data stored which we found to be much more important to have working.