Aviation Flight Computer

Author: Patience Chowning

Note: This flight computer will give a general solution to finding airspeed, time en route, as well as basic information regarding any plane. It will also give specific information regarding the Piper Cub and the Piper Arrow.

Piper Cub:

Source: https://www.flyingmag.com/piper-j-3-cubs-heritage-of-simplicity-reliability/

Piper Arrow:

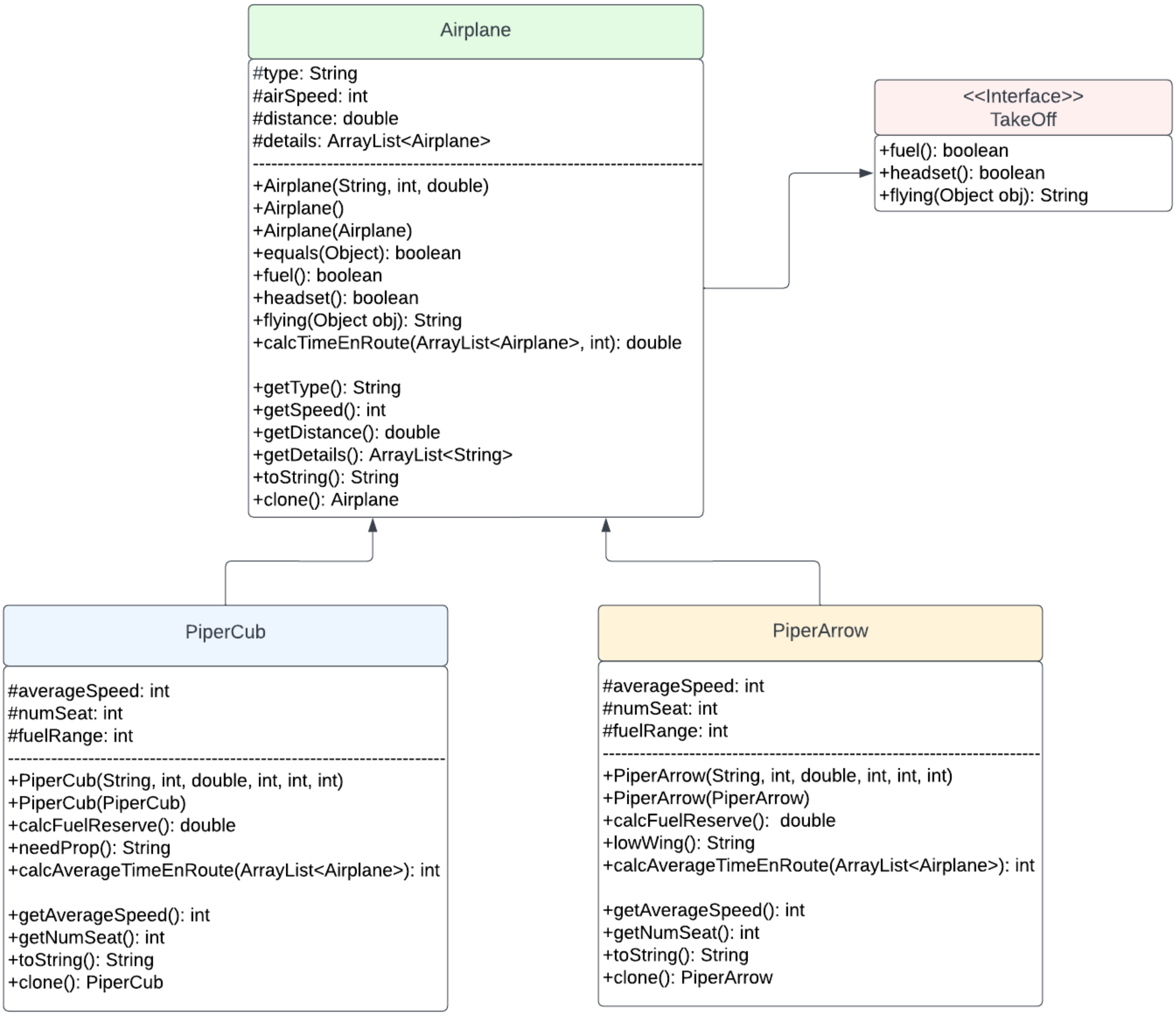
Source: https://www.aopa.org/go-fly/aircraft-and-ownership/aircraft-guide/aircraft/piper-arrow

Table of Contents:

Introduction:

The Aviation Flight Computer is a tool that will simplify two basic E6B calculations, airspeed and time en route. Using an E6B can become strenuous and tiresome over time so the use of the Aviation Flight Computer will ease your troubles and save you time. This flight computer aims to calculate and provide a “rough” estimate of both airspeed and time en route for any plane. It will also provide basic information regarding the Piper Cub and the Piper Arrow.

UML:



Airplane

#type: String
#airSpeed: int
#distance: double
#details: ArrayList<Airplane>

+Airplane(String, int, double)
+Airplane()
+Airplane(Airplane)
+equals(Object): boolean
+fuel(): boolean
+headset(): boolean
+flying(Object obj): String
+calcTimeEnRoute(ArrayList<Airplane>, int): double

+getType(): String
+getSpeed(): int
+getDistance(): double
+getDetails(): ArrayList<String>
+toString(): String
+clone(): Airplane

<<Interface>>
TakeOff

+fuel(): boolean
+headset(): boolean
+flying(Object obj): String

PiperCub

#averageSpeed: int
#numSeat: int
#fuelRange: int

+PiperCub(String, int, double, int, int, int)
+PiperCub(PiperCub)
+calcFuelReserve(): double
+needProp(): String
+calcAverageTimeEnRoute(ArrayList<Airplane>): int

+getAverageSpeed(): int
+getNumSeat(): int
+toString(): String
+clone(): PiperCub

PiperArrow

#averageSpeed: int
#numSeat: int
#fuelRange: int

+PiperArrow(String, int, double, int, int, int)
+PiperArrow(PiperArrow)
+calcFuelReserve(): double
+lowWing(): String
+calcAverageTimeEnRoute(ArrayList<Airplane>): int

+getAverageSpeed(): int
+getNumSeat(): int
+toString(): String
+clone(): PiperArrow

Explanation of Classes:

Airplane:

This superclass… Airplane holds basic instance variables and methods that can be implemented in both the classes: PiperCub and PiperArrow. It aids in the creation of an airplane object and helps to create better functionality/reuseability of the code. This class implements the interface TakeOff as well.

INSTANCE VARIABLES:
→ type: This is a string value of the type of airplane (either a cub or an arrow).
→ airspeed: This is an integer value of the indicated or “real time” airspeed of the airplane.
→ distance: This is a double value of the total distance in which the plane will be travelling.
→ planes: This is an array list of String values regarding details of the airplane. This includes the type of the airplane (i.e., plane/helicopter/commuter), the make of the plane (i.e, Cub, Arrow, Fighter), the airspeed, and the distance in which it will travel.

CONSTRUCTORS/Methods:
→ Airplane: This constructor is used to create an object of the airplane class. It includes the type of airplane, the speed of the airplane, followed by the distance in which the airplane will travel.
→ Airplane(Airplane): This will return a copy of the Airplane object which will help to implement the clone() method later on.
→ Airplane(): This is the default constructor for the Airplane class.
→ equals: This method will demonstrate whether the time en route is the same for two objects.
→ calcTimeEnRoute(ArrayList<Airplane>, int): This method calculates the time en route for any plane and returns a double value. This is found by taking the (distance / indicated air speed) * 100.

INTERFACE METHODS:
→ fuel: This will tell the pilot whether they have sufficient fuel in the airplane to fly by returning a boolean value of true or false (this must be true before flying).
→ headset: This will check whether the pilot has their headset connected (this must be true before flying).
→ flying(Object obj): This will indicate whether the plane is flying by returning a String value. In order to be flying, both headset and fuel need to be true. This method takes in an Airplane object to be tested.

GETTER METHODS:
→ getType: This will return the type of the plane as a string value.
→ getSpeed: This will return the airspeed of the plane as an integer value.
→ getDistance: This will return the total distance that the plane plans to fly as a double value.
→ getDetails: This will return an array list of details.

TakeOff:

METHODS(These will be implemented in the Airplane class):
→ fuel: This will tell the pilot whether they have sufficient fuel in the airplane to fly by returning a boolean value of true or false (this must be true before flying).
→ headset: This will check whether the pilot has their headset connected (this must be true before flying).
→ flying: This will indicate whether the plane is flying. In order to be flying, both headset and fuel need to be true.

PiperCub:

INSTANCE VARIABLES:
→ averageSpeed: This will return the integer value of the average speed of the Piper Cub. This number should be 65 knots which is equivalent to ~75mph. *This is not to be confused with the variable airspeed which indicates the current speed of the plane.
→ numSeat: This will return an integer value of the number of seats in the Piper Cub. This number should be 2.
→ fuelRange: This will return the average fuel range for a Piper Cub. This value is 460 miles on a full tank.
*Note: This number is a general estimate and the actual mileage you can fly will vary on multiple factors not taken into account in this project.

CONSTRUCTORS
→ PiperCub(String, int, double, int, int, int): This will create a PiperCub object using inheritance from the Airplane class. This method should use the keyword *super*.
→ PiperCub(PiperCub): This will return a copy of the Piper Cub object which will help to implement the clone() method later on.

METHODS:
→ calcFuelReserve(): This will return the amount of miles left that the Cub can fly by using the fuelRange value - distance value.
→ needProp(): This will return a string value for the Piper Cub. To fly a Cub you have to have someone prop (spin the blade) manually.
→ calcAverageTimeEnRoute(ArrayList<Airplane>): This will calculate the estimated time en route by taking the ((fuelRange / average speed) * 100) / 60) and return an integer value for the pilot.
→ toString(): This will return a string representation of the PiperCub object as well as additional details regarding the plane.
→ clone(): This will clone the PiperCub object.

GETTER METHODS:
→ getAverageSpeed(): This will return the average speed of the PiperCub.
→ getNumSeat(): This will return the number of seats in the PiperCub.
→ getFuelRange(): This will return the fuel range for the Piper Cub.

PiperArrow:

INSTANCE VARIABLES:
→ averageSpeed: This will return the integer value of the average speed of the Piper Arrow. This number should be 131 knots which is equivalent to 151mph. *This is not to be confused with the variable airspeed which indicates the current speed of the plane.
→ numSeat: This will return an integer value of the number of seats in the Piper Arrow. This number should be 4.
→ fuelRange: This will return the average fuel range for a Piper Arrow. This value is 900 miles on a full tank.
*Note: This number is a general estimate and the actual mileage you can fly will vary on multiple factors not taken into account in this project.

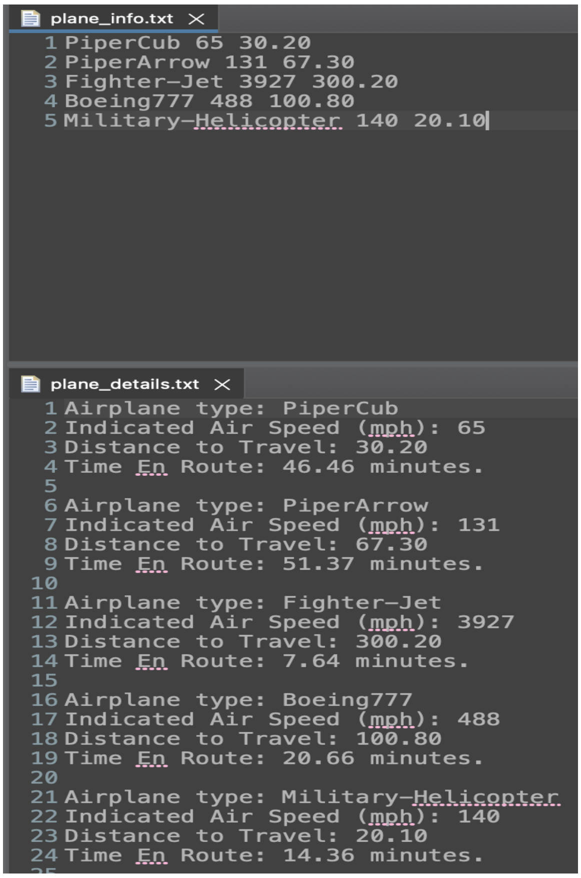
CONSTRUCTORS
→ PiperArrow(String, int, double, int, int, int): This will create a PiperArrow object using inheritance from the Airplane class. This method should use the keyword *super*.
→ PiperArrow(PiperArrow): This will return a copy of the Piper Arrow object which will help to implement the clone() method later on.

METHODS:
→ calcFuelReserve(): This will return the amount of miles left that the Arrow can fly by using the fuelRange value - distance value.
→ lowWing(): This will return a string value because the Piper Arrow is considered a low wing airplane.
→ calcAverageTimeEnRoute(ArrayList<Airplane>): This will calculate the estimated time en route by taking the ((fuelRange / average speed) * 100) / 60) and return an integer value for the pilot.
→ toString(): This will return a string representation of the PiperArrow object as well as additional details regarding the plane.
→ clone(): This will clone the PiperArrow object.

GETTER METHODS:
→ getAverageSpeed(): This will return the average speed of the PiperCub.
→ getNumSeat(): This will return the number of seats in the PiperCub.
→ getFuelRange(): This will return the fuel range for the Piper Arrow

Text Files:

This text file holds basic information regarding the airplane (i.e., cub, arrow, helicopter, etc.). It then provides the indicated airspeed of the aircraft followed by the total distance to be travelled. A new text file will be created and it should hold all airplane objects, as well as their calculated time en route. This output file will be called, “plane_details.txt”.



```
PiperCub 65 30.20
PiperArrow 131 67.30
Fighter-Jet 3927 300.20
Boeing777 488 100.80
Military-Helicopter 140 20.10
```

```
Airplane type: PiperCub
Indicated Air Speed (mph): 65
Distance to Travel: 30.20
Time En Route: 46.46 minutes.

Airplane type: PiperArrow
Indicated Air Speed (mph): 131
Distance to Travel: 67.30
Time En Route: 51.37 minutes.

Airplane type: Fighter-Jet
Indicated Air Speed (mph): 3927
Distance to Travel: 300.20
Time En Route: 7.64 minutes.

Airplane type: Boeing777
Indicated Air Speed (mph): 488
Distance to Travel: 100.80
Time En Route: 20.66 minutes.

Airplane type: Military-Helicopter
Indicated Air Speed (mph): 140
Distance to Travel: 20.10
Time En Route: 14.36 minutes.
```

*plane_info.txt file to be used initially