AVAJ LAUNCHER NOTES

1) INTRO:

- I am a self-taught software engineering student
- This is my first Java Project called **Avaj Launcher**
- The subject of this project was created by Academy+Plus which is a branch of 42

2) SUBJECT:

- implement a minimal **aircraft simulation program** based on a given UML class diagram.
- due to **frequent weather changes at an airport they is a bottleneck on some of the landing tracks**
- to find a solution, we need to know **which scenarios create the worst bottlenecks**
- so **we use a simulator to configure and analyze multiple scenarios** and hope that this will highlight them were the real problem is.

3) WHAT YOU NEED TO KNOW:

1) **UML Diagrams**: - relationships between classes,
   - visibity in attributes and methods,
   - interfaces and abstract classes
2) **Design Patterns**:
   - **Factory** : - creates an object based on the type of input passed.
     - e.g if the type is helicopter, only the helicopter class will be instantiated
   - **Observer**: - allows subject to **add (register)**, **update (notify)** and **remove (unregister)** observers
   - **Singleton**: - Creates only a single instance of a class
     - its constructor is private, only accessed by a get method
3) **File Handling:**
   - **Reading From a text file** e.g using a fileReader or bufferedReader
   - **Writing on a text file** e.g using a fileWriter or bufferedWriter
   - **Exception handling**, when you fail to read or write on a file

4) <u>HOW MY PROGRAM IS CONSTRUCTED</u>:

- First i read scenarios from a scenario.txt file using a **file/bufferedReader**
- if they is an error, an Exception is caught throught **Exception Handling**
- if no error a new flyable **object is created using the Factory Pattern**, then it is saved on array list. The created aircraft has its own unique ID
- I then read all my aircrafts from the array list registering them one by one to the weather tower **(adding observers in observer pattern)**
- in every cycle *(number of simulations which where read from a scenario file)* a **singleton instance** is called which generates new weather
- the weather is updated and registered **observers are notified using observer pattern**
- if the height of an aircraft becomes less or equal to zero, the aircraft is unregistered from the weather tower **(removing observer in observer pattern)**
- the **UML diagram** was converted to classes and nothing was modified
- I created another array list that stores messages as strings
- everytime a tower or aircraft says something, the messages are stored on my list
- at the end of my program, i write all my saved messages to a simulation.txt file using a **bufferedWriter or fileWriter**

5) <u>REQUIREMENTS AND PROGRAM BEHAVIOR:</u>

- implement an aircraft simulation program based on the class diagram provided
- **All classes** are required **to be implemented respecting every detail** provided in the diagram
- **you can add more classes or include additional attributes** if you think it is necessary, **but do not change access modifiers or the class hierarchy** for the classes provided in the diagram.
- The program **takes one and only one argument** from the command line which represents the name of a text file that will contain the scenario that needs to be simulated
- Executing the program will **generate a file simulation.txt** that describes the outcome of the simulation.
- 4 types of weather: **Rain**, **Sun**, **Fog** and **Snow**
- 3 types of aircraft: **Baloon**, **Helicopter** and **JetPlane**
- The height is in the 0-100 range. If an **aircraft needs to pass the upper limit height it remains at 100** and If an **aircraft reaches height 0 or needs to go below it, the aircraft lands, and unregisters** from the weather tower

6) <u>**THESE COMMANDS ARE RUN AT THE ROOT OF THE FOLDER**</u>:

a) find -name *.java > sources.txt
b) javac -sourcepath . @sources.txt
c) java azulu.main.Main scenario.txt

7) <u>**MY PROJECT RESOURCES**</u>:

a) <u>***Youtube:***</u>

- UML class diagrams ~Lucichart
- How to draw a UML diagram ~Barter Sessions
- UML class diagram ~Dr Mike Murphy
- UML to code conversion
- **Derek Banas** Java tutorials:
    - UML 2.0 class diagram
    - Design Pattern Video 4: Observer Pattern
    - Design Pattern Video 5: Factory Pattern
    - Design Pattern Video 7: Singleton Pattern
    - Java tutorial 6 : Exception Handling
    - Java tutorial 11: Collection classes *(ArrayList)*
    - Java tutorial 15: Interfaces and Abstract classes
    - Java tutorial: A 1000 page book in one video *(ArrayList and Exception Handling)*

b) <u>***Web Pages:***</u>

- Design patterns in java: - tutorialsPoint, javatPoint, codeGeeks, calliCoder, geekforGeeks, JornalDev, Dzone, Baeldung
- UML class diagram tutorial step by step ~ Salma
- UML 2 class diagram an Agile introduction
- Class diagram relationships in UML explained with examples
- Class diagram wikipedia

*avaj_launcher@**azulu2019***