**Lab-Report**

**Exp. Name :** Association and Aggregation UML Code

**Course Code : CSE-326**

**Course Title : System Analysis and Design Lab**

**Submitted To: Sputa Richard Philip**

**Senior Lecture**

**City University**

**Department of computer Science and Engineering**

**Faculty of Science and Engineering**

**Submitted By:**

**ID :171442560**

**Name :Md. Rasel Miah**

**Program :CSE(Eve)**

**Batch : 44th**

**Semester :Spring**

**Association**

Association is a relationship between two separate classes that establishes through their objects. Each object have their own life-cycle and there is no owner. Association can be one-to-one, one-to-many, many-to-one, many-to-many.

Let's take an example of Teacher and Student. Multiple students can associate with single teacher and single student can associate with multiple teachers, but there is no ownership between the objects and both have their own life-cycle. Both can be created and deleted independently.

Teacher.java:

import java.util.ArrayList;

import java.util.List

public class Teacher {

private final String name;

private final List<Student> students = new ArrayList<>();

// teacher name

Teacher(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void addStudent(Student student) {

student.addTeacher(this);

this.students.add(student);

}

public List<Student> getStudents() {

return students;

}

public void print() {

System.out.println("Teacher " + this.name + "'s students are:");

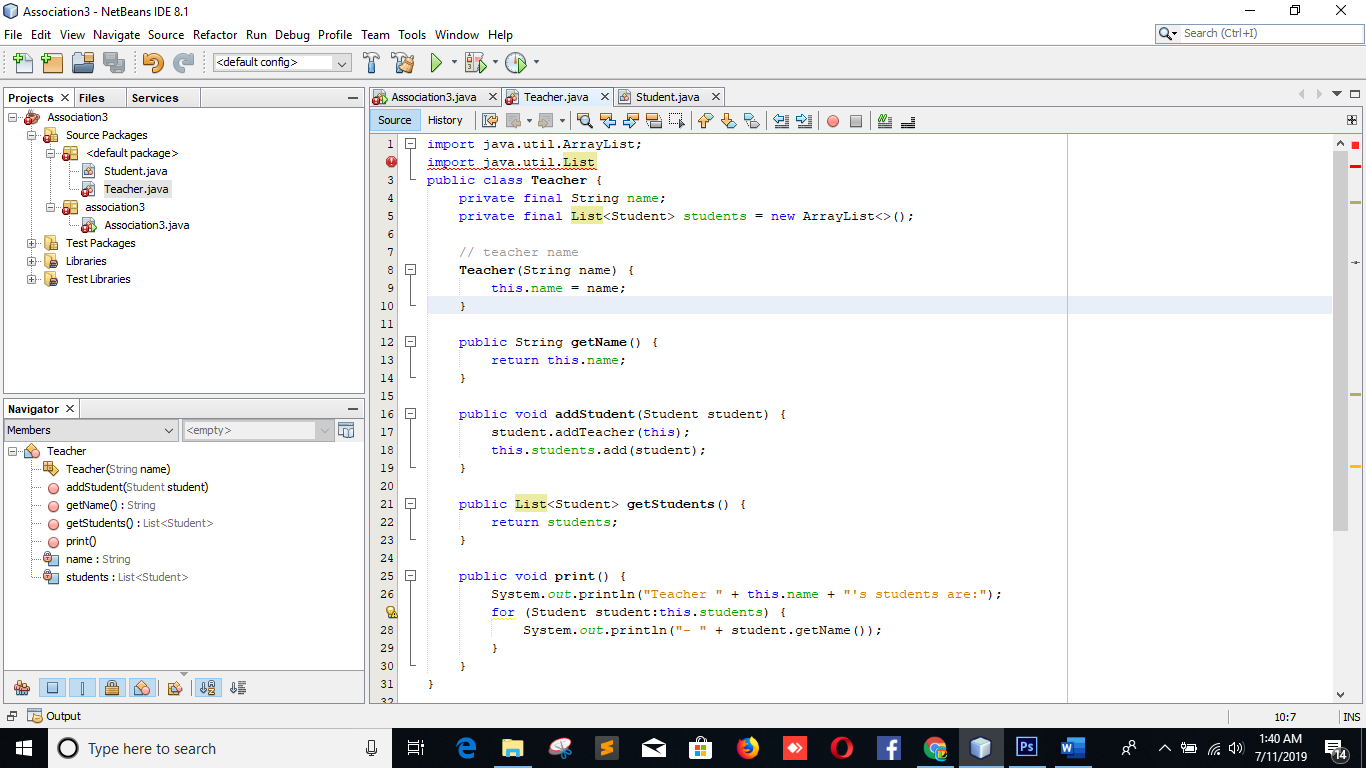
for (Student student:this.students) {

System.out.println("- " + student.getName());

}

}

}



Student.java:

import java.util.ArrayList;

import java.util.List;

public class Student {

private final String name;

private final List<Teacher> teachers = new ArrayList<>();

// student name

Student(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void addTeacher(Teacher teacher) {

this.teachers.add(teacher);

}

public List<Teacher> getTeachers() {

return teachers;

}

public void print() {

System.out.println("Student " + this.name + "'s teachers are:");

for (Teacher teacher:this.teachers) {

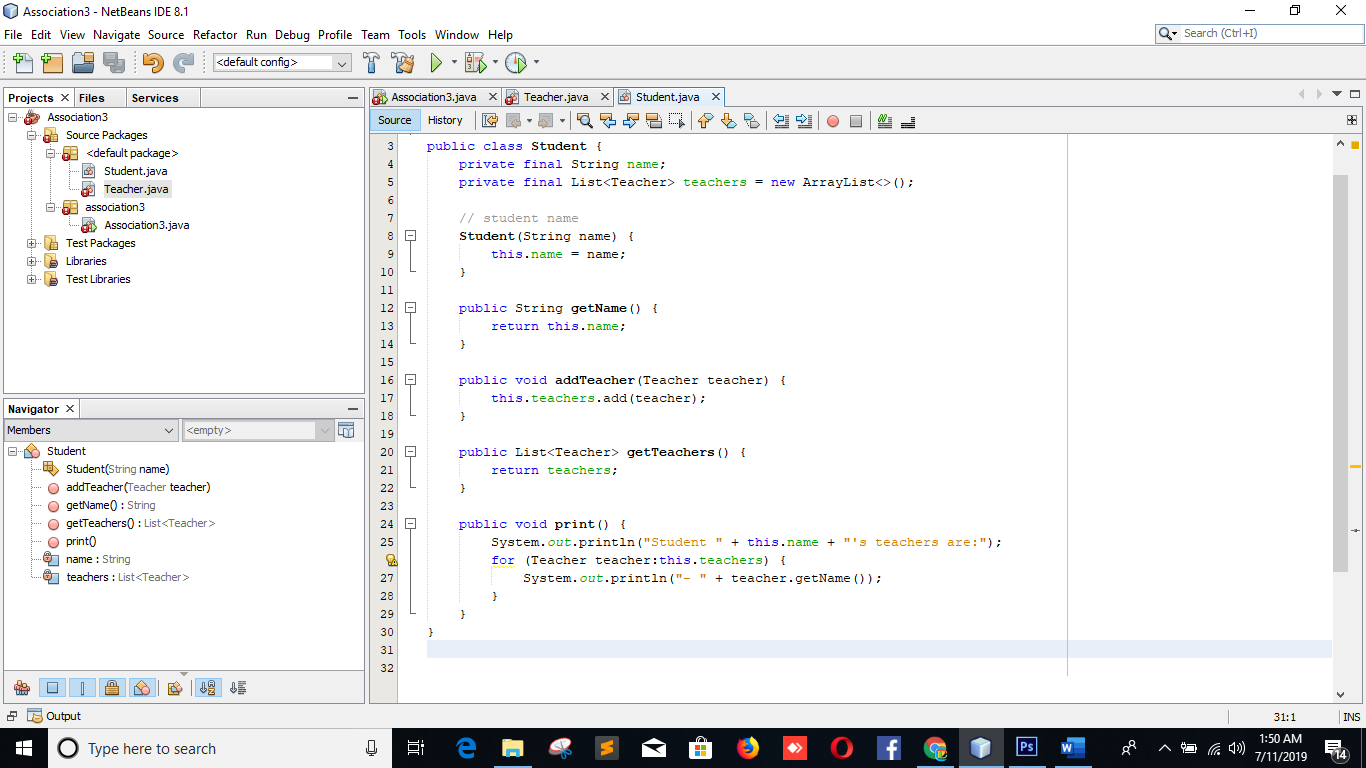
System.out.println("- " + teacher.getName());

}

}

}

|  |  |
| --- | --- |
|  |  |



Association.java

package association3;

public class Association3 {

public static void main(String[] args) {

Teacher teacher1 = new Teacher("Onizuka");

Teacher teacher2 = new Teacher("Fuyutsuki");

Student student1 = new Student("Nomura");

Student student2 = new Student("Aizawa");

Student student3 = new Student("Yoshikawa");

Student student4 = new Student("Uehara");

teacher1.addStudent(student1);

teacher1.addStudent(student2);

teacher1.addStudent(student3);

teacher2.addStudent(student2);

teacher2.addStudent(student3);

teacher2.addStudent(student4);

teacher1.print();

teacher2.print();

student1.print();

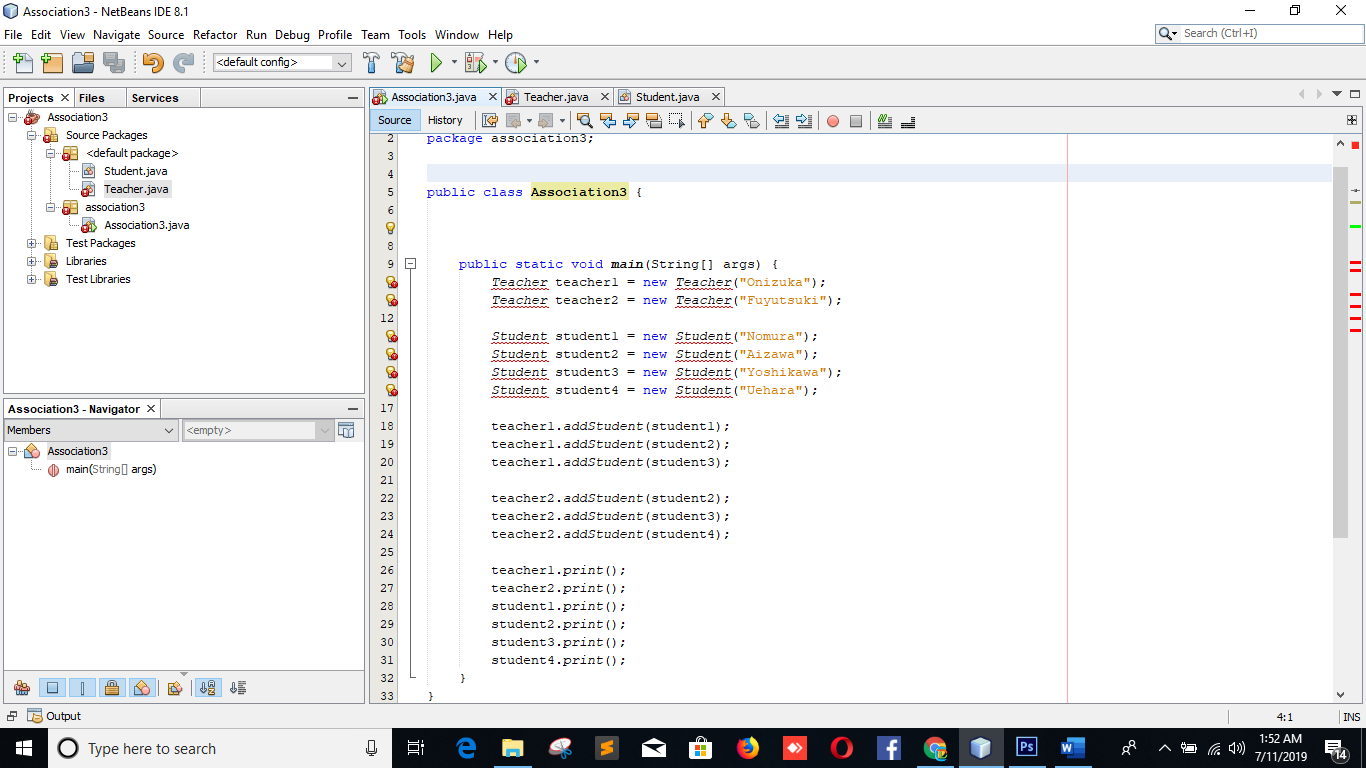
student2.print();

student3.print();

student4.print();

}

}



Aggregation:

**Aggregation** is a specialized form of Association where all objects have their own life cycle, where the child can exist independently of the parent. Aggregation is also called a “**Has-a**” relationship.

Let's take an example of Supervisor and Subordinate. An employee (as a subordinate) can not belong to multiple supervisors, but if we delete the supervisor, the employee object (subordinate) will *not* be destroyed. We can think about it as a “**has-a**” relationship.

Employee.java

import java.util.ArrayList;

import java.util.List;

public class Employee {

rivate final String name;

private Employee supervisor;

private final List<Employee> subordinates = new ArrayList<>();

// teacher name

Employee(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public Employee getSupervisor() {

return supervisor;

}

public void setSupervisor(Employee supervisor) {

this.supervisor = supervisor;

supervisor.subordinates.add(this);

}

public void print() {

System.out.println("Employee " + this.name + "'s supervisor is:" +

(this.supervisor==null?"N.A.":supervisor.getName()));

System.out.println("Employee " + this.name + "'s subordinates are:");

for (Employee employee:this.subordinates) {

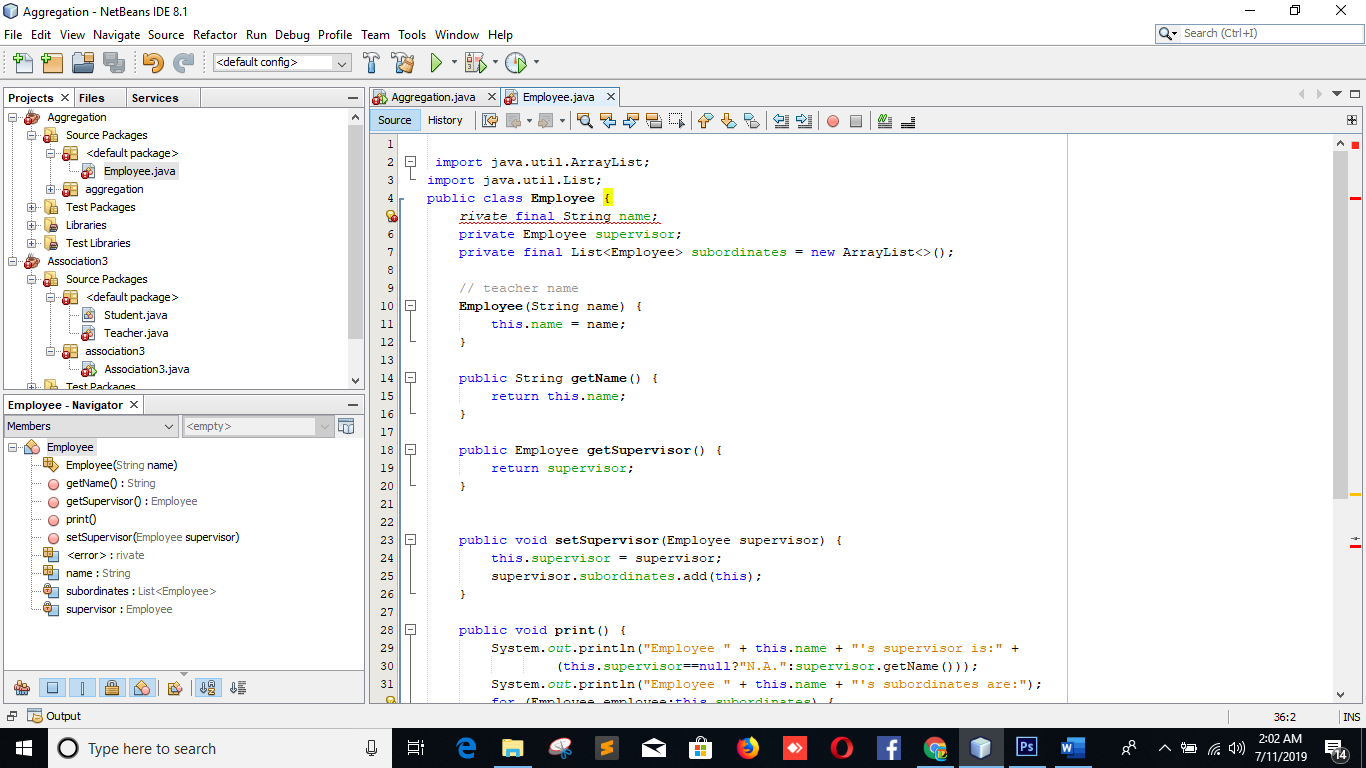
System.out.println("- " + employee.getName());

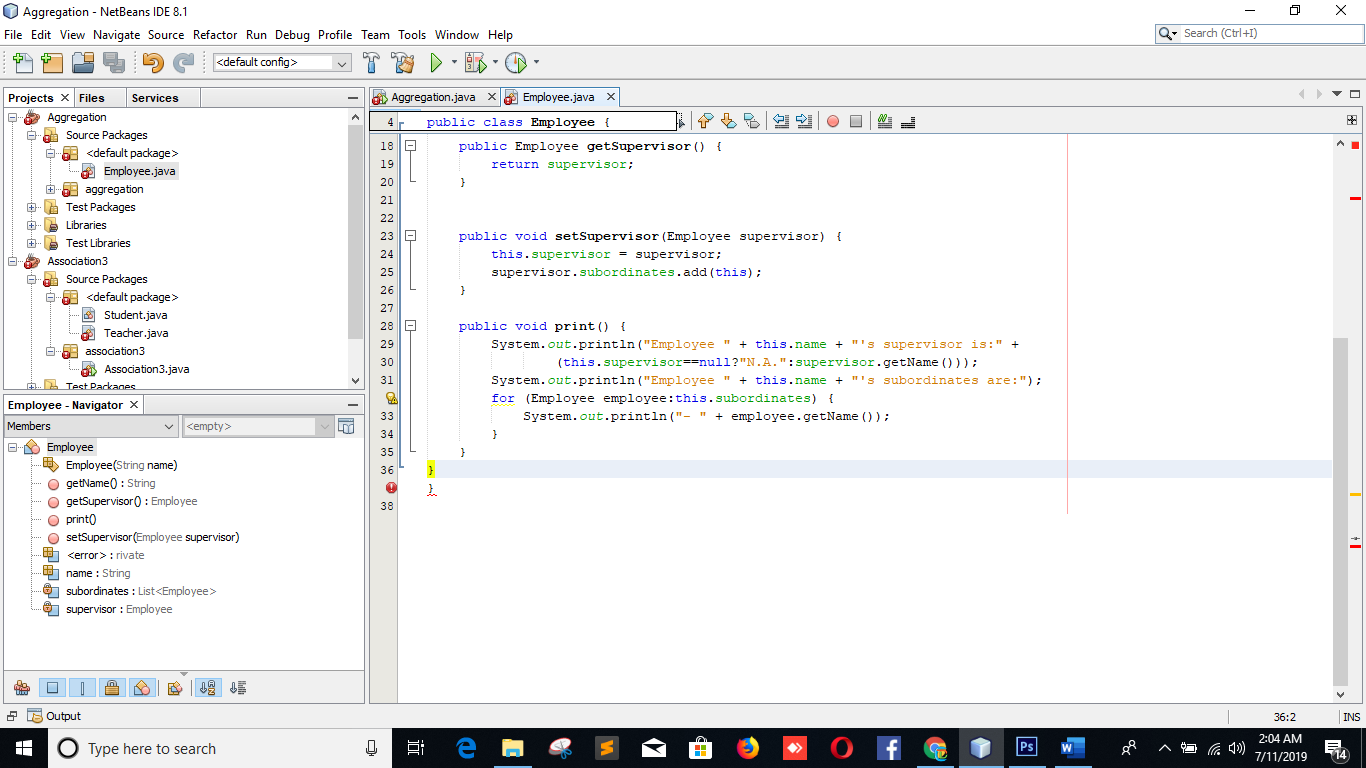
}

}

}

}





Aggregation

package aggregation;

public class Aggregation {

public static void main(String[] args) {

Employee employee1 = new Employee("Systrom");

Employee employee2 = new Employee("Krieger");

Employee employee3 = new Employee("Riedel");

Employee employee4 = new Employee("Sweeney");

Employee employee5 = new Employee("Zollman");

Employee employee6 = new Employee("Cole");

Employee employee7 = new Employee("Hochmuth");

Employee employee8 = new Employee("McAllister");

employee3.setSupervisor(employee1);

employee4.setSupervisor(employee1);

employee5.setSupervisor(employee1);

employee6.setSupervisor(employee2);

employee7.setSupervisor(employee2);

employee8.setSupervisor(employee2);

employee1.print();

employee2.print();

employee3.print();

employee8.print();

}

}

