**Design – Sortie Logger**

**Zachary Hager, Charles Kimmel, and Matthew White**

**Section 6381**

**Group 4**

**13 September 2022**

**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) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Prelude**

**Program Summary:**

This program will be 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.

**Design**

**Design Environment:**

Operating System: Windows 10 Home (latest version)

\*RAM: 256 MB

\*Disk space: 256 MB Total: (124 MB for JRE; 2 MB for Java Update, tentative 128 MB for program)

\*Processor: Minimum Pentium 2 266 MHz processor

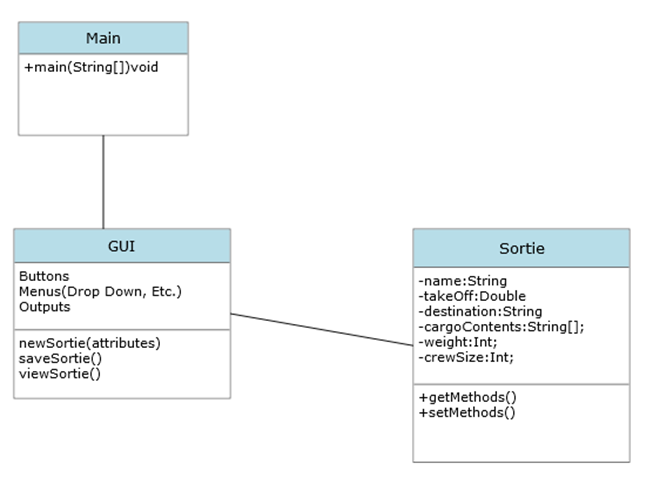
IDE: Eclipse and IntelliJ IDEA (latest version’s)

\*Denotes minimum requirements. Actual hardware being used will be exponentially more powerful, but is not pertinent as long as the minimum requirements are met as we are not test performance.

**Performance Considerations:**

Performance will not be a significant metric of evaluation in our program. This is just due to the nature of the type of program it is. Its main goal is archiving user input and there is no input sequence that would make it perform differently. Retrieving records by file names created by the program should not impact performance either no matter how large the archive becomes. Moreover, the machine that our program is run on should not make a significant difference in performance, as long as the minimums are met.

**Class Diagram:**

****

**Operation Diagrams**

**Normal Operation Add Sortie:**

DATABASE

MAIN

INPUT

GUI

Selects Tab (Add Sortie)

Opens Tab

Enters data in text fields

Selects Save Sortie Input Data named/sent to database

DATABASE

MAIN

INPUT

GUI

**Normal Operation Retrieve Sortie:**

DATABASE

MAIN

INPUT

GUI

Selects Tab (View Sortie)

Opens Tab

Enters data in fields or, select data in drop down

Selects Search Searches database

Returns flight data in applicable fields

MAIN

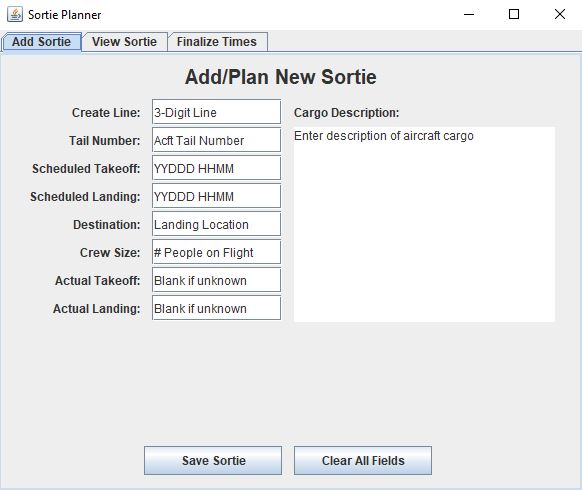
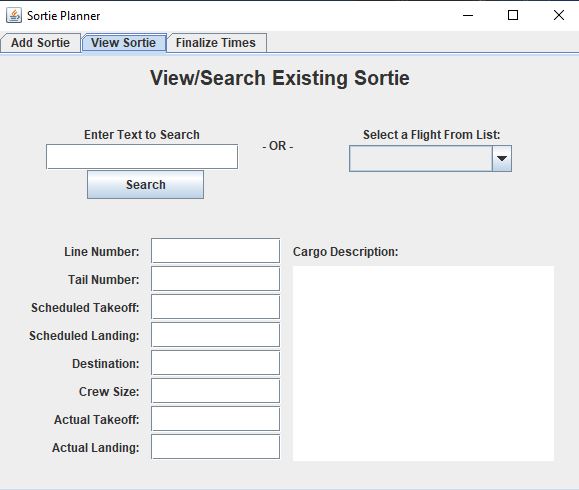
DATABASE

GUI

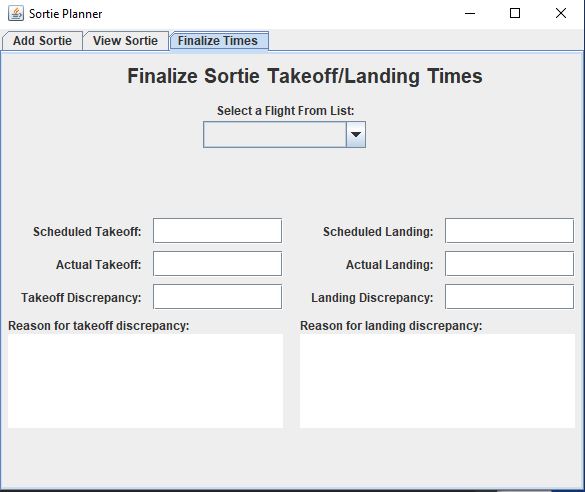
INPUT

**Screen Shots**

**Initial Page/Add Sortie Tab: View Sortie Tab:**

**\*Finalize Times Tab:**



\*Experimental feature we may migrate to for handling times and deviations.

**Pseudocode**

public class Main {

WindowGUI window = new WindowGUI();

void main (){

//to be worked