Object Oriented Programming – SCJ2153

Association

Associate Prof. Dr. Norazah Yusof

Association

Association represents a general binary relationship that describes an activity between two classes.


```
public class Student {
  /** Data fields */
  private Course[]
    courseList;

  /** Constructors */
  /** Methods */
}
```

```
public class Course {
  /** Data fields */
  private Student[]
    classList;
  private Faculty faculty

  /** Constructors */
  /** Methods */
}
```

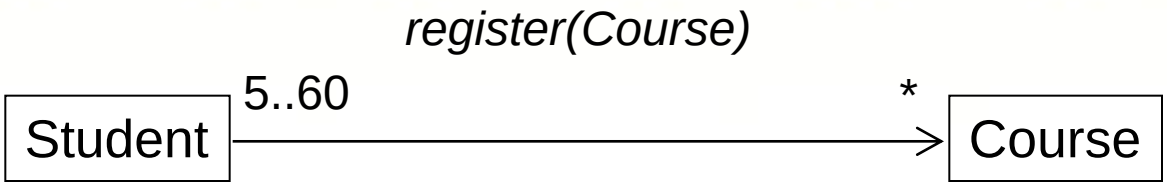
```
public class Faculty {
  /** Data fields */
  private Course[]
    courseList;

  /** Constructors */
  /** Methods */
}
```

An association is usually represented as a data field in the class.

Association

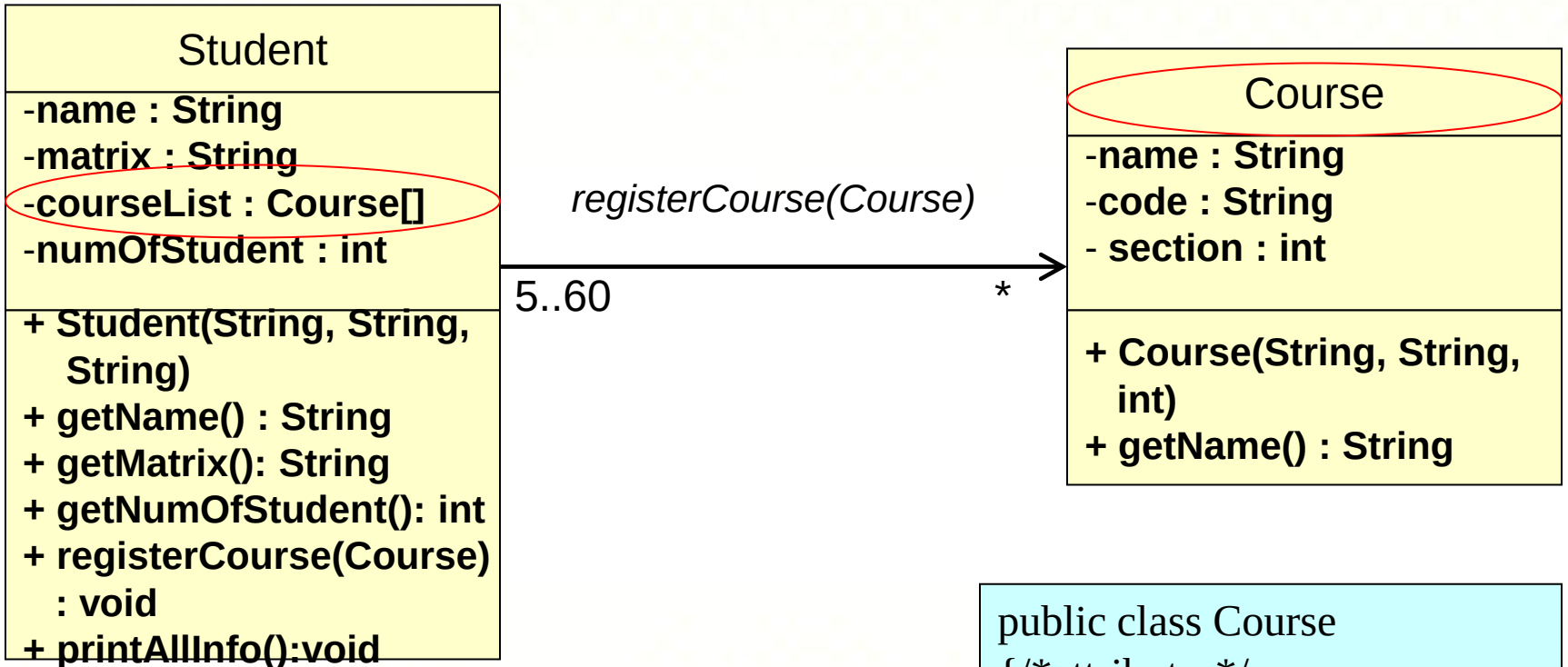
- Example: Student register course.



- The arrow is optional and specifies navigability.
- No arrow implies bidirectional navigability

Association

- An association is usually represented as a data field in the class.

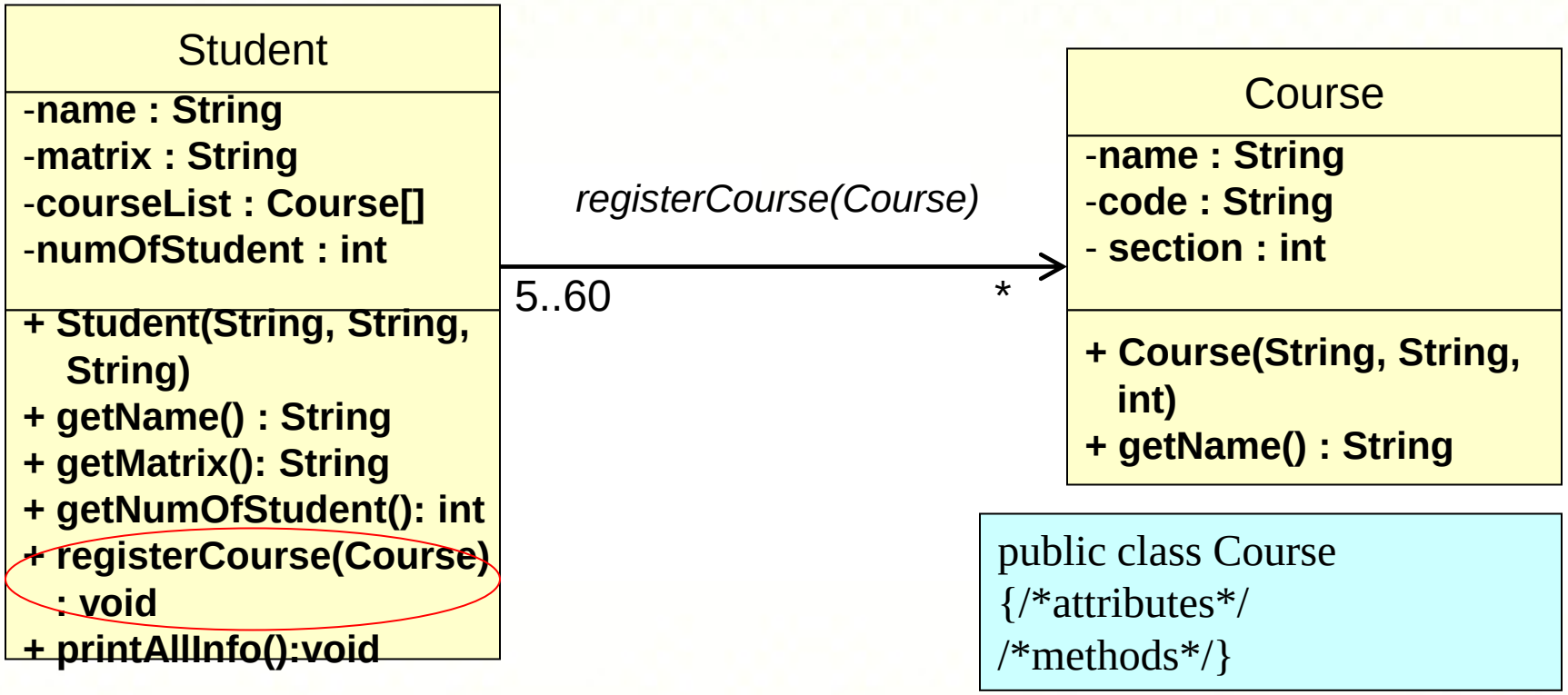



```
public class Student
{ /*attributes*/
  private Course[] courseList;
/*methods*/}
```

```
public class Course
{/*attributes*/
/*methods*/}
```

Association

- The relationship allows objects to call methods in other objects.

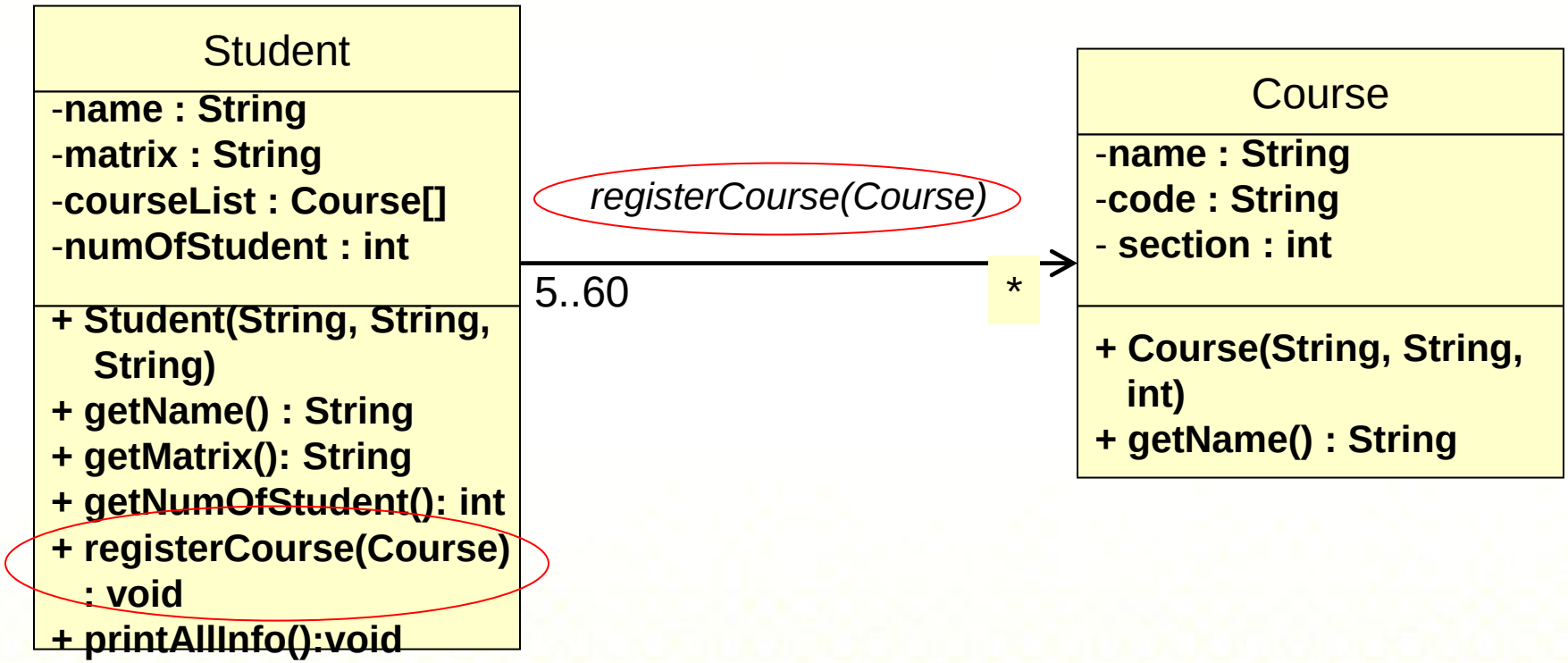



```
public class Course
{/*attributes*/
/*methods*/}
```

```
public class Student
{ /*attributes*/
  private Course[] courseList;
/*methods*/}
```

Association

- This is the same as an object sending another object a message.
- Therefore, it is implemented as object references.



Association Example 1: Student register Course

```
 1 import java.util.*;
 2 public class TestAssociate{
 3   public static void main(String args[]){
 4    Course cs1 = new Course("OOP","SCP3103",3);
 5    Course cs2 = new Course("TP1","SCJ1013",3);
 6    Course cs3 = new Course("TP2","SCJ1213",3);
 7    Course cs4 = new Course("KP","SCP2113",3);
 8
 9   Student s1 = new Student ("ALI","AC1234","2SCS");
10    s1.registerCourse(cs1);
11    s1.registerCourse(cs2);
12    s1.printAllInfo();
13    System.out.println();
14
15   Student s2 = new Student ("AHMAD","AC1122","3SCK");
16    s2.registerCourse(cs1);
17    s2.registerCourse(cs3);
18    s2.registerCourse(cs4);
19    s2.printAllInfo();
20     }
21 }
```

Association Example 1: Student register Course (Student class)

```
1  public class Student {
2    private String name;
3    private String matrix;
4
5    private Course[] courseList;          //showing association
6    private int numOfCourse;
7    public Student(String n,String m,String c){
8     name=n;
9     matrix=m;
10
11    courseList = new Course[10];          //showing association
12   }
13   public String getName() {
14      return name;
15   }
16
17   public String getMatrix() {
18      return matrix;
19   }
```

Association Example 1: Student register Course (Student class – cont.)

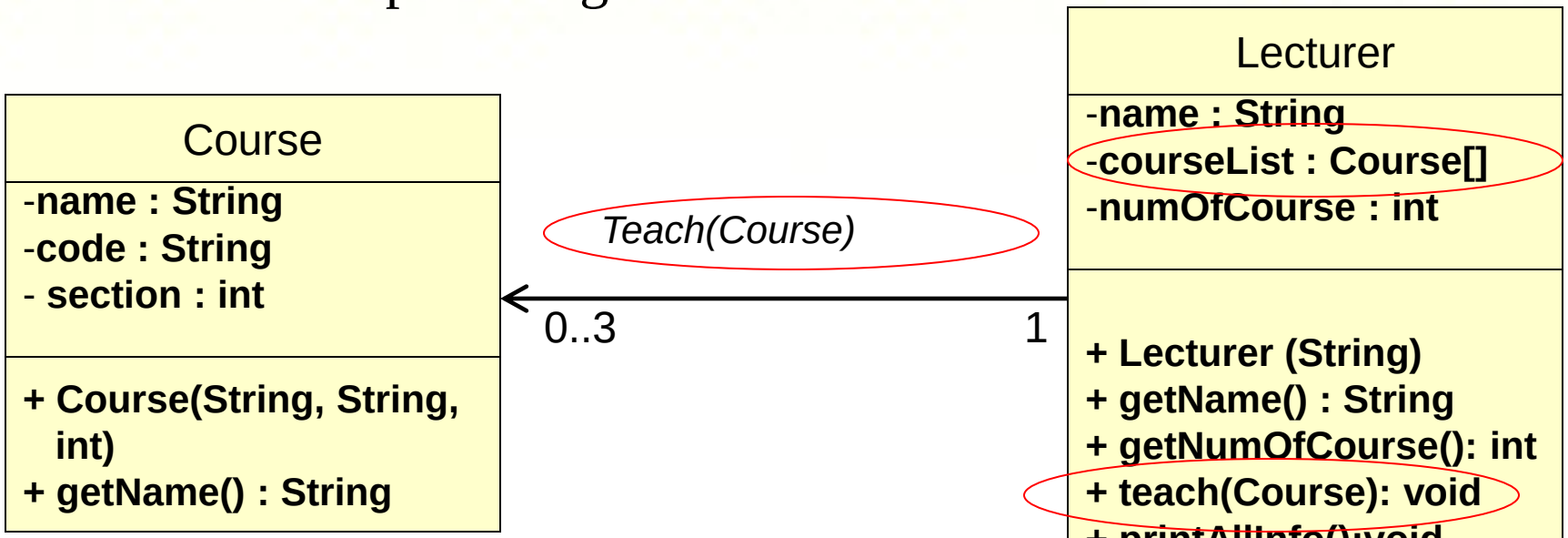
```
21  public void registerCourse(Course cs)   //showing association
22  {     courseList[numOfCourse] = cs;
23        numOfCourse++;
24    }
25
26    public int getNumOfCourse() {
27       return numOfCourse;
28    }
29    public void printAllInfo() {
30      System.out.println("\nSTUDENT  NAME          :"+name);
31      System.out.println("NUMBER OF SUBJECT(s) TAKEN :" +
32         numOfCourse);
33     System.out.println("LIST OF SUBJECT(s) TAKEN   :");
34      for(int i=0;i<numOfCourse;i++) {
35          Course s=(Course)courseList[i];
36          System.out.println((i+1) + ". " + s.getName());
37      }
38    }
39  }
```

Association Example 1: Student register Course (Course class)

```
1  public class Course {
2    private String name;
3    private String code;
4    private int section;
5    public Course(String n,String c,int s){
6       name = n;
7       code = c;
8       section =s;
9     }
10
11    public String getName() {
12          return name;
13    }
14 }
```

Association Example 2: Lecturer teach Course

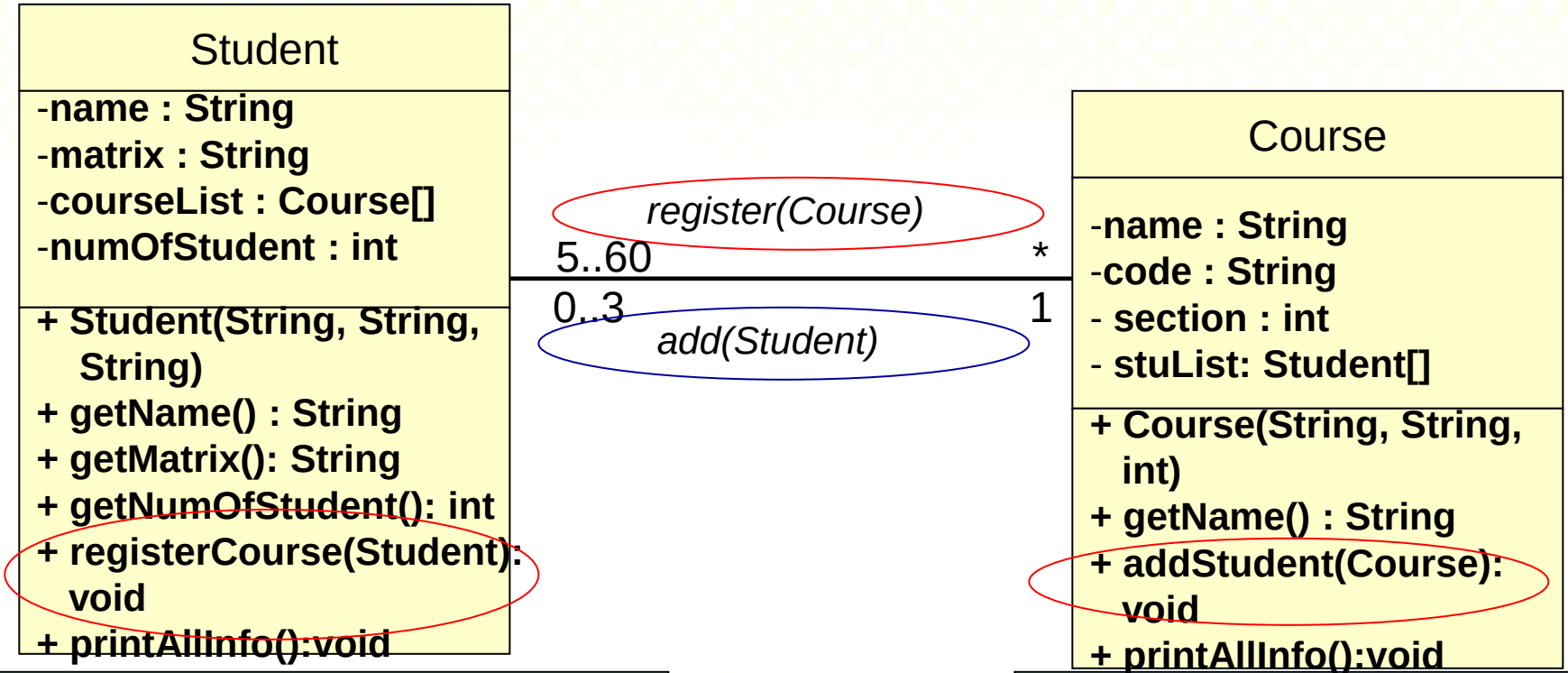
- Write the classes and test the classes to show association relationships among them.




```
public class Course
{/*attributes*/
/*methods*/}
```

```
public class Lecturer
{ /*attributes*/
  private Course[] courseList;
/*methods*/}
```

Association Example 3: Bidirectional Relationship Student register Course and Course add Students




```
public class Student
{ /*attributes*/
  private Course[] courseList;
/*methods*/}
```

```
public class Course
{/*attributes*/
 private Student[] stuList;
/*methods*/}
```

Association Example 3: Bidirectional Relationship Student register Course and Course add Students

```
 1 import java.util.*;
 2 public class TestAssociate2 {
 3   public static void main(String args[]){
 4    Course1 cs1 = new Course1("OOP","SCP3103",3);
 5    Course1 cs2 = new Course1("TP1","SCJ1013",3);
 6    Course1 cs3 = new Course1("TP2","SCJ1213",3);
 7    Course1 cs4 = new Course1("KP","SCP2113",3);
 8    Student1 s1 = new Student1("Ali","AC0021","2SCK");
 9     Student1 s2 = new Student1("Abu","AC0022","3SCK");
10     Student1 s3 = new Student1("Ben","AC0023","3SCS");
11     cs1.addStudent(s1);
12     cs1.addStudent(s2);
13     cs1.addStudent(s3);
14     cs2.addStudent(s2);
15     cs3.addStudent(s2);
16     cs1.printAllInfo();
17     cs2.printAllInfo();
18     s1.printAllInfo();
19     s2.printAllInfo();
20     s3.printAllInfo();
21    }
22 }
```

Association Example 3: Bidirectional Relationship Student register Course and Course add Students

```
1  public class Student1 {
2    private String name;
3    private String matrix;
4    private String major;
5    private Course1[] courseList; //showing association
6    private int numOfCourse;
7    public Student1(String n,String m,String j){
8     name=n;
9     matrix=m;
10    major = j;
11    courseList = new Course1[4]; //showing association
12   }
13   public String getName() {
14      return name;
15   }
16   public String getMatrix() {
17      return matrix;
18   }
19
```

Association Example 3: Bidirectional Relationship Student register Course and Course add Students

```
21  public String getMajor() {
22        return major;
23    }
24    public void register(Course1 cs)  {      //showing association
25       courseList[numOfCourse] = cs;
26       numOfCourse++;
27    }
28    public int getNumOfCourse() {
29       return numOfCourse;
30    }
31    public void printAllInfo() {
32      System.out.println("\nSTUDENT  NAME              :"+name);
33      System.out.println("NUMBER OF SUBJECT(s)
34  TAKEN :"+numOfCourse);
35     System.out.println("LIST OF SUBJECT(s) TAKEN    :");
36      for(int i=0;i<numOfCourse;i++) {
37          Course1 s=(Course1)courseList[i];
38          System.out.println((i+1) + ". " + s.getName());
39      }
40    }
41  }
```

Association Example 3: Bidirectional Relationship Student register Course and Course add Students

```
 1  public class Course1 {
 2    private String name;
 3    private String code;
 4    private int section;
 5    private Student1[] stuList; //showing association
 6    private int numOfStudent;
 7    public Course1(String n,String c,int s){
 8          name = n;
 9          code = c;
10          section =s;
11          stuList = new Student1[60];
12     }
13     public String getName()     {
14          return name;
15     }
16       public void addStudent(Student1 st){    //showing association
17        stuList[numOfStudent] = st;
18        numOfStudent++;
19        st.register(this);
20     }//addStudent
```