Practice Problem Set – Classes and Objects

1. Write code to develop a class called `Circle` which contains the following public attributes and methods. Decide appropriate attributes/parameters and return types.
   - Attributes: `radius` and `color`
   - Methods:
     - `setRadius(r)`: sets radius value to `r`
     - `getRadius()`: returns radius value
     - `setColor(c)`: sets color name to `c`
     - `getColor()`: returns color name
     - `getCircumference()`: returns circumference of the circle
     - `getArea()`: returns area of the circle

   Instantiate necessary objects to test the functionality of this class.

2. Write code to develop a class `BankAccount` that supports the following methods:
   - `withdraw()`: takes an amount as input from user and withdraws it from the balance
   - `deposit()`: takes an amount as input from user and adds it to the balance
   - `balance()`: returns the balance on the account

   Make all the methods public and decide appropriate parameter(s) and return types wherever needed. Also create appropriate public attributes. Instantiate necessary objects to test the functionality of this class.

3. Repeat problem 2, making all attributes private. Make appropriate changes in the code wherever needed.

4. Write code to implement class `Worker` that supports two private attributes `hoursWorked` and `wageRate`, and the following public methods:
   - `setHoursWorked(h)`: sets `hoursWorked` to `h`
   - `changeRate(r)`: changes the worker's pay rate to the new hourly rate `r`
   - `pay()`: returns the pay as product of `hoursWorked` and `wageRate`

   Instantiate necessary objects to test the functionality of this class.

5. Repeat problem 4, adding constructor to initialize the attributes. Also set attributes' default values.

6. Write code to develop a class named Vehicles having three private attributes `noOfWheels`, `color` and `modelNo`; define related constructor and public getter/setter methods. Use appropriate types for all attributes, method parameters and return values wherever needed. Instantiate necessary objects to test the functionality of this class.

7. Write code to develop a class `Engine` having private attributes `engineNo` and `dateOfManufacture`, along with appropriate constructor and public getter/setter methods. Decide appropriate types for all attributes, method parameters and return values for this new class.

8. Write code to create a class that imitates part of the functionality of the basic data type int. Call the class `Int` (note different capitalization). The only data in this class is an int variable. Include methods to initialize an `Int` to any value (0 being default), to change it anytime and to display it, and to add two `Int` values. Write a program that exercises this class by creating one uninitialized and two initialized `Int` values, adding the two initialized values and placing the response in the uninitialized value, and then displaying this result.

9. Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay a Rs. 50 toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Write code to model this tollbooth with a class called `TollBooth` having two attributes: to hold the total number of cars, and to hold the total amount of money collected. A constructor initializes both of these to 0. A method called `payingCar()` increments the car total and adds Rs. 50 to the cash total. Another method, called `nopayCar()`, increments the car total but adds nothing to the cash total. Finally, a method function called `display()` displays the two totals. Include a program to test this class. This program should allow

the user to select one key to count a paying car, and another to count a nonpaying car. Selecting the Esc key should cause the program to print out the total cars and total cash and then exit.

10. Write code to create a class called `Time` that has separate member data for hours, minutes, and seconds. Make constructor to initialize these attributes, with 0 being the default value. Add a method to display time in 11:59:59 format. Add another method `addTime` which takes one argument of `Time` type and add this time to the current time of the *self* object. Instantiate two objects `t1` and `t2` to any arbitrary values, display both the objects, add `t2` to `t1` and display the result.

11. In ocean navigation, locations are measured in degrees and minutes of latitude and longitude. Thus if you're lying off the mouth of Papeete Harbor in Tahiti, your location is 149 degrees 34.8 minutes west longitude, and 17 degrees 31.5 minutes south latitude. This is written as 149°34.8' W, 17°31.5' S. There are 60 minutes in a degree (an older system also divided a minute into 60 seconds, but the modern approach is to use decimal minutes instead). Longitude is measured from 0 to 180 degrees, east or west from Greenwich, England, to the international dateline in the Pacific. Latitude is measured from 0 to 90 degrees, north or south from the equator to the poles. Write code to create a class `Angle` that includes three member variables: int for degrees, a float for minutes, and a char for the direction letter (N, S, E, or W). This class can hold either a latitude variable or a longitude variable. Write one method to obtain an angle value (in degrees and minutes) and a direction from the user, and a second to display the angle value in 179°59.9' E format. Also write a three-argument constructor. Write a main program that displays an angle initialized with the constructor, and then, within a loop, allows the user to input any angle value, and then displays the value. You can use the hex character constant `'\xF8'`, which usually prints a degree (°) symbol.

12. Write code to create a class `Tracker` that includes a data member that holds a *serial number* for each object created from the class. That is, the first object created will be numbered 1, the second 2, and so on. For this, create a class attribute named `count`; then, as each object is created, its constructor can examine this count member variable to determine the appropriate serial number for the new object. Add a method that permits an object to report its own serial number. Then write a main program that creates three objects and queries each one about its serial number. They should respond I am object number 2, and so on.

13. Create a class called `Ship` that incorporates a ship's number and location. Use the approach of problem 14 to number each ship object as it is created. Use two variables of the `Angle` class from problem 13 to represent the ship's latitude and longitude. A method of the `Ship` class should get a position from the user and store it in the object; another should report the serial number and position. Write a main program that creates three ships, asks the user to input the position of each, and then displays each ship's number and position.