This assignment will introduce you to writing a class to represent a specific topic (a clock) and then work with instances (objects) of that class. You then utilize your Clock class (**clock.py**) in a ClockShop class.

Your first task is to create a class called Clock. A clock represents time in hours, minutes, and seconds. The type of time it represents is military time (hours are from 0 to 23, with 0 representing midnight and 23 representing 11pm). Here are the things a user of a Clock object should be able to do:

- Clock Constructor **__init__(self, hour, minute, second)**: create a Clock object and specify the starting hour, minute, and second via parameters (we will use integer values for our data)
    - it defaults to 00:00:00 (hours, minutes, seconds) (12am) if no values are specified at creation
- **__str__(self)**: get the current time in String format -- the object will return a String with the hours, minutes and seconds as follows: hh:mm:ss (ex: 9:49:59)
- **hour(self):** -- the object will report as follows: hh (ex: 18 -- which means it is 6pm) -- the value returned should be the hour
- **minute(self):** -- the object will report as follows: mm (ex: 56) -- the value returned should be the minute
- **second(self):** -- the object will report as follows: ss (ex: 6) -- the value returned should be the second
- **set_hour(self, new_hour):** set the current hour (a number from 0 through 23 will be passed in to the object from outside) (any value outside this range should raise an exception with an appropriate error message)
- **set_minute(self, new_minute):** set the current minute (a number from 0 through 59 will be passed in to the object from outside) (any value outside this range should raise an exception with an appropriate message)
- **set_second(self, new_second):** set the current second (a number from 0 through 59 will be passed in to the object from outside) (any value outside this range should raise an exception with an appropriate message)
- **advance_hour(self, amount_to_advance):** advance the current hour (a number 0 or greater will be passed in to the object from outside -- the resulting hour should be represented in the range of 0 to 23) (HINT: % is your friend) (any value less than 0 should raise an exception with an appropriate message)
- **advance_minute(self, amount_to_advance):** advance the current minute (a number 0 or greater will be passed in to the object from outside -- the resulting minute should be represented in the range of 0 to 59 ***AND*** the hour should be updated accordingly) (any value less than 0 should raise an exception with an appropriate message)
- **set_to_current_time(self):** set the time to the current time (hours, minutes, and seconds) based on what your computer reports.
- **__eq__(self, other):** compare against another Clock object for equality (an equals method)
- **__lt__(self, other):** compare against another Clock to see if current clock's time is before other clock
- **__gt__(self, other):** compare against another Clock to see if current clock's time is after other clock

You are free to add more functionality to your Clock class as you see fit. Write code as necessary to test that your Clock class functions as specified.

Next create a class **ClockShop (place in clockshop.py)**. This class will hold list of **Clock** objects. It will provide behaviors (methods) that allow for sorting, searching, retrieving, and printing the **Clocks** it contains. Specifics for this class are as follows:

- Declare a private field that is list of type **Clock**
- **fill_clock_shop(self, list_of_times)**: the method walks through the list of times (strings formatted hh:mm:ss, guaranteed), creates Clock objects, and adds them to your list of clocks.
- **sort_clocks(self)**: sorts the **Clock**s in ascending (smallest time to largest time) order. You must write the code to sort the array of **Clock**s
- **find_clock(self, a_clock)**: it is passed a **Clock** object and checks to see if it exists in the **Clock** list. If it does exist, return the index of where the first one was found. If it does not exist, return a -1.
- **__str__(self)**: builds a **String** containing each **Clock** object in **String** format (via the **__str(self)__** in the **Clock** class) and returns that **String**. Each **Clock** object should be separated from the others by a newline (\n).
- **get_clock(self, index)**: it is passed an integer index of which **Clock** to retrieve from the list of **Clock**s. If the index is outside the range of the array, raise an exception with an appropriate message. If the index falls within the list, return the **Clock** object at that index.
- **set_clock(self, a_clock, index)**: it is passed a **Clock** object and an index. The method places the **Clock** object at the specified index in the list. If the index is outside the range of the list, raise an exception with an appropriate message.

Your code must work properly with the ClockShopTester code given below. Once you are done with your two classes, utilize the tester and make sure your code works correctly. This code will be posted within a couple of days to Canvas.