Worksheet Seven

""Forty-two!", yelled Loonquawl. "Is that all you have to show for seven and a half million years's work?"
"I checked it very thoroughly", said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the questions is".
Douglas Adams, The Hitch Hikers Guide to the Galaxy 1979

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

Make electronic copies of all of your algorithms. Place these, along with your Java code, in your P07 directory.

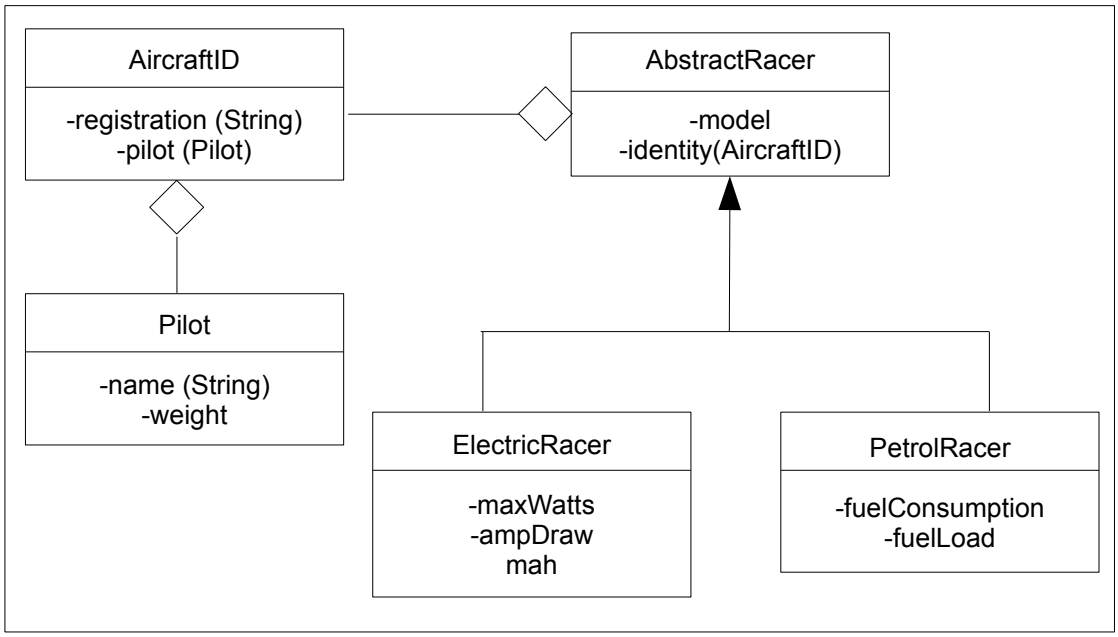
For worksheets 7, 8 and 9 you will be developing a set of classes which will build upon each other (e.g. worksheet seven classes will be required for worksheet eight, worksheet seven and eight classes will be required for worksheet nine. It is therefore essential that you keep up to date with these exercises (remember the senior tutor and I are available for help).

Background
Air racing has become a highly popular sport around the world. The Brown Cow Air Race stages a series of races in cities across the world. They have contracted your company to develop a series of software tools that the air race competitors can use to fine tune their aircraft and ensure that they are as competitive as possible. Traditionally the racing planes have been powered by 6-8 cylinder petrol engines but break through technology has resulted in a new class of battery powered racing planes.

The Task
The software engineering requirements and design team have come up with a set of specifications for 372 classes which will be combined in various ways to form the required software team. You have been instructed to design in pseudo code, implement in Java and then finally to test the classes shown on the UML class diagram shown on the next page.. This will be accomplished via worksheet's seven and eight and will be required for your assignment as well as you being tested on your solutions in test two.

Each class icon shows the class name and below that, the class fields required for each class.

-



IMPORTANT INFORMATION ABOUT RACE PLANES:

Aircraft were around long before the world went metric. Errors in converting between metric and imperial measurements have been the cause of countless air crashes over the years. For this reason the racing organisers have elected to keep all measurements imperial. This means that:

- All weights are measured in pounds (lbs).
- All volumes are measured in U.S. gallons.

Other related useful information is:

- Fuel consumption of petrol driven aircraft engines is measured in gallons per hour (gph).
- Fuel consumption is specified as the fuel consumption of the aircraft engine at full throttle.
- One gallon of fuel weighs 6 lbs.
- Because the race flights are short they are measured in minutes and not hours.
- All real numbers are expressed to an accuracy of 2 decimal places.

Your task for worksheet seven is to design in pseudo code and implement in Java the following classes:

- Pilot
- AircraftID

The remaining classes will be implemented in worksheet 8.

Each of the classes above should have all of the standard accessors, mutators and constructors as stressed in the lecture examples and in the DateClass example (available on the ST151 web area).

Each of the classes is described in more detail below.
The Class AircraftID has the following class fields:

- registration: A String containing the registration of the aircraft (e.g. VH-SRR). This cannot be validated. Default registration is "VH-NON"
- pilot: A Pilot class object (see below).

The class Pilot has the following class fields:

- name: A string containing the name of the pilot. This cannot be validated. Default name is "No Name".
- weight: A real number specifying the weight of the pilot in pounds (allowable range is 100 to 200 lbs). Default weight is 120.0 lbs.

Your design should ensure that objects of any of these classes will always have a valid object state. Use the lecture examples and the DateClass example as a guide in this.

Design and implement Pilot, followed by AircraftID. That way you will be tackling them in the order of least to most difficult.

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 initialising 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.

For each class design a test harness which tests the constructors in the class. Use the test harness example form the lectures (note not the lecture notes but the example covered in the lecture) to guide you.