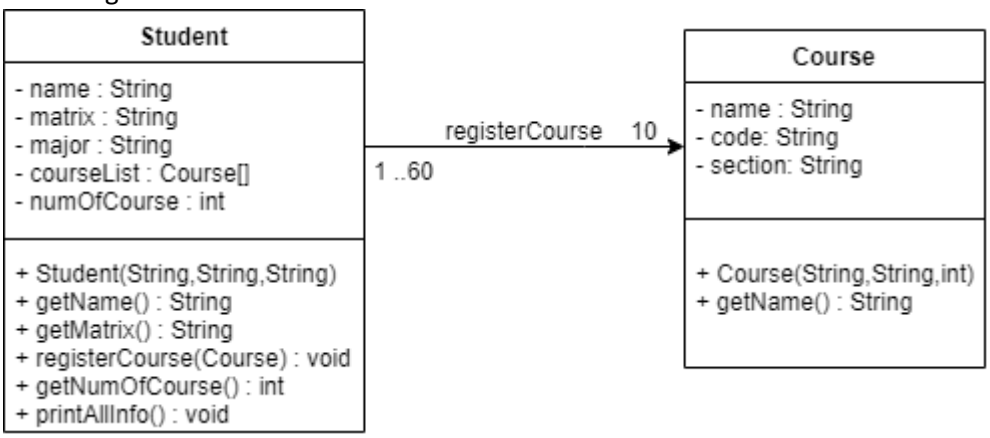
Exercise 1: Association Relationship Question 2

i. UML Diagram



ii.

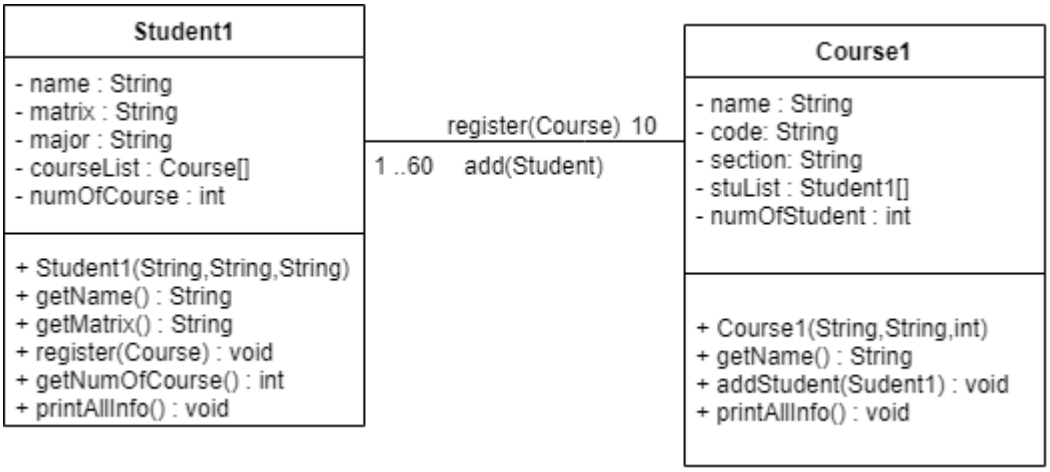
```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\2>javac *.java   && java TestAssociate

STUDENT NAME :ALI
NUMBER OF SUBJECT(s) TAKEN :2
LIST OF SUBJECT(s) TAKEN :
1. OOP
2. TP1


STUDENT NAME :AHMAD
NUMBER OF SUBJECT(s) TAKEN :3
LIST OF SUBJECT(s) TAKEN :
1. OOP
2. TP2
3. KP

STUDENT NAME :ABU
NUMBER OF SUBJECT(s) TAKEN :5
LIST OF SUBJECT(s) TAKEN :
1. OOP
2. TP1
3. TP2
4. KP
5. DM
```

iii.

```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\2>javac *.java   && java TestAssociate

STUDENT NAME :ALI
NUMBER OF SUBJECT(s) TAKEN :2
LIST OF SUBJECT(s) TAKEN :
1. OOP
2. TP1


STUDENT NAME :AHMAD
NUMBER OF SUBJECT(s) TAKEN :3
LIST OF SUBJECT(s) TAKEN :
1. OOP
2. TP2
3. KP
```

Exercise 1: Association Relationship Question 3

i. UML



ii.

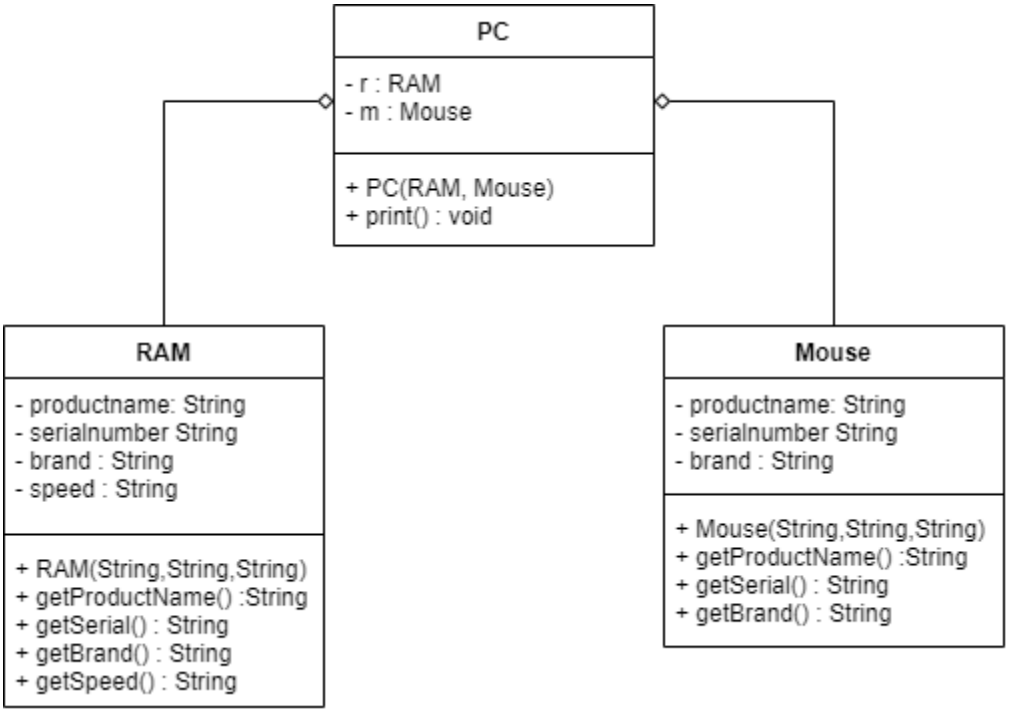
```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\3>javac *.java   && java TestAssociate2

COURSE NAME: OOP

NUMBER OF STUDENT(s): 3
LIST OF STUDENT(s):
1. ALI
2. ABU
3. BEN

COURSE NAME: TP1

NUMBER OF STUDENT(s): 1
LIST OF STUDENT(s):
1. ABU

STUDENT NAME: ALI
NUMBER OF SUBJECT(s) TAKEN: 1
LIST OF SUBJECT(s) TAKEN:
1. OOP

STUDENT NAME: ABU
NUMBER OF SUBJECT(s) TAKEN: 3
LIST OF SUBJECT(s) TAKEN:
1. OOP
2. TP1
3. TP2

STUDENT NAME: BEN
NUMBER OF SUBJECT(s) TAKEN: 1
LIST OF SUBJECT(s) TAKEN:
1. OOP

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\3>
```

iii.

```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\3>javac *.java   && java TestAssociate2

COURSE NAME: OOP

NUMBER OF STUDENT(s): 4
LIST OF STUDENT(s):
1. ALI
2. ABU
3. BEN
4. ABU

COURSE NAME: TP1

NUMBER OF STUDENT(s): 1
LIST OF STUDENT(s):
1. ABU

COURSE NAME: DM

NUMBER OF STUDENT(s): 4
LIST OF STUDENT(s):
1. ALI
2. ABU
3. BEN
4. ABU

STUDENT NAME: ALI
NUMBER OF SUBJECT(s) TAKEN: 2
LIST OF SUBJECT(s) TAKEN:
1. OOP
2. DM

STUDENT NAME: ABU
NUMBER OF SUBJECT(s) TAKEN: 4
LIST OF SUBJECT(s) TAKEN:
1. OOP
2. TP1
3. TP2
4. DM

STUDENT NAME: BEN
NUMBER OF SUBJECT(s) TAKEN: 2
LIST OF SUBJECT(s) TAKEN:
1. OOP
2. DM

STUDENT NAME: ABU
NUMBER OF SUBJECT(s) TAKEN: 3
LIST OF SUBJECT(s) TAKEN:
1. OOP
2. TP2
3. DM

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\3>
```

Exercise 2: Association Relationship Question 4

i. UML



ii.

```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\4>run.bat

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\4>javac *.java   && java PCTest
Information about this PC's RAM
Its brand: Corsair
Its serial number: A12345
Its manufacturer :Samsung
Its speed: 1G
Information about this PC's Mouse:
Its product name: Creative Labs FreePoint
Its serial number: 7300000000245
Its brand: Creative Labs

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\4>
```

iii.

```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\4>javac *.java   && java PCTest
Information about this PC's RAM
Its brand: Corsair
Its serial number: A12345
Its manufacturer :Samsung
Its speed: 1G
Information about this PC's Mouse:
Its product name: Creative Labs FreePoint
Its serial number: 7300000000245
Its brand: Creative Labs
Information about this PC's Keyboard:
Its product name: Ducky One 2 Mechanical Keyboard
Its serial number: DKON1787ST
Its brand: Ducky

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\4>
```

Exercise 3: Composition Relationship Question 1

	Aggregation	Composition
Lifetime	Independent lifetime	Owner's lifetime
Relation	Has-A	Owns
Upon Destruction	Destroying owning object does not affect the containing object	Destroyed when owning object is destroyed

Exercise 3: Composition Relationship Question 4

```
C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\Page 196 Composition Relationship Q3>javac *.java   && java PersonTest

Mohammed Ali
Jalan Pulai 13/2
Skudai,Johor 81310


C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\Page 196 Composition Relationship Q3>
```

Figure 1: Output of the program

The Person class have a aggregation relationship with the Address class, and have a composition relationship with the Name class. This can be justified since Name object is constructed in the Person's constructor, where Address object is being passed as a parameter for the Person's constructor. Hence indicates that Name object can only exist if the owning Person exist, while the Address class can exist independently (which is logical on the other hand which multiple people may share one same address).

The output of the program is generated by calling the println function, which the Person class has overloaded from the parent Java Object class. The Person class's toString implementation returns the output of the getFullName method from the Name object and the getFullAddress method from the Address object.

Exercise 3: Problem Solving Question 4

Section.Java

```java
public class Section{

    private String lectureName;
    private String sectionNum;

    public Section(String l, String s){
        lectureName = l;
        sectionNum = s;
    }

    public String getName(){
        return lectureName;
    }

    public String getSection(){
        return sectionNum;
    }


}
```

Subject.java

```java
public class Subject{

    private Section[] section;
    private String subjName;
    private String subjCode;

    public Subject(String n,String c){

        subjName = n;
        subjCode = c;

        section = new Section[10];

        if(c == "SCJ2153"){
            section[0] = new Section("Dr. Radiziah", "01");
            section[1] = new Section("Dr. Radiziah", "04");
            section[2] = new Section("Assoc. Prof. Dr. Norazah", "02");
        }

        if(c == "SCJ3253"){
            section[0] = new Section("Dr. Dayang Norhayati", "01");
        }

    }

    public void printInfo(){

        System.out.println("Subject Name: " + subjName);
        System.out.println("Subject Code: " + subjCode);

        for(int i=0;i<section.length;i++){
            if(section[i] != null){
                System.out.println("Section Code: " + section[i].getSection());
                System.out.println("Lecturer Name: " + section[i].getName());
            }
        }

        System.out.println("\n");

    }

}
```

SubjectTest.java

```java
public class SubjectTest{
    public static void main(String args[]){

        Subject oop = new Subject("OOP","SCJ2153");
        Subject rse = new Subject("Real-Time Software Engineering","SCJ3253");

        oop.printInfo();
        rse.printInfo();

    }

}
```

Output

```
C:\Windows\System32\cmd.exe

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\Page 201 Problem Solving Q4>run.bat

C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\Page 201 Problem Solving Q4>javac *.java   && java SubjectTest
Subject Name: OOP
Subject Code: SCJ2153
Section Code: 01
Lecturer Name: Dr. Radiziah
Section Code: 04
Lecturer Name: Dr. Radiziah
Section Code: 02
Lecturer Name: Assoc. Prof. Dr. Norazah


Subject Name: Real-Time Software Engineering
Subject Code: SCJ3253
Section Code: 01
Lecturer Name: Dr. Dayang Norhayati



C:\Users\Junyao\Documents\GitHub\OOP-Lab-2020\Page 201 Problem Solving Q4>
```

Discussion:

To create composition relationship between the Section class and Subject class, an if block is introduced in the Subject class's constructer to identify the Subject being created, hence construct corresponding Section object.

The printInfo method of Subject object is then called to display output.