Air traffic control system

Assignment brief:

Time 3 hr (+1 hr to upload) (closed book).

You have been given some code that is currently being developed as part of an **Air Traffic Control** application. Not all the requirements have been implemented. It is your task to implement these, test your **solution** classes and raise the coding standards of all the code.

Create a **project solution** (named **p2<your student id> e.g. p26048201).** Create a package named **p2**. Add **Aircraft.java** to the solution**. Ensure your name and student number are placed in the Javadoc comments of all the classes you create.**

Part 1 – 40%

Using your knowledge of OOP you should update the code based on the following:

1. The system is expected to support many ***derivative*** (sub classes) of *Aircraft* such as **Airline**, *Helicopter*, *Glider, Drone* etc. There will be no need to ever instantiate an *Aircraft* class in the system, it this therefore *Abstract*. You are responsible for developing the ***Airline*** class. This is the only subclass you need to develop. It should have all the properties and methods of the *Aircraft* class but also include a new attribute, **Engine Type** which should be limited to the values ***jet*** or ***propeller***.

2. **Business rules for class attributes –**
   - **Current Speed -** minimum value is 0 and maximum is 800 (inclusive). This is in mph.
   - **Distance to Airfield –** minimum value is 1 and maximum is 20000 (inclusive). This is in miles.
   - **Aircraft Code –** exactly 6 characters (any characters including letters, numbers or symbols but first character should be an uppercase **A.** No trailing whitespace characters.)
   - **Engine Type** - which should be limited to the values **jet** or **propeller**.
   - Return an appropriate exception with an appropriate exception message i.e. "*INVALID CODE*", "*INVALID SPEED*" etc. if an attempt is made to set outside the range or allowable values.

3. Some but not all of the Aircraft class derivatives (sub classes) are expected to support *a method that will calculate the estimated time (in minutes) any aircraft subclass will take to reach the airfield.* This will require each class that needs this functionality to have a method named **timeToAirfield.** The Airline class requires this method and should return the time (in minutes) value based on the following formula.

   time = distance to airfield / current speed * 60

[CONTINUED OVER]

Part 2 - Airtraffic Control class - 30%

Create an ***Airtraffic Control*** class to support the system for searching and other utility methods. Each method should be ***static*** and accept and return an ***ArrayList*** of **appropriate type**. Using your knowledge of OOP you should update the code based on the following:

1. Create a **searchBySpeed** method i.e. search for all objects in the parameter argument ArrayList that are *moving (current speed) within a specified range* e.g. between 100 and 300 (range inclusive of both values). You should return an ArrayList containing any that satisfy the search criteria.

2. Create a **searchForAllByEngineType** method i.e. search for all objects in the parameter argument ArrayList that match a specified engine type. You should return an ArrayList containing any that satisfy the search criteria.

3. Create a **searchForLandingList** method i.e. search for all objects in the parameter argument ArrayList that are within a given time (inclusive) of the airfield and have a specified engine type. You should return an ArrayList containing any that satisfy the search criteria.

Part 3 – Testing – 30%

1. Unit Test the application.

When complete compress (zip) the entire ***Eclipse solution*** and upload to **Assignments** (P2 assessment) on CANVAS. Remember to record and then upload a short commentary walk-through of your code with your solution. Keep the separate screen recording safe (no need to upload at this point).

Now : check the uploads to ensure you have submitted the correct files.

[END]