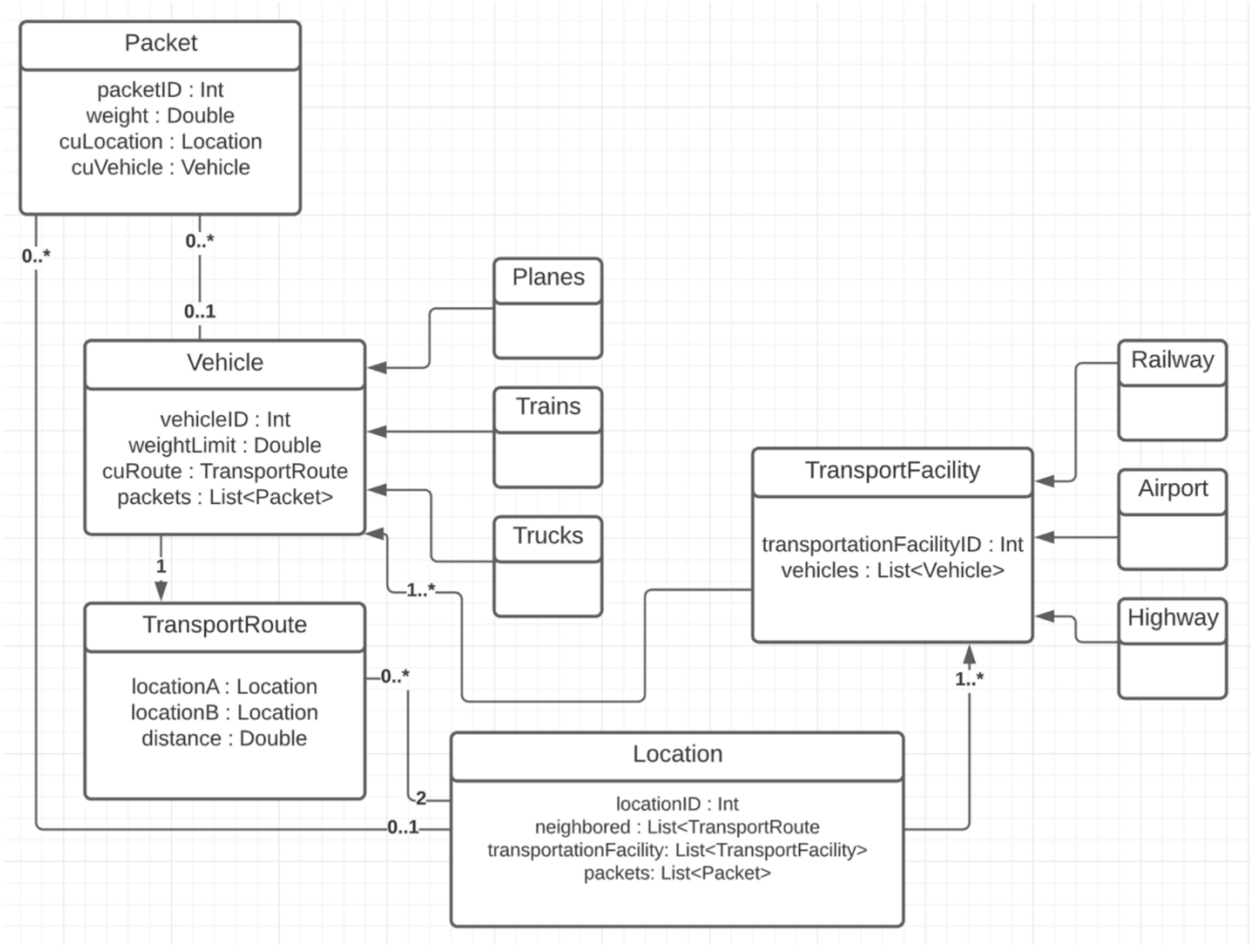
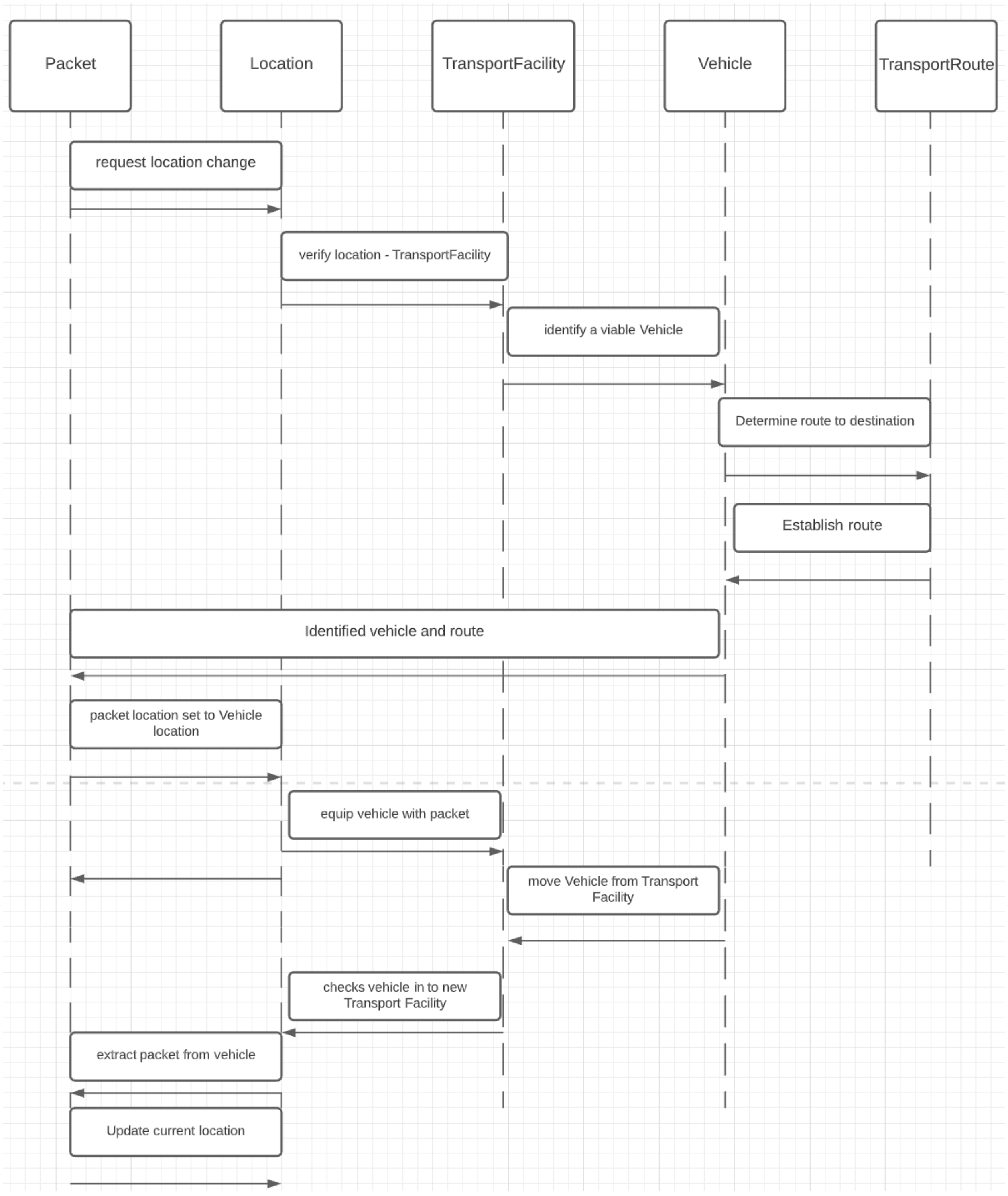
Problem 1 (20 points)
Assume the following requirements:

Packets are sent from one location to another. Packets have a certain weight. Locations are characterized by their transportation facilities, e.g. railway stations, airports and highway connections. Some locations are neighbored, i.e. there exists a direct transportation route between these locations. The transportation route between the locations has a certain length, i.e. the distance between the locations. Planes, trains, and trucks are used for transportation; each plane / train / truck may load a maximum packet weight. For each packet we want to know where it is, i.e. at which location or transport (plane, train, truck).
(i) Draw a class diagram for this problem; identify the semantic relationships and cardinalities. (10 points)



(ii) Draw the sequence diagram corresponding to a packet being sent from one location to another. (10 points)


Packet
Location
TransportFacility
Vehicle
TransportRoute
request location change
verify location - TransportFacility
identify a viable Vehicle
Determine route to destination
Establish route
Identified vehicle and route
packet location set to Vehicle location
equip vehicle with packet
move Vehicle from Transport Facility
checks vehicle in to new Transport Facility
extract packet from vehicle
Update current location

Problem 2 (10 points)
Given the following UML write the corresponding (skeleton) Java code.

```
public class A {
        public void dependencyFunctionOnClassB(B argB) {
        }
}
public class B {}
public class C extends A {
        public void dependencyFunctionOnClassD(D argD){
        }
}
public class D {
        Public B[] arrayB;
        Public F[] arrayF = new F[5];
}
Public class E extends C {}
Public class F {
        Public D[] arrayD = new D[2];
}
```

Problem 3 (20 points)
Integer is part of the Java API. Suppose you attempt to extend the Integer class and add a new method that returns the integer as a String that is written in hexadecimal.

(i) Explain why you're having trouble doing it. (5 points)

- Having trouble doing it because it has a 'final' configuration which disallows subclasses

(ii) Alright, so the people who wrote the code had their reasons to not want you to extend the class. Give an example of how things could go very wrong if they didn't do it this way. (5 points)

- Things can go wrong if an Integer subclass overrides an Integer method.

(iii) Recommend a solution to the problem that doesn't involve subclassing. (10 points)

- A solution to this problem is to have an Integer object defined within the Integer class that implements hexadecimal behavior to equip the methods within the Integer class. After this implementation, the additional methods can be included within the newly created object.

Problems 4 & 5 are completed as java programs within the artifacts of this deliverable