UECS2344 Software Design: Practical 9

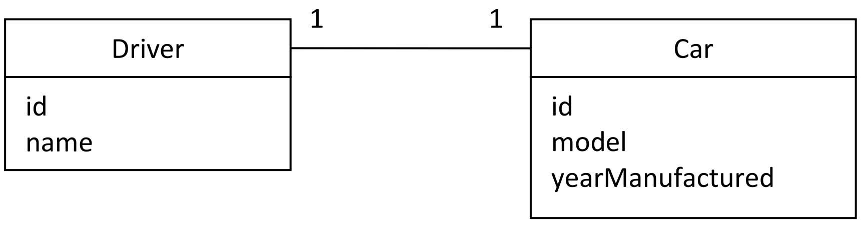
1. Consider the following use case description for a Course Management System.

Use Case Name	Enroll student in course
Actor	Course Administrator
Flow of Events: 1. Course Administrator enters code of the course to select. 2. System searches for the course record and displays title of the course. 3. Course Administrator enters name of student to search. 4. System searches for the student record and displays id of the student. 5. System adds the student to the list of students enrolled for the course.	

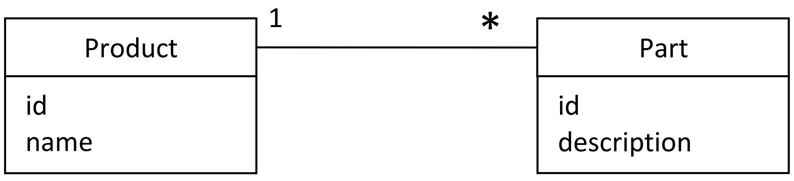
Draw a Data Flow Diagram that corresponds to the processing in the use case description.

2. For the Analysis Class Diagrams below, perform mapping of the entity classes to relational database tables. Note: The "id" attribute in the entity classes contains a unique value for objects of the class.

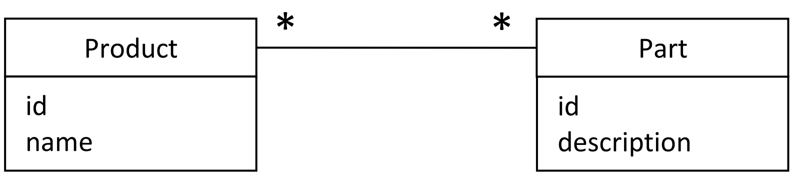
(a)



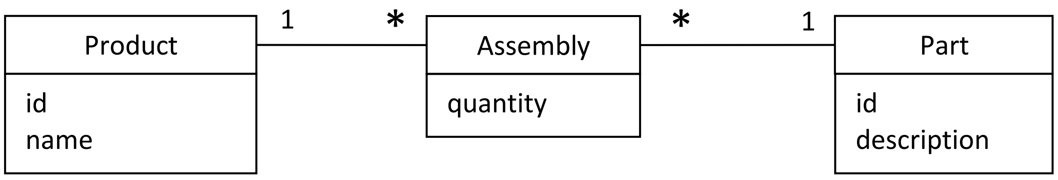
(b)



(c)



(d)

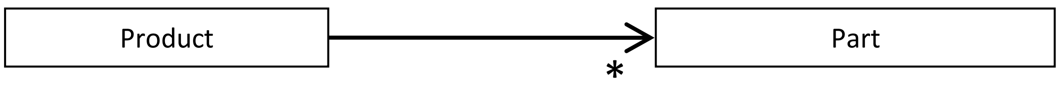


3. Refer to the Course Management System of Practicals 6 and 7. The table below shows the list of use cases and the entity classes. The table can be used to analyse the effect of the use cases on objects of the classes. The effect can be one or more of the following:
   - C – create
   - R – read
   - U – update
   - D – delete

   Fill up the table below to show the effect of each use case on objects on the classes:

Use Case	**Course**	**Student**
Display all courses (Iteration 1)		
Add new course (Iteration 1)		
Select a course (Iteration 2)		
Check if student enrolled in course (Iteration 2)		
Enroll student in course (Iteration 2 – a student can enroll in 1 course only)		
Enroll student in course (Iteration 3 – a student can enroll in 1 or more courses)		

4. Consider the following Design Class Diagram (with only the class names shown):

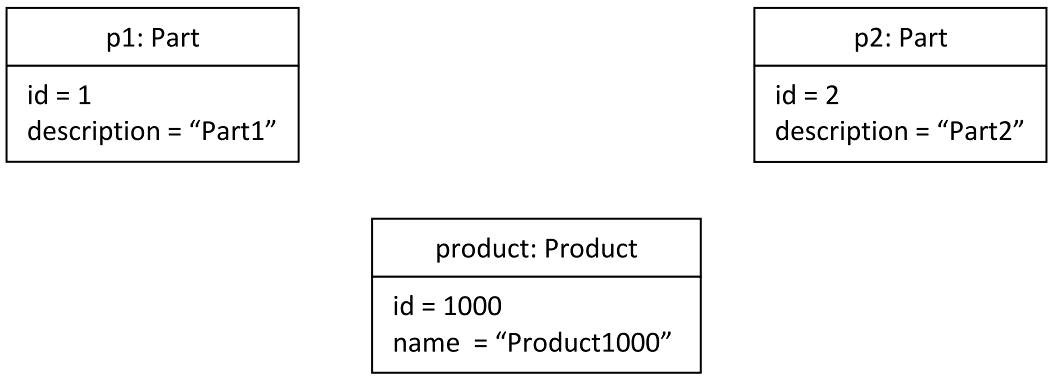


Suppose there is a class called App as follows:

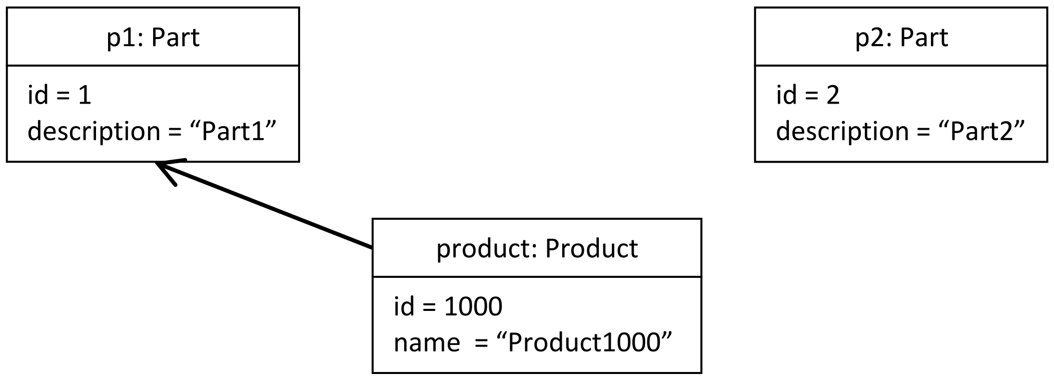
```
public class App {
    public static void main(String [] args) {
        Part p1 = new Part(1, “Part1”);
        Part p2 = new Part(2, “Part2”);

        Product product = new Product(1000, “Product1000”);
        product.assemble(p1);
        product.assemble(p2);
    }
}
```

The effect of the code is to create the objects and the links as follows:

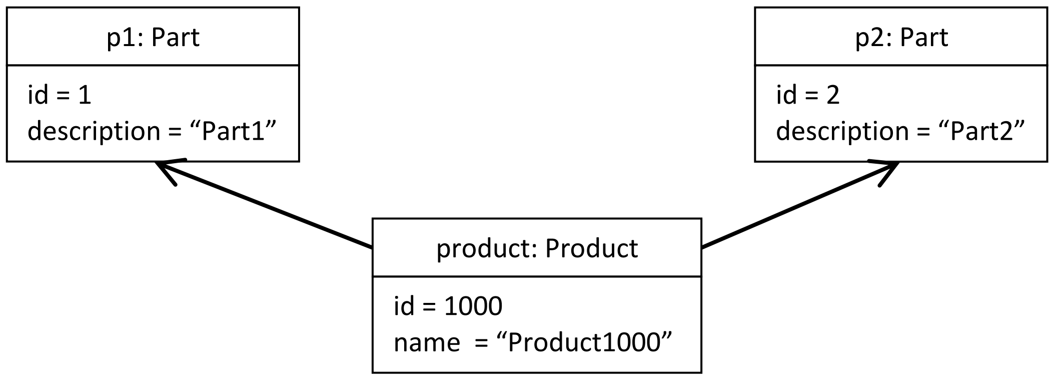
```
Part p1 = new Part(1, "Part1");
Part p2 = new Part(2, "Part2");
Product product = new Product(1000, "Product1000");
```



```
product.assemble(p1);
```

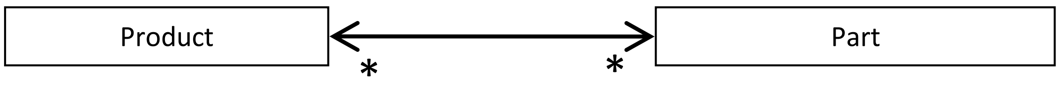


```
product.assemble(p2);
```



Write the code for the Product and Part classes and ensure that the links demonstrated above are created.

5. Consider the following Design Class Diagram (with only the class names shown):

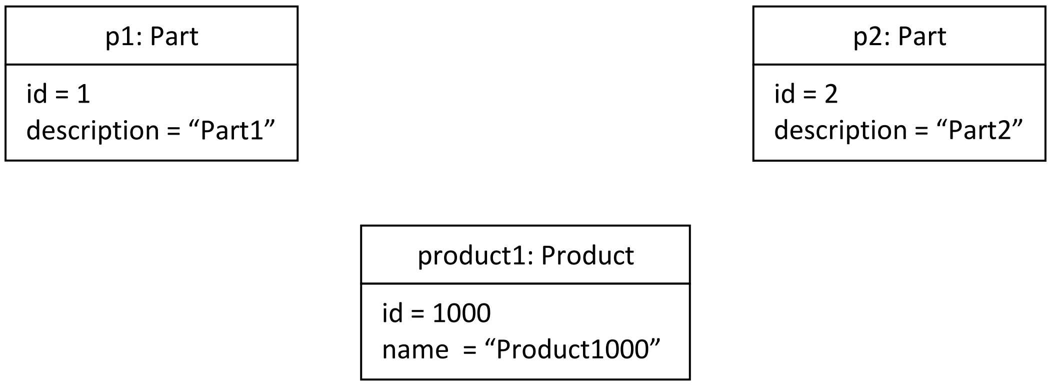


Suppose there is a class called App as follows:

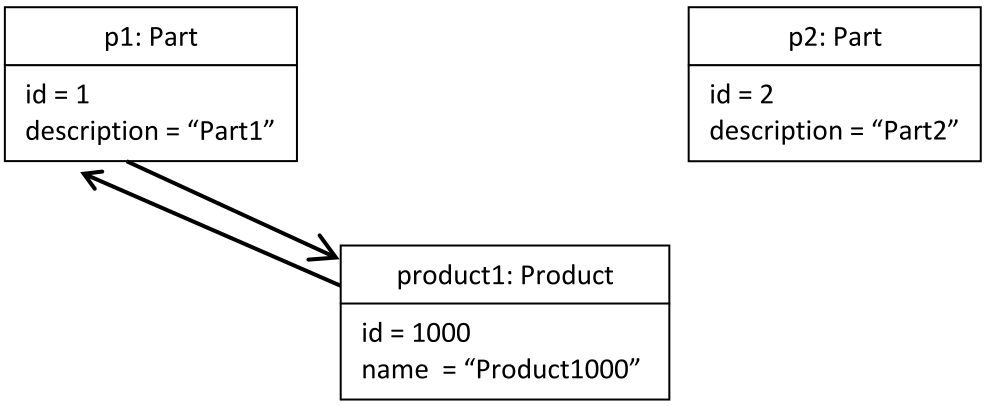
```
public class App {
    public static void main(String [] args) {

        Part p1 = new Part(1, “Part1”);
        Part p2 = new Part(2, “Part2”);

        Product product1 = new Product(1000, “Product1000”);
        product1.assemble(p1);
        product1.assemble(p2);

        Product product2 = new Product(2000, “Product2000”);
        product2.assemble(p1);
        product2.assemble(p2);
    }
}
```

The effect of the code is to create the objects and the links as follows:

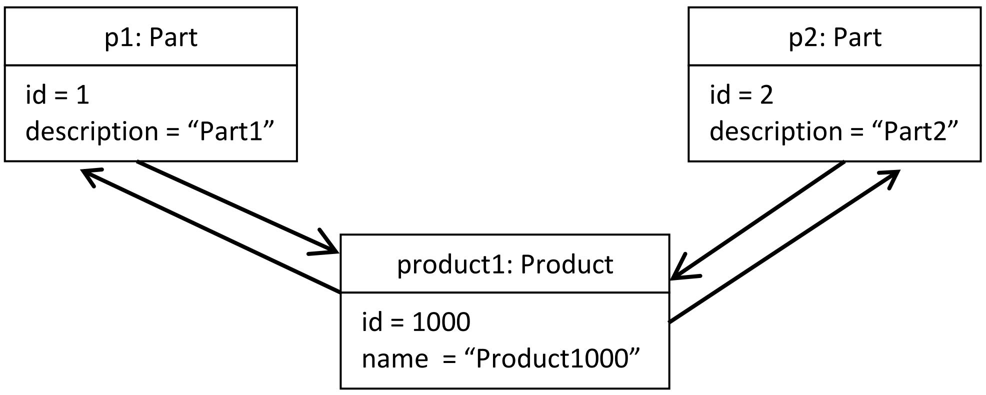
```
Part p1 = new Part(1, “Part1”);
Part p2 = new Part(2, “Part2”);
Product product1 = new Product(1000, “Product1000”);
```



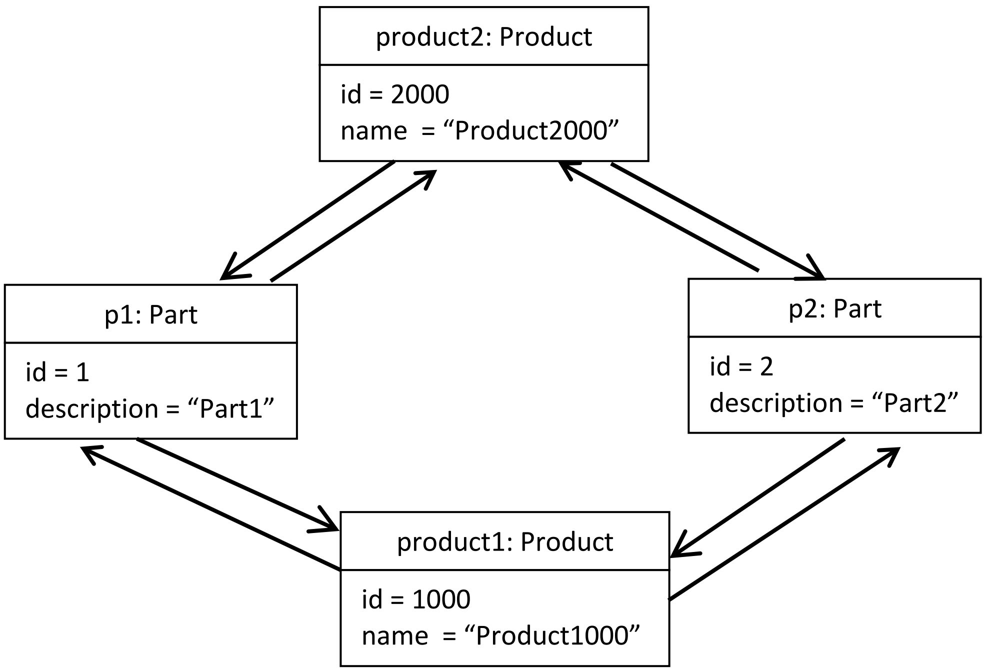
```
product1.assemble(p1);
```



```
product1.assemble(p2);
```

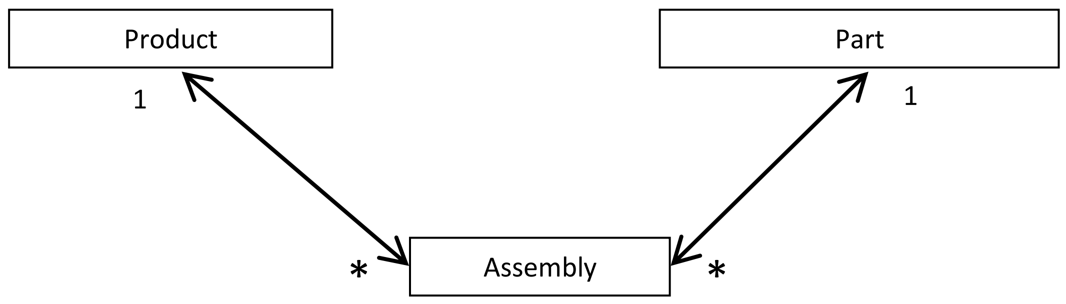


```
Product product2 = new Product(2000, “Product2000”);
product2.assemble(p1);
product2.assemble(p2);
```



Write the code for the Product and Part classes and ensure that the links demonstrated above are created.

6. Consider the following Design Class Diagram (with only the class names shown):

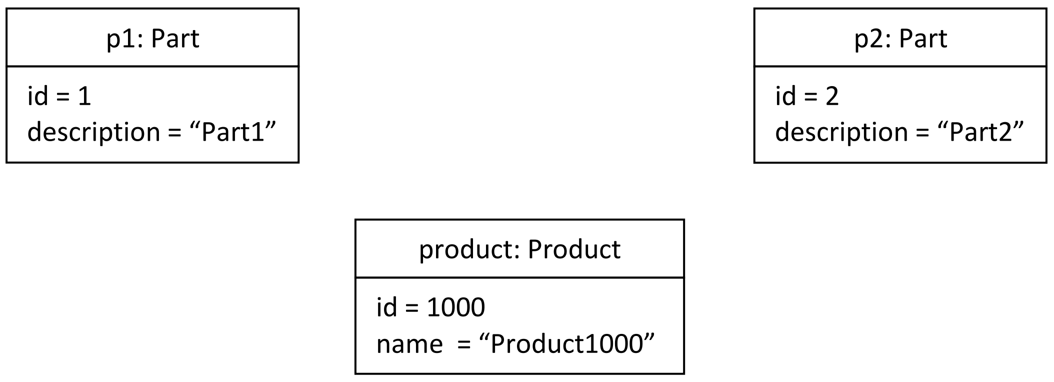


Suppose there is a class called App as follows:

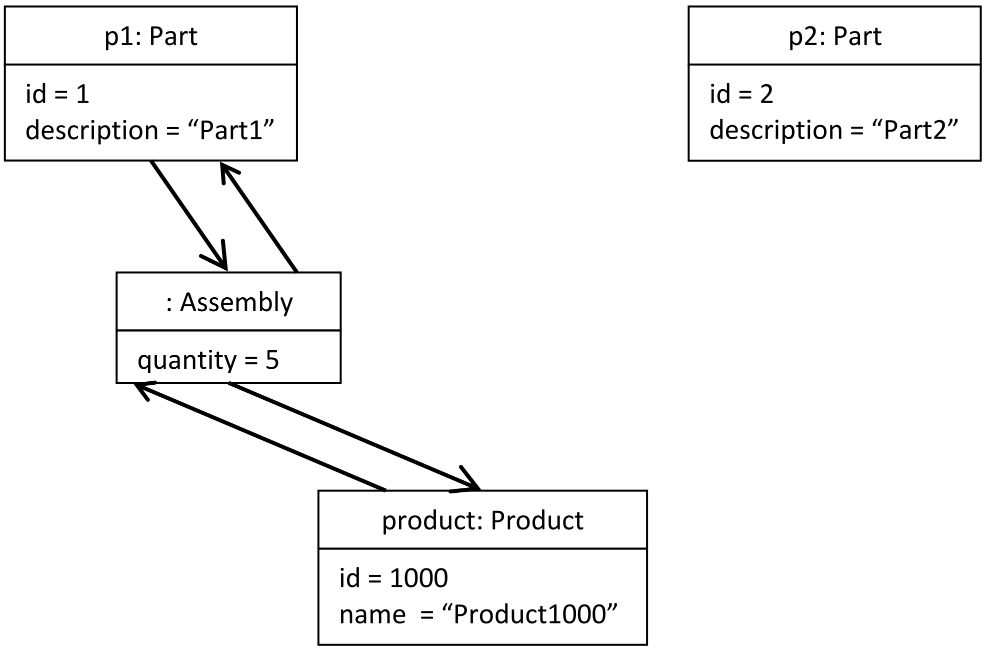
```
public class App {
    public static void main(String [] args) {
        Part p1 = new Part(1, “Part1”);
        Part p2 = new Part(2, “Part2”);

        Product product = new Product(1000, “Product1000”);
        product.assemble(p1, 5);
        product.assemble(p2, 3);
    }
}
```

The effect of the code is to create the objects and the links as follows:

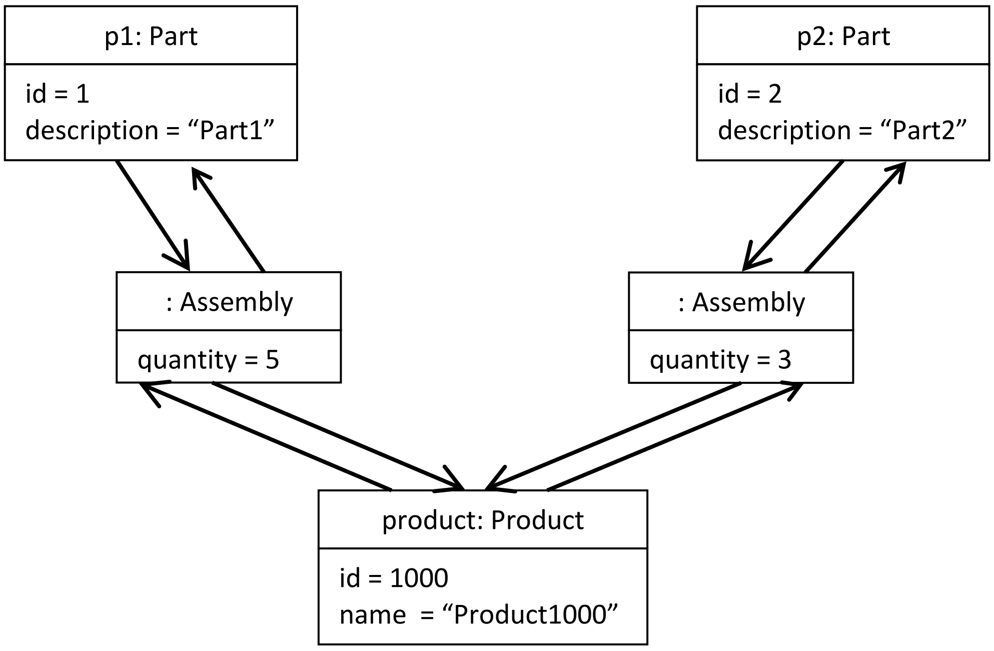
```
Part p1 = new Part(1, “Part1”);
Part p2 = new Part(2, “Part2”);
Product product = new Product(1000, “Product1000”);
```



```
product.assemble(p1, 5);
```

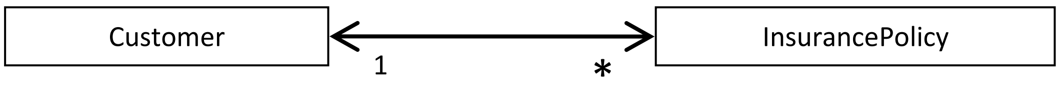


```
product.assemble(p2, 3);
```



Write the code for the Product, Part, and Assembly classes and ensure that the links demonstrated above are created.

7. Consider the following Design Class Diagram (with only the class names shown):

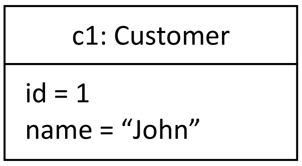


Suppose there is a class called App as follows:

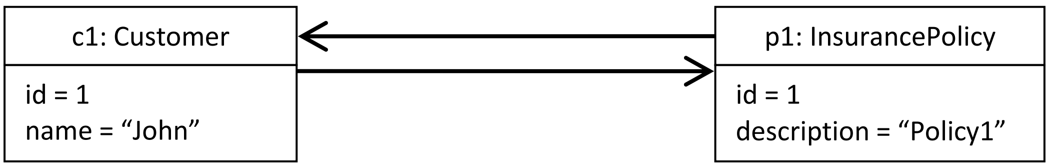
```
public class App {
    public static void main(String [] args) {
        Customer c1 = new Customer(1, “John”);
        InsurancePolicy p1 = new InsurancePolicy(1, “Policy1”, c1);
        InsurancePolicy p2 = new InsurancePolicy(2, “Policy2”, c1);
    }
}
```

The effect of the code is to create the objects and the links as follows:

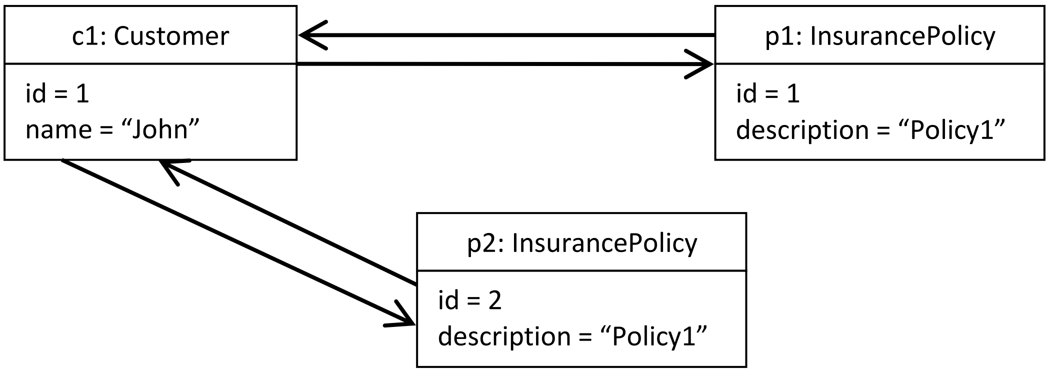
```
Customer c1 = new Customer(1, “John”);
```



```
InsurancePolicy policy1 = new InsurancePolicy(1, “Policy1”, c1);
```

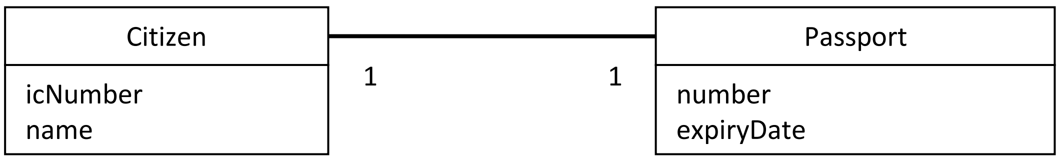


```
InsurancePolicy policy2 = new InsurancePolicy(2, “Policy2”, c2);
```



Write the code for the Customer and InsurancePolicy and ensure that the links demonstrated above are created.

8. Consider the following Analysis Class Diagram:



Suppose for the application, it is necessary to know the following:

- the passport that belongs to a citizen, and
- the citizen that a passport belongs to.

(i) Draw a Design Class Diagram

(ii) Write the code for the Citizen and Passport classes. Note that it is necessary that when a Passport object is created, the links between the Passport and Citizen object (both ways) are created as well.