Lab05: Association and Aggregation

Designing and implementing Java programs that deal with:

1. **Association and**
2. **Aggregation**

1. Association

Consider an association of student register for a course. Create the student class as follows:

```
/*
 * Student Class
*/
public class Student {
      private String name;
      private String matrix;
      private String major;
      private Vector courseList; //declare vector of course
      Student(String n,String m,String maj){
            name=n;
            matrix=m;
            major=maj;
            courseList = new Vector();//initialize the vector
      }
      public String getName()
         { return name; }
      public void regCourse(Course course) //showing association
         {courseList.addElement(course); }
      public void printInfo()
      {
      System.out.println("\nSTUDENT NAME :"+ name);
      System.out.println("NUMBER OF COURSE(s) TAKEN :" +
                           courseList.size());
      System.out.println("LIST OF COURSE(s) TAKEN :");
            for(int i=0;i<courseList.size();i++){
                  Course c=(Course)courseList.elementAt(i);
                  System.out.println(c.getTitle());
            }
      }

 } // end of class Student
```

Create the course class as follows:

```
/*
 * Course Class
*/
public class Course {
    private String courseTitle;
    private String courseCode;

      Course(String title,String code)
      {     courseTitle = title;
            courseCode = code;
      }
      public String getTitle()
      {
            return courseTitle;
      }
}//end of Cource class
```

Create the MyAppliction class according to the following code snippet.

```
/*
 *  MyApplcation Class
 */
public class MyApplication {
    public static void main(String [] args){
      Course cs1 = new Course("OOP","CSC-210");
      Course cs2 = new Course("DATA STRUCTURE","CSC-320");
      Student s1 = new Student("Tariq","DC0021","Software
                                                  Engineering");
      Student s2 = new Student("Sarmad","DC0022","Multimedia");
      Student s3 = new Student("Wasim","DC0023","Information Systems");
      s1.regCourse(cs1);
      s2.regCourse(cs1);
      s2.regCourse(cs2);
      s3.regCourse(cs1);

      s1.printInfo();
      s2.printInfo();
      s3.printInfo();
    }
}
```

You have noticed:

How the above code adds course objects into student class?

Now

Use **ArrayList** instead of Vector to add and retrieve course objects into Student class.

2. Aggregation

Consider an example of aggregation of address. Create the address class as follows:

```
/*
 * Addresss class
 */
public class Address {
	private String	streetAddress,
			city,
			state;
	private long	zipCode;
	public Address (String street, String town, String st,long zip)
	{	streetAddress = street;
		city = town;
		state = st;
		zipCode = zip;
	}

	public String toString()
	{
		String result;
		result = streetAddress + "\n";
		result += city + ", " + state + " " + zipCode;
		return result;
	}
}
```

Create a Student class that have the fields homeAddress and schoolAddress of type class Address.

```
/*
 * Student2 class
 */
public class Student2
{
    private String firstName, lastName;
    private Address homeAddress, schoolAddress;
    public Student2 (String first, String last, Address home, Address school)
      {
            firstName = first;
            lastName = last;
            homeAddress = home;
            schoolAddress = school;
      }
      public String toString()
      {
            String result;
            result = firstName + " " + lastName + "\n";
            result += "Home Address:\n" + homeAddress + "\n";
            result += "School Address:\n" + schoolAddress + "\n";
            return result;
      }
}
```

toString()

Java's Object class provides a method called toString() that returns the string representation of the object on which it is called. You can create your own toString() method by overriding toString() method to printout your own objects as it is done above.

Modify the MyAppliction class according to the following code snippet and rename it as MyApplication2.

```
/*
 * MyApplication2 class
 */
public class MyApplication2 {
      public static void main(String[] args)
      {
      Address school = new Address ("Govt. High School", "Defense View",
                                                    "Karachi", 72750);
      Address home1 = new Address ("DD-11","Defense View", "Karachi", 72750);
      Student2 s1 = new Student2 ("Luqman", "Abdul Hamid",
                                          home1, school);
      Address home2 = new Address ("F-12", "DHA Phase-V","Karachi", 7250);
      Student2 s2 = new Student2 ("Norain", "Yusouf",home2,
                                          school);
      System.out.println (s1);
      System.out.println ();
      System.out.println (s2);
      }
}
```

Exercises

Exercise 1(a) *(Time.java, Passenger.java, Flight.java,)*

Association & Aggregation

Based On the class diagram given below, you are required to write complete program for Flight's class, Time's class and Passenger's class with the concept of association and aggregation.Functions information also been given in the table below . Program should display information supplied to flight object.

Class Diagram

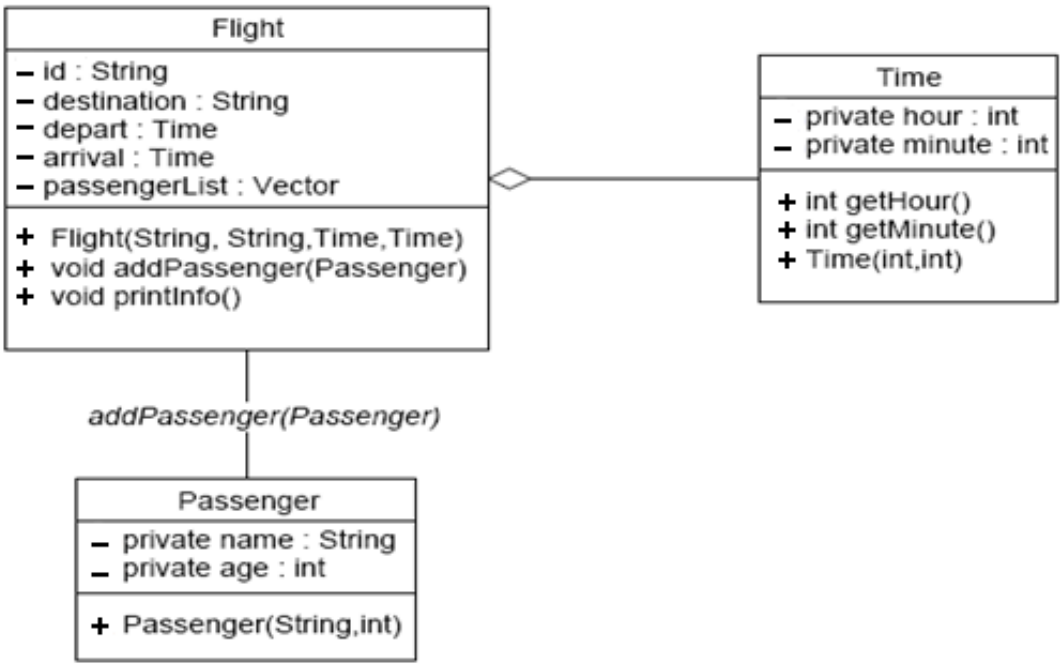


Table of Methods for Flight Class

Method	**Description**
addPassenger(Passenger)	This method will add Passenger's object to vector passengerList.
printInfo()	This method will display all flight information namely ID (Flight number), destination, departure time, arrival time and number of passengers. For Example: Flight no : PK-303 Destination : Lahore Departure : 8:10 Arrival : 9:00 Number of passenger :2
getHour()	This method will return the value of attribute **hour**
getMinute()	This method will return the value of attribute **minute**

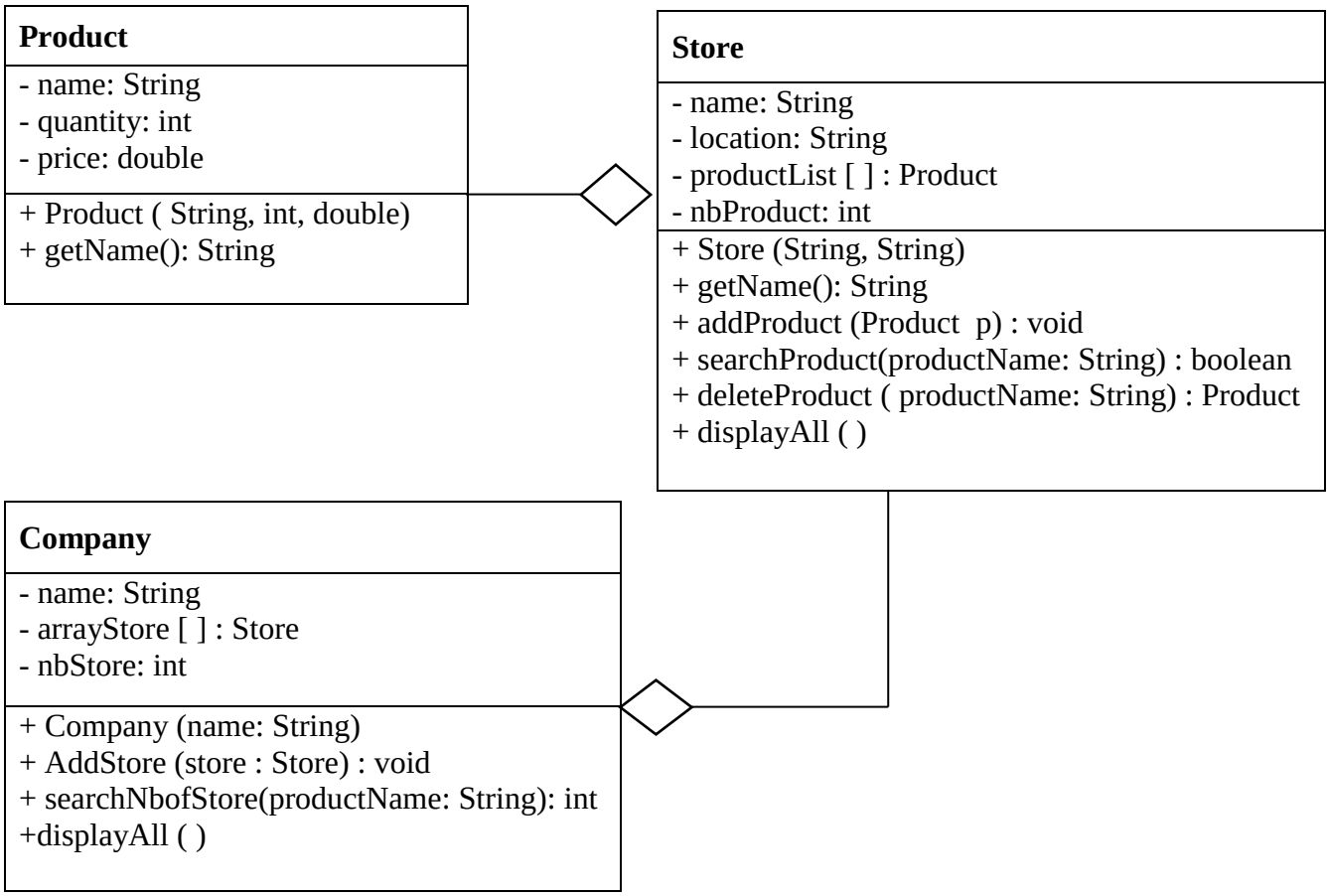
Exercise 1(b) *(FlightTester.java)*

Use the above classes to create the objects in the FlightTester class and call methods:

```
/*
 * FlightTester class
 */
public class FlightTester {
      public static void main(String arg[])
      {
      Time dep=new Time (8,10);
      Time arr=new Time (9,00);
      Flight f = new Flight("PK-303","Lahore",dep,arr);
      Passenger psg1= new Passenger("Umair", 30);
      Passenger psg2= new Passenger("Manzoor", 44);
      f.addPassenger(psg1);
      f.addPassenger(psg2);
      f.printInfo();
      }
}
```

Exercise 2(a) (Product.java, Store.java, Company.java)

A company manages many stores. Each Store contains many Products. The UML diagram of this compnay system is represented as follow.



Class Store:

Attribute: name, location, productList, nbProduct
Constructor: Store (name: String, location: String):
Method:

addProduct() that adds a new product. Maximum 100 products can be added.

searchProduct() that accepts the name of product and return **True** if exist, **False** otherwise.

deleteProduct() that accepts the name of product that has to be deleted and returns the deleted object.

displayAll() prints the name of products available in store.

Class Company:

Attribute: name, arrayStore, nbStore
Constructor: Company (name: string):
Method:

addStore() that adds a new Store. Maximum 10 stores can be added.

searchNbofStore() that accepts the name of product and returens the number of stores containing the product.

displayAll() prints the name of stores belongs to company.

Exercise 2(b) (TestCompany.java)

Implement Product, Store and Company classes and use the following class to test.

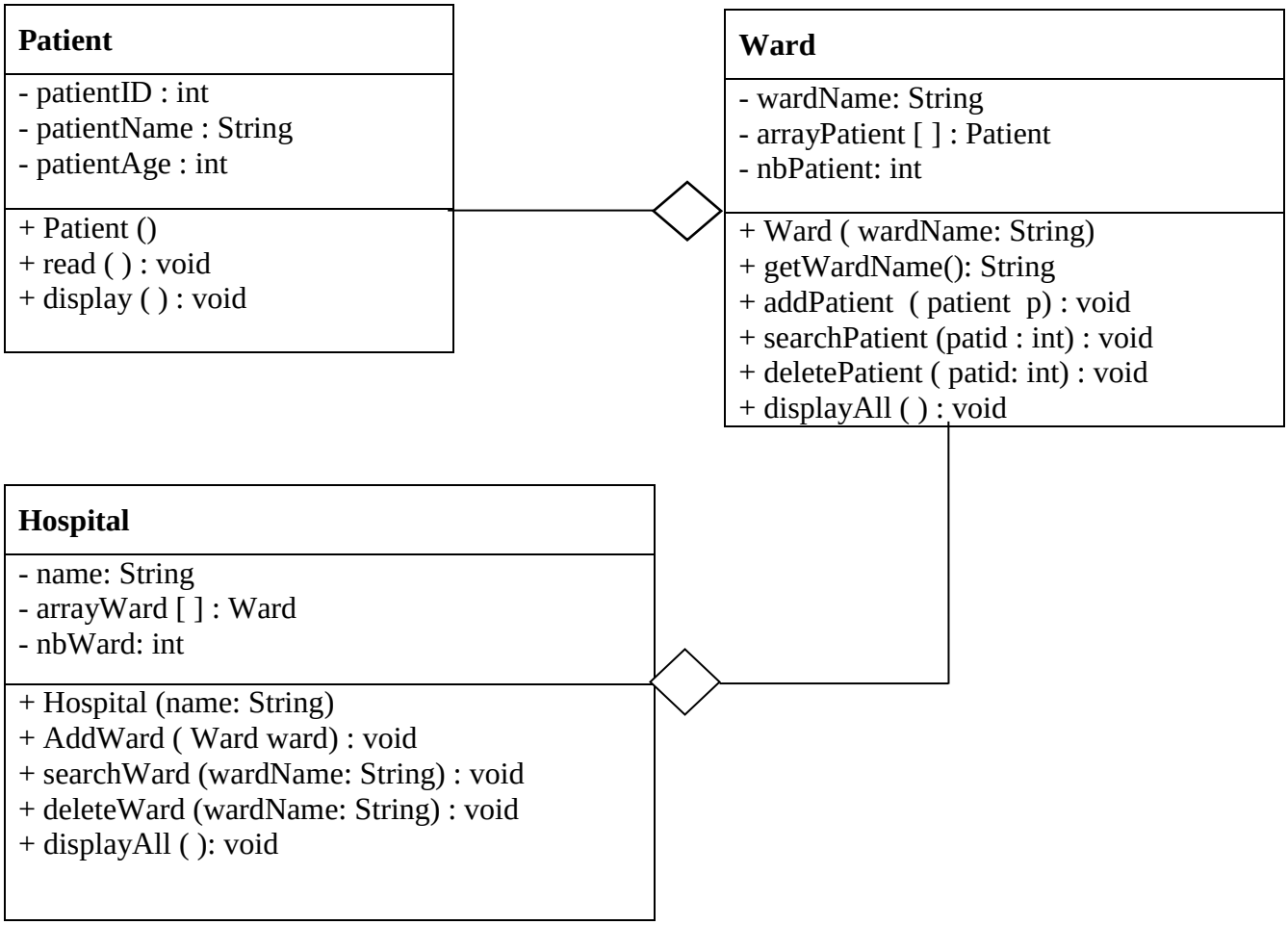
```
public class TestCompany {
    public static void main(String [] args){
        Product p1 = new Product("TV",4,34000);
        Product p2 = new Product("Bicycle", 4, 5500);
        Product p3 = new Product("Fridge", 3,70000);

        Store s1 = new Store("Makro", "Karachi");
        Store s2 = new Store("Hypermart","Karachi");
        s1.addProduct(p1);
        s1.addProduct(p2);
        s1.addProduct(p3);
        s1.displayAll();
        Product tempProduct = s1.deleteProduct("Bicycle");
        if (tempProduct!=null)
            System.out.println("Product "+tempProduct.getName()+ " is
            deleted");
        else
            System.out.println("There is no product to delete");
        s1.displayAll();
        s2.addProduct(p1);
        s2.addProduct(p2);
        s2.addProduct(p3);
        s2.displayAll();
        Company c1 = new Company("Unilever");
        c1.addStore(s1);
        c1.addStore(s2);
        c1.displayAll();
        int n= c1.searchNbofStore("TV");
        System.out.println("Number of stores have TV "+n);

    }
}
```

Exercise 3(a) (Patient.java, Ward.java, Hospital.java)

Consider the following class diagram:



Class Patient:
Attribute: patientID, patientName, patientAge
Constructor: Patient( )
Method:

read ( ): this method reads the attributes of the patient from keyboard.
display ( ) : Displays patient id name and age

Class Ward
Attribute: wardName, arrayPatient, nbPatient
Constructor: Ward (wardName: String)
Methods:

addPatient (Patient p): this method adds the patient to array
searchPatient (patid : int) : this method should search in ward for a particular id and display the patient details.
deletePatient (patid: int) : this method should delete the patient from the ward.
displayAll( ): this method prints the attributes of all the objects that belong to patient

Class Hospital:
The class Hospital has the same functionality as Ward.

Translate into java code patient, ward and hospital classes.

Exercise 3(b) (TestHospital.java)

Use the TestHospital class to test the above classes.

```
public class TestHospital {
    public static void main(String [] args){
        Patient p1 = new Patient();
        p1.read();
        Patient p2 = new Patient();
        p2.read();
        Patient p3 = new Patient();
        p3.read();
        Ward w1 = new Ward("Childern");
        w1.addPatient(p1);
        w1.addPatient(p2);
        w1.displayAll();
        w1.searchPatient(2);
        w1.deletePatient(2);
        w1.displayAll();
        Ward w2 = new Ward("General");
        w2.addPatient(p2);
        w2.addPatient(p3);
        w2.displayAll();
        Hospital h1 = new Hospital("Jinnah");
        h1.addWard(w1);
        h1.addWard(w2);
        h1.displayAll();
        h1.searchWard("General");
        h1.deleteWard("Childern");
        h1.displayAll();
   }
}
```

Project

From last week you have found the detailed Attributes, Constructors and Methods. Now find the relationships between the classes and draw class diagram according to their relationships and also write the necessary code for each class.