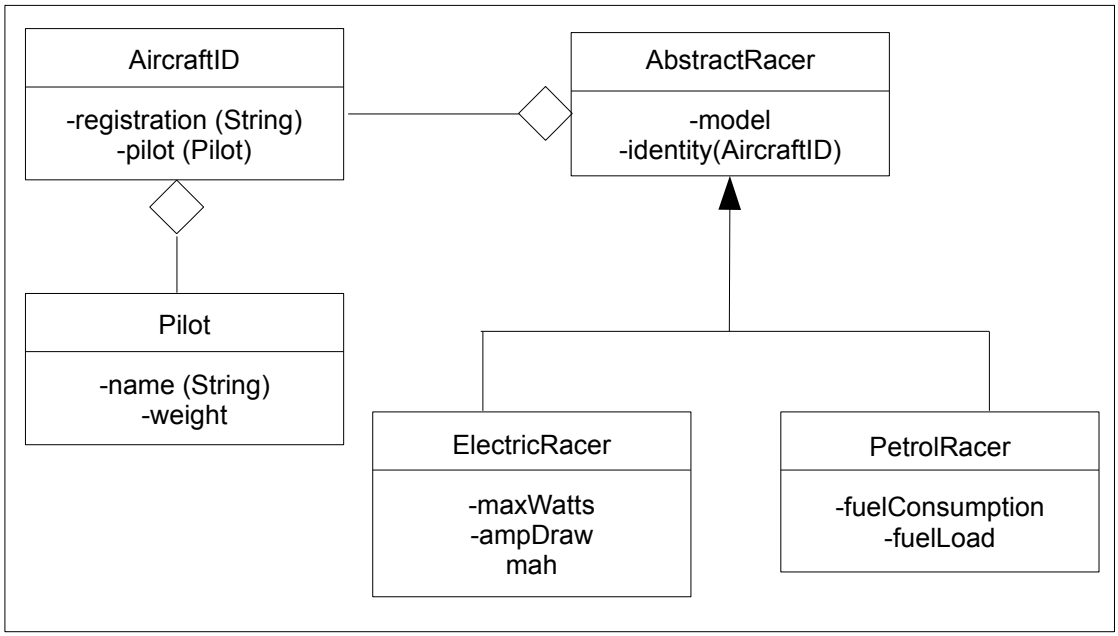
Worksheet Eight

There are of course many problems connected with life, of which some of the most popular are "Why are people born?", "Why do they die? and "Why do they spend so much of the intervening time wearing digital watches?".
Douglas Adams, The Hitch Hikers Guide to the Galaxy 1979

Unit Learning Outcomes Addressed by this worksheet: 1, 2, 3 & 4.

Make electronic copies of all of your algorithms. Place these, along with your Java code, in your P08 directory. You will need to copy AircraftID.java, Pilot.Java and Battery.java from your P07 directory.



The Task Continues
Last week you completed the classes:

- AircraftID
- Pilot

This week you need to complete the remaining classes. The class fields for each of these classes are described on the following pages.

The Class AbstractRacer has the following class fields:

- model: A String describing the model of the plane (e.g. "Edge-540"). Default is "No-Model".
- identity: An AircraftID object describe the pilot and registration (see worksheet 7).
- Two AbstractRacer objects are considered equal if, and only if, they have the same model and identity.

The class PetrolRacer has the following class fields:

- fuelConsumption: A real number specifying the fuel consumption, measured in gallons per hour (gph). The allowable range is 8.0 to 15.0 gph. The default values is 10.0 gph.
- fuelLoad: A real number specifying the amount of fuel, in U.S. gallons that the plane will carry on each race flight. The allowable range is 5.0 to 25.0 gallons. The default is 10.0 gallons. Allowing for take off and landing and waiting before each race, each plane should ensure that they have enough fuel for a minimum of 30 minutes of flight at full power.
- Two PetrolRacer objects are considered equal, if and only if, they have the same model, identity and fuelLoad.

The class ElectricRacer has the following class fields:

- maxWatts: A real number specifying the maximum power output of the electric motor. The allowable range is 400.0 to 600.0 and the default value is 400.0.
- ampDraw: a real number specifying the maximum amps which can be drawn from the battery at any time (allowable range is 20 to 60 amps). Default amperage is 40.0 amps.
- mah: A real number specifying the maximum capacity of the battery when full charges (allowable range 10000.0 to 20000.0). Default mah is 10000.0 mah.
- Two ElectricRacer objects are considered equal, if and only if, they have the same model, identity and maximum watts.

As with the classes in worksheet seven all of your classes should:

- Conform to the dept of Computing Java coding standard.
- Declare all class fields as being private.
- Have a default constructor.
- An alternate constructor which IMPORT's appropriate data for intialising the class fields.
- A copy constructor.
- A set of accessors which:
    - Allow the extraction of the state information (i.e. the *get'ers*).
    - Include an appropriate equals method.
    - Include an appropriate toString method.
- Mutator(s) which allow the object state to be updated (i.e. the *set'ers*).
- Any required private methods.

AbstractRacer also has an abstract method. Develop the all the classes without the abstract method and, when you have them all compiling, add the following abstract method to AbstractRacer and its full implementation in each of the sub classes.

The abstract method is:

- Called calcFlightDuration
- IMPORTs nothing.
- EXPORTS duration: A real number specifying the maximum amount of time (in minutes) the plane can stay in the air with the motor running at full power.

In petrol planes the duration is calculated via:

duration = (fuelLoad/fuelConsumption) x 60.0

In electric planes the duration is calculated via:

duration = 60.0 / ((ampDraw x 1000.0) / mah)

Points to note:

- Validating the fuel load in petrol planes must ensure a minimum flight time of 30 minutes.
- You will need to access superclass functionality from the sub classes so keep track of what class fields are in which classes.
- Use the lecture examples to guide your understanding of how super and sub classes interact (e.g. super class constructor calls from the sub class constructor).
- DO NOT deal with the abstract method until AFTER you have everything else working. You needed to understand the other concepts first or you will get confused.

Finally Design in pseudo code and implement in Java, an algorithm which will:

- Prompt for the aircraft type (petrol or electric).
- Inputs the required information and constructs the appropriate aircraft object (i.e. ElectricRacer or Petrolracer).
- Outputs the maximum flight duration for the racer.

Note that this algorithm will be in another class (called CalcFlight) which will contain main and some sub modules.