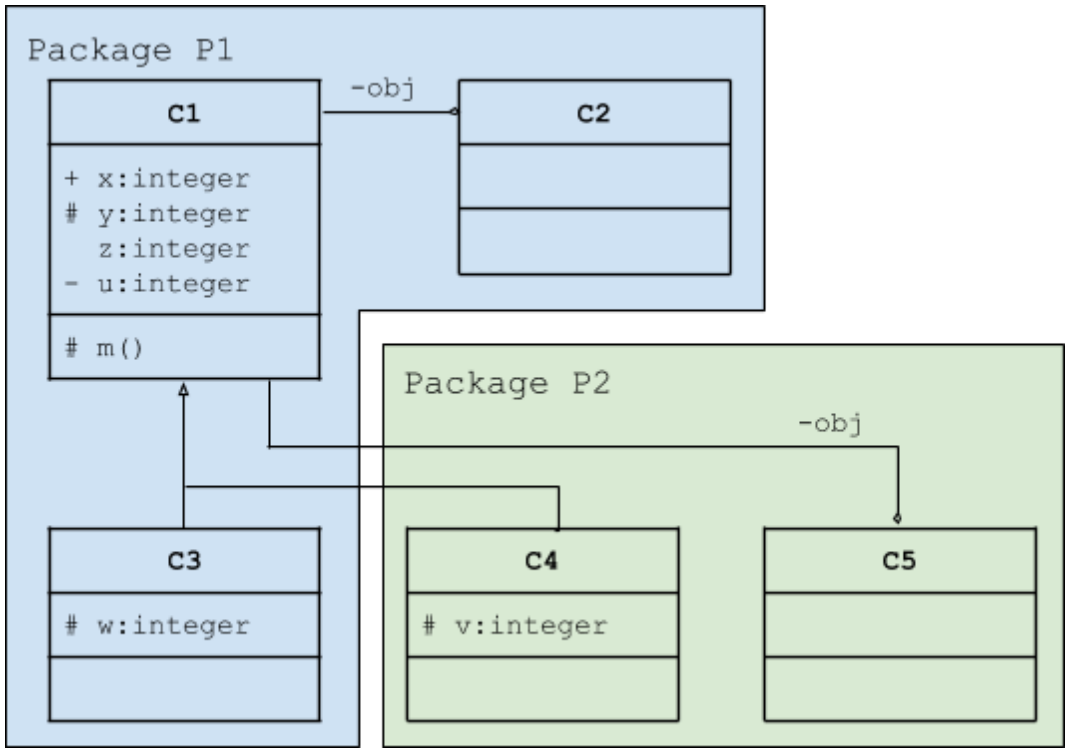
Exercise 1: Consider the following UML class diagram, and fill in the tables:



Can	access x	access y	access z	access u	access w	access v	invoke m
C1							
C3							
C4							

Can	access obj.x	access obj.y	access obj.z	access obj.u	invoke obj.m
C2					
C5					

Exercise 2: Show the output of the following program

```
class Vehicle{
  protected String owner;

  public Vehicle(){
    this("No owner");
    System.out.println("Owner:"+owner);
  }
  public Vehicle(String owner){
    setOwner(owner);
    System.out.println("Owner:"+owner);
  }
  public void setOwner(String owner){
    this.owner = owner;
    System.out.println("Owner is set");
  }
}
```

```
class MotorVehicle extends Vehicle{
  protected int nWheels;
  protected double sEngine;

 public MotorVehicle(){
   this(2, 2.5);
   System.out.println(nWheels+" Wheels");
   System.out.println(sEngine+" L");
 }
 public MotorVehicle(int n,double e){
   setNumWheels(n);
   System.out.println(nWheels+" Wheels");
   setEngineSize(e);
   System.out.println(sEngine+" L");
 }
 public void setNumWheels(int n){
   nWheels = n;
   System.out.println("Wheels are set");
 }
 public void setEngineSize(double e){
   sEngine = e;
   System.out.println("Engine is set");
 }
}
```

```
class Bike extends MotorVehicle{
}
```

```
class Car extends MotorVehicle{
  public Car(){
    super(4, 4.5);
  }
}
```

```
class Test{
  public static void main(String[] a){
    Bike b = new Bike();
    System.out.println("--------------");
    Car c = new Car();
  }
}
```

Exercise 3: Design, implement and compile:

A real estate agency lists a number of properties of different types for sale. Properties that gets sold are removed from the list, and new ones can be added. The maximum number of properties the agency can handle at a given time is 100.

All property entries show the property area in m^2, number of rooms, neighborhood name, and price.

Description of each of the types of listed properties include:
Villa: has a swimming pool or not, and number of adjacent streets
Apartment: on which floor? and is there a parking lot?
Furnished apartment: furniture quality on a scale 1(best) to 5

Your design should provide :

A constructor for each class to initialize all of its attributes

A display method that prints the type of the class in addition to its attribute values

Suitable access modifiers for all class members

* Please use a dictionary when encountering a new word to you

Solutions:

Exercise 1:

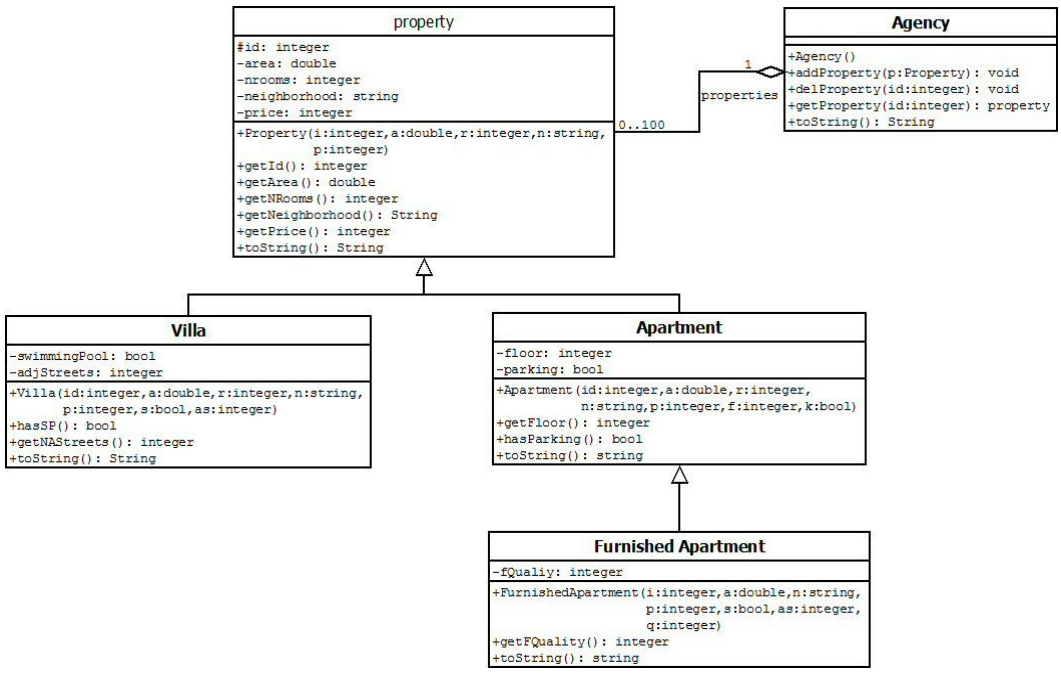
Can	access x	access y	access z	access u	access w	access v	invoke m
C1	✔	✔	✔	✔	✘	✘	✔
C3	✔	✔	✔	✘	✔	✘	✔
C4	✔	✔	✘	✘	✘	✔	✔

Can	access obj.x	access obj.y	access obj.z	access obj.u	invoke obj.m
C2	✔	✔	✔	✘	✔
C5	✔	✘	✘	✘	✘

Exercise 2:

```
Owner is set
Owner:No owner
Owner:No owner
Wheels are set
2 Wheels
Engine is set
2.5 L
2 Wheels
2.5 L
--------------
Owner is set
Owner:No owner
Owner:No owner
Wheels are set
4 Wheels
Engine is set
4.5 L
```

Exercise 3:



```
public class Property {
  private int id;
  private double area;
  private int nrooms;
  private String neighborhood;
  private int price;

  public Property(int i, double a, int r, String n, int p) {
    id = i;
    area = a;
    nrooms = r;
    neighborhood = n;
    price = p;
  }

  public int getId() { return id; }

  public double getArea() { return area; }

  public int getNRooms() { return nrooms; }

  public String getNeighborhood() { return neighborhood; }

  public int getPrice() { return price; }

  public String toString() {
    return id+":\tin "+getNeighborhood()+" has "+getNRooms()+" on "+getArea()+" m2";
  }
}
```

```
public class Villa extends Property {
  private boolean swimmingPool;
  private int adjStreets;

  public Villa(int i, double a, int r, String n, int p, boolean s, int as) {
    super(i, a, r, n, p);
    swimmingPool = s;
    adjStreets = as;
  }

  public boolean hasSP() { return swimmingPool; }

  public int getAStreets() { return adjStreets; }

  public String toString() {
    return "Villa #"+super.toString()+(hasSP()? " with a swimming pool":"")+
           " located on "+getAStreets()+" street"+(getAStreets()>1? "s":"");
  }
}
```

```
public class Apartment extends Property {
  private int floor;
  private boolean parking;

  public Apartment(int i, double a, int r, String n, int p, int f, boolean k) {
    super(i, a, r, n, p);
    floor = f;
    parking = k;
  }

  public boolean hasParking() { return parking; }

  public int getFloor() { return floor; }

  public String toString() {
    return "Apartment #"+super.toString()+(hasParking()? " with a parking":"")+
```

```
            " located on floor "+getFloor();
  }
}
```

```
public class FurnishedApartment extends Apartment {
  private int fQuality;

  public FurnishedApartment(int i, double a, int r, String n, int p, int f, boolean
k, int q) {
    super(i, a, r, n, p, f, k);
    fQuality = q;
  }

  public int getFQuality() { return fQuality; }

  public String toString() {
    return "Apartment #"+super.toString()+(hasParking()? " with a parking":"")+
           " located on floor "+getFloor()+" furnished with "+
           (fQuality==1? "excellent":(fQuality==2? "v.good":(fQuality==3? "good":
           (fQuality==4? "not bad":"poor quality"))))+" furniture";
  }
}
```

```
public class Agency {
  private Property[] properties;

  public Agency() {
    properties = new Property[100];
    for (int i=0; i<100; i++) properties[i] = null;
  }

  public void addProperty(Property p) {


  }

  public void delProperty(int i) {

  }
}
```