CS1527 Object-Oriented Programming (2023-2024)

Assessment 1: Boat Management System (8/2 -22/2, 2024)

Important
This assessment is worth 30% of the overall marks for course CS1527.

You must **submit your code via Codio before 23.59 on 22th February**. Make sure your solution is ready to run before then.

The markers will look at your code and will visually check its output. If the code does not run, or the output is significantly different from that expected, you may get a much lower mark.

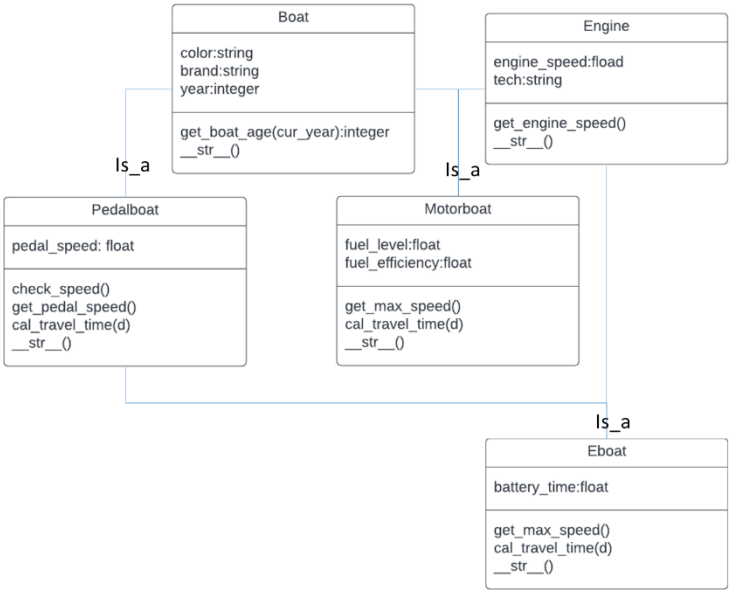
Remember that as this is an individual assessment, work submitted must be your own. If you haven't already done so, you should familiarise yourself with the University guidance on plagiarism, available at https://www.abdn.ac.uk/sls/online-resources/avoiding-plagiarism/

If you reuse other people's code or libraries, you must explicitly mention this in your code and acknowledge their work. This includes:

I. Where did you acquire them (e.g. URLs, books, GitHub)?
II. Who are the authors? They cannot be your classmates, of course!!
III. How did you modify them? In particular, if you make minor changes to a piece of other people's code, you must clearly mark in your code which part is your own code.

Introduction

You have been recruited to join a software team that is developing a boat management system. The diagram to the right shows part of a solution for the system that includes three typical kinds of boats: the traditional pedal boat run by pedalling; the fuel-powered motorboats; and the increasingly popular pedal assist electric boats (eboats). Note that motorboats and eboats have different engines, with gas powered technology for the former and electrically powered technology for the latter.



Definition of instance attributes are as follows:

- *color*: color of boat
- *brand*: boat brand
- *year*: manufacture year
- *engine_speed*: the maximum speed (miles per hour, mph) of a boat provided by its engine.

- *tech:* technology used in engine. It has two possible values: 'gas' and 'electric'. The attribute *engine_speed* gets a value of 80 mph for *tech* = 'gas' and 20 mph for *tech*='electric'.
- *fuel_level:* Remaining level of fuel (gas, gallon) in the tank of a motorboat.
- *fuel_efficiency:* it shows how far your boat can travel with a certain amount of fuel. It is described as miles per gallon (mpg).
- *pedal_speed:* the speed (mph) of a boat through pedalling.
- *battery_time:* the remaining battery draining time (hour).

Definitions of instance methods are as follows:

- *get_boat_age(cur_year):* calculates the age of a boat when the current year (*cur_year*) is provided.
- *get_engine_speed():* get the boat speed provided by engine, i.e. engine_speed.
- *get_max_speed():* computes the maximum speed that a boat can have.
- *cal_travel_time(d):* calculates the shortest time required by a boat to travel a distance d (miles). If a boat (only motorboat has this problem as other boats can achieve it at least by pedalling) cannot finish the given distance due to running out of fuel, then return the actual travel time before it runs out of fuel and prompt a message to the display to show the remaining distance, such as "This motorboat runs out of fuel 100 miles away from the destination."
- *check_speed():* check if a pedalboat's speed is within the realistic range [10, 20], return true or false. In the false case, the pedal_speed should be set to the closest value in this range.
- *get_pedal_speed():* returns the correct pedalling speed.
- *__str__():* get a string representation of an instance of class. The *__str__* method should be defined in a way that is easy to read and outputs all the members of the class

In addition, a class attribute called *all_boats* should be defined in the Boat class to hold list of Boats created.

You have been given a template corresponding to this design produced by another programmer. Please use this template to complete the design (appropriate implementation of all above-mentioned class attribute, instance attributes, and instance methods in their corresponding classes). Additional instance methods other than the above required ones are allowed, should they help ease your implementation. At the end of the template, test instances are provided, and your solution should run smoothly without any modification of these test instances. Otherwise, your solution won't pass these tests when you submit your code in Codio. The markers will grade your work by inspecting your code regardless of the tests. **Please make sure you use a single python file 'assessment1.py' to develop your solution. A solution that uses multiple python files will not be accepted.**

You must submit your code for **this first assessment via Codio**.

This is a new assignment, and the best materials you should use are the practicals 2 & 3. To save your time for the assessment, I have done extensive googling of possible solutions and nothing can be found.

Marking criteria

You will be given a mark ranging from 0 to 100 (bear in mind this assessment is worth 30% of the overall marks).

0 – no submission

10-40 – minimal additions to starting code with some methods working

40-60 – some (up to half) instance attributes and instance methods in all classes are correctly implemented. The program runs smoothly

60-70 – most instance attributes and instance methods in all classes are correctly implemented. The program runs smoothly

70-85 – everything runs as it should and the design is fully achieved, but poorly structured methods

85-100 – all requirements satisfied and code well-structured in single-purpose methods to realise the design