Lab 05
Composition / Containership (Has-a relationship)

Objective:

The purpose of lab is to make students understand the concept of has-a relationship in object-oriented programming.

Activity Outcomes:

Students will be able to understand that complex objects can be modeled as composition of other objects.

Students will be able to implement programs related to composition.

Instructor Note:

As pre-lab activity, read Chapter 10 from the text book “Introduction to Java Programming”, Y. Daniel Liang, Pearson, 2019.

.

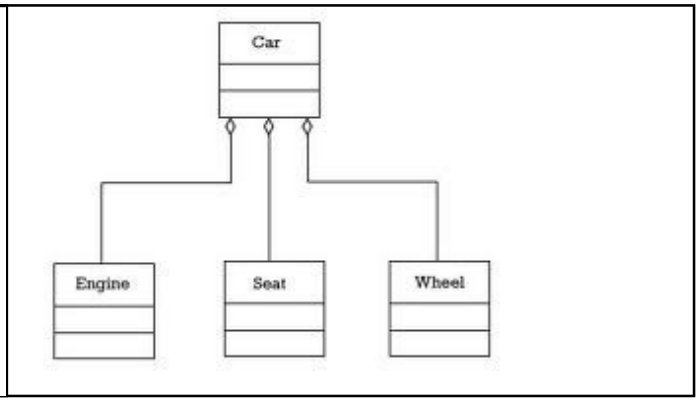
1) Useful Concepts

In real-life, complex objects are often built from smaller, simpler objects. For example, a car is built using a metal frame, an engine, some tires, a transmission, a steering wheel, and a large number of other parts. A personal computer is built from a CPU, a motherboard, some memory, etc. Even you are built from smaller parts: you have a head, a body, some legs, arms, and so on. This process of building complex objects from simpler ones is called composition (also known as object containership).

More specifically, composition is used for objects that have a —has-a relationship to each other. A car has-a metal frame, has-an engine, and has-a transmission. A personal computer has-a CPU, a motherboard, and other components. You have-a head, a body.

```
class wheel
{
}

class car
{
  wheel w1;
}
```



2) Solved Lab Activities (Allocated Time 45 min)

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
Activity 1	*20 mins*	*Medium*	*CLO-4*
Activity 2	*25 mins*	*Medium*	*CLO-4*

Activity 1:

Composition is about expressing relationships between objects. Think about the example of a manager. A manager has the properties e.g Title and club dues. A manager has an employment record. And a manager has an educational record. The

phrase "has a" implies a relationship where the manager owns, or at minimum, uses, another object. It is this "has a" relationship which is the basis for composition. This example can be programmed as follows:

Solution:

```
class studentRecord {

    private String degree;

    public studentRecord() {
    }

    public void setDegree(String deg) {
        degree = deg;

    }

    public String getDegree() {
        return degree;
    }

}

class employeeRecord {

    private int emp_id;
    private double salary;

    public employeeRecord() {
    }

    public void setEmp_id(int id) {
        emp_id = id;
    }

    public int getEmp_id() {
```

```
        return emp_id;
    }

    public void setSalary(int sal) {
        salary = sal;
    }

    public double getSalary() {
        return salary;
    }

}

class Manager {

    private String title;
    private double dues;
    private employeeRecord emp;
    private studentRecord stu;

    public Manager(String t, double d,
employeeRecord e, studentRecord s) {
        title = t;
        dues = d;
        emp = e;
        stu = s;
    }

    public void display() {

        System.out.println("Title is : " +
title);
        System.out.println("Dues are : " +
dues);

        System.out.println("Emplyoyee record is
as under:");
```

```
        System.out.println("EmployeeId is : " +
emp.getEmp_id());
        System.out.println("EmployeeId is : " +
emp.getSalary());

        System.out.println("Student record is as
under: ");

        System.out.println("Degree is : " +
stu.getDegree());
    }

}

public class Runner {

    public static void main(String args[]) {
        studentRecord s = new studentRecord();
        s.setDegree("MBA");
        employeeRecord e = new employeeRecord();
        e.setEmp_id(1);
        e.setSalary(25000);
        Manager m1 = new
Manager("financeManager", 5000, e, s);
        m1.display();
    }
}
```

Activity 2*:*

The program below represents an employee class which has two Date objects as data members.

Solution:

```
class Date {

    private int day;
```

```
    private int month;
    private int year;

    public Date(int theMonth, int theDay, int
theYear) {
        day = checkday(theDay);
        month = checkmonth(theMonth);
        year = theYear;
    }

    private int checkmonth(int testMonth) {
        if (testMonth > 0 && testMonth <= 12) {
            return testMonth;
        } else {
            System.out.println("Invalid month" +
testMonth + "set to 1");
            return 1;
        }
    }

    private int checkday(int testDay) {
        int daysofmonth[] = {0, 31, 28, 31, 30,
31, 30, 31, 31, 30, 31, 30, 31};

        if (testDay > 0 && testDay <=
daysofmonth[month]) {
            return testDay;
        } else if (month == 2 && testDay == 29
&& (year % 400 == 0 || (year % 4 == 0 && year %
100
                != 0))) {
            return testDay;
        } else {
            System.out.println("Invalid date" +
testDay + "set to 1");
        }
        return 1;
```

```
    }

    public int getDay() {
        return day;
    }

    public int getMonth() {
        return month;
    }

    public int getYear() {
        return year;
    }

    public void display() {
        System.out.println(day + " " + month + "
" + year);
    }

}

class employee {

    private String name;
    private String fname;
    private Date birthdate;
    private Date hiredate;

    employee() {

    }

    employee(String x, String y, Date
birthofDate, Date dateofHire) {
        name = x;
        fname = y;
        birthdate = birthofDate;
```

```
        hiredate = dateofHire;

    }

    public void setname(String x) {
        name = x;
    }

    public String getname() {
        return name;

    }

    public void setfname(String x) {
        fname = x;
    }

    public String getfname() {
        return fname;
    }

    public void setbirthdate(Date b) {
        birthdate = b;

    }

    public Date getbirthdate() {
        return birthdate;

    }

    public void sethiredate(Date h) {
        hiredate = h;
    }

    public Date gethiredate() {
        return hiredate;
```

```
    }

    public void display() {

        System.out.println("Name: " + name + "
Father Name: " + fname);
        birthdate.display();
        hiredate.display();

    }
}

public class Employrun {

    public static void main(String[] args) {
        Date b = new Date(1, 12, 1990);
        Date h = new Date(5, 6, 2016);
        employee e1 = new employee("xxx",
"yyyy", b, h);
        e1.display();
    }

}
```

3) Graded Lab Tasks (Allocated Time 2 Hrs 15 min.)

Note: The instructor can design graded lab activities according to the level of difficulty and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Lab Task 1

Create an Address class, which contains street#, house#, city and code. Create another class Person that contains an address of type Address. Give appropriate get and set functions for both classes. Test class person in main.

Lab Task 2

Create a class Book that contains an author of type Person (Note: Use the Person class created in the first exercise). Other data members are bookName and publisher. Modify the address of the author in runner class.

Lab Task 3

Design a class Point with two data members x-cord and y-cord. This class should have an arguments constructor, setters, getters and a display function.
Now create another class ―Line‖, which contains two Points ―startPoint‖ and ―endPoint‖. It should have a function that finds the length of the line.

Hint: formula is: sqrt((x2-x1)2 + (y2-y1)2) Create two line objects in runner and display the length of each line.

Lab Task 4

Create a class named Pizza that stores information about a single pizza. It should contain the following:

Private instance variables to store the size of the pizza (either small, medium, or large), the number of cheese toppings, the number of pepperoni toppings, and the number of ham toppings.
Constructor(s) that set all of the instance variables.
Public methods to get and set the instance variables.
A public method named calcCost( ) that returns a double that is the cost of the pizza. Pizza cost is determined by:
Small: $10 + $2 per topping Medium: $12 + $2 per topping Large: $14 + $2 per topping

public method named getDescription( ) that returns a String containing the pizza size, quantity of each topping.

Write test code to create several pizzas and output their descriptions. For example, a large pizza with one cheese, one pepperoni and two ham toppings should cost a total of $22.
Now Create a PizzaOrder class that allows up to three pizzas to be saved in an order. Each pizza saved should be a Pizza object. Create a method calcTotal() that returns the cost of order.
In the runner order two pizzas and return the total cost.