LAB#06 EXERCISES

Note: Attempt at least five questions in Lab up to 4:00 pm where question four and 5 both are mandatory to attempt in Lab
There is 1.5 weightage work performed in Lab.

QUESTION#1

Write a Java program that has a class named "Course".

- The class Course has the attributes course name, course code, class venue and credit hours, all are protected members.
- Set all these attributes with a parameterized constructor.
- Derive a class "Java Course" that has an attribute teacher name.
- Make a constructor and invoke the base class's parameterized constructor.
- Set the teacher's name in the constructor.
- The derived class has a function Display that displays all the details of the course and the derived class.
- In the main, display all the details.

QUESTION#2

Write a Java program that has a class named "Person".

- The class has a default constructor that displays "I am a person".
- The class has attributes name, age, nationality, address and CNIC.
- The class has an input function that prompts the user to enter all the details. For CNIC, the total number of digits should be exactly 13. If it's less than 13 or greater display an error message.
- The class also has a display function that displays all the details.

Derive a class Employee from Person.

- The class Employee has a default constructor that invokes the base class's constructor and displays "I am an Employee".
- The class has the attributes name of company, company's location (city), no of years worked.
- The class has an input function that prompts the user to enter all the details. It also has a display function that displays all the details.

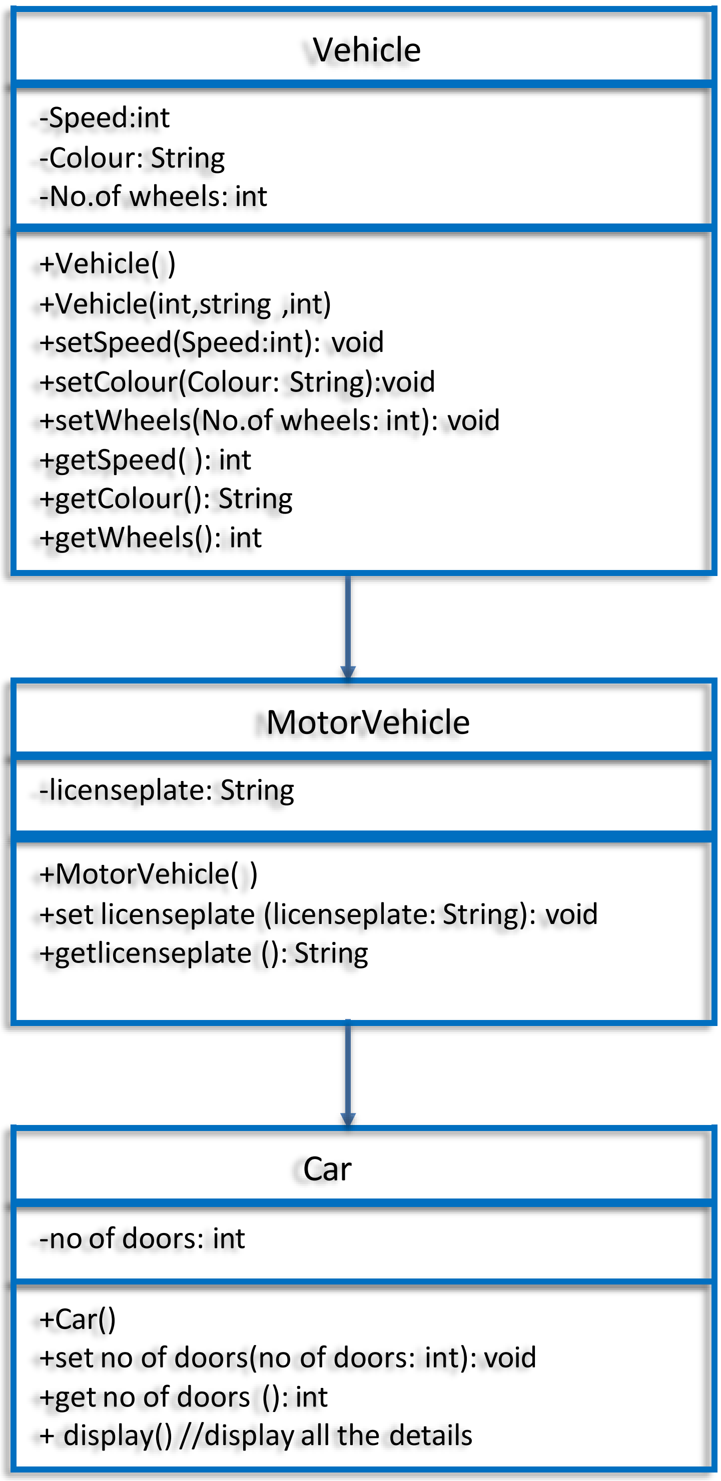
Derive a class Manager from Employee.

- The class Manager has a default constructor that invokes the base class's constructor and displays "I am a Manager".
- The class has an array that contains the names of employees who are working under the manager's supervision. Input atleast five employee's in the array from the user and display all these employee's too.

In the main program, call all the functions and display the details.

QUESTION#3

Implement the scenario given in the figure.


Vehicle
-Speed:int
-Colour: String
-No.of wheels: int
+Vehicle( )
+Vehicle(int,string ,int)
+setSpeed(Speed:int): void
+setColour(Colour: String):void
+setWheels(No.of wheels: int): void
+getSpeed( ): int
+getColour(): String
+getWheels(): int
MotorVehicle
-licenseplate: String
+MotorVehicle( )
+set licenseplate (licenseplate: String): void
+getlicenseplate (): String
Car
-no of doors: int
+Car()
+set no of doors(no of doors: int): void
+get no of doors (): int
+ display() //display all the details

QUESTION#4

Create classes for School, Department, and Teacher. Use aggregation to represent the relationship between a school and its departments, and between a department and its teachers. Write code to add teachers to departments and departments to the school. **To do this Create following classes:**

Teacher Class:

The Teacher class represents a teacher and has a single attribute, name, to store the teacher's name.

The constructor Teacher(String name) is used to initialize the name attribute.

There's a getName() method to retrieve the teacher's name.

Department Class:

The Department class represents a department in a school.

It has attributes for name, an array of Teacher objects (teachers), and a count (teacherCount) to keep track of the number of teachers in the department.

The maxTeachers parameter in the constructor specifies the maximum number of teachers a department can have.

The addTeacher(Teacher teacher) method adds a teacher to the department's array if there is space available.

The getTeachers() method returns the array of teachers in the department.

The getName() method returns the department's name.

School Class:

The School class represents a school and is similar in structure to the Department class.

It has attributes for name, an array of Department objects (departments), and a count (departmentCount) to keep track of the number of departments in the school.

The maxDepartments parameter in the constructor specifies the maximum number of departments a school can have.

The addDepartment(Department department) method adds a department to the school's array if there is space available.

The getDepartments() method returns the array of departments in the school.

The getName() method returns the school's name.

Main Method:

- In the main method, create instances of teachers, departments, and a school.
- create three teachers: mathTeacher1, mathTeacher2, and scienceTeacher1.
- create two departments: mathDept and scienceDept, specifying the maximum number of teachers they can have.
- add the teachers to their respective departments using the addTeacher method.
- create a school, school, specifying the maximum number of departments it can have.
- add the departments to the school using the addDepartment method.
- Finally, print out the school's details, including its name, departments, and teachers within each department. The code checks for null values in the arrays to handle cases where the maximum number of teachers or departments has been reached.

QUESTION#5

Imagine you are building a music library application in Java. You need to design a class structure to represent the library's contents, including songs, albums, and artists. Songs can belong to one or more albums, and artists can be associated with multiple songs and albums. Design a class structure using aggregation to model this relationship. **Create following classes:**

Artist Class:

The Artist class represents an artist with a single attribute, name, which stores the name of the artist.

The constructor Artist(String name) initializes the artist's name.

The getName() method allows you to retrieve the artist's name.

Album Class:

The Album class represents an album with two attributes: title to store the album title and songs as a list to hold the songs in the album.

The constructor Album(String title) initializes the album's title and initializes an empty list of songs.

The addSong(Song song) method allows you to add a song to the album.

The getSongs() method returns the list of songs in the album.

The getTitle() method retrieves the album title.

Song Class:

The Song class represents a song with two attributes: title to store the song title and artists as a list to hold the artists associated with the song.

The constructor Song(String title) initializes the song's title and initializes an empty list of artists.

The addArtist(Artist artist) method allows you to add an artist to the song.

The getArtists() method returns the list of artists associated with the song.

The getTitle() method retrieves the song title.

MusicLibrary Class:

In the MusicLibrary class's main method, create instances of artists, songs, and albums.

- create two artists: artist1 and artist2.
- create two songs: song1 and song2, associating each song with one or more artists using the addArtist method.
- create two albums: album1 and album2, adding the songs to each album using the addSong method.
- Finally, print the details of albums and songs, including their titles and associated artists.

QUESTION#6

Create a class named 'Rectangle' with two data members 'length' and 'width' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and width is used to initialize length and width of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as super(). Print the area and perimeter of a rectangle and a square.

QUESTION#7

- A library wants to organize its system by categorizing books such as Java, C, C++, etc. Implement a program that contains a base class called Books that will contain members such as book ID, book name, book author, ISBN and price. All are protected members.
- Derive one class from the base class and name it as “Category1”.
- The class has one data member that is the category.
- Make a parameterized constructor and invoke the base class’s constructor.
- Create a display function and display all the details of the books in Category1 (show at least 3 books details).
- In the main program, class the display function to view the details