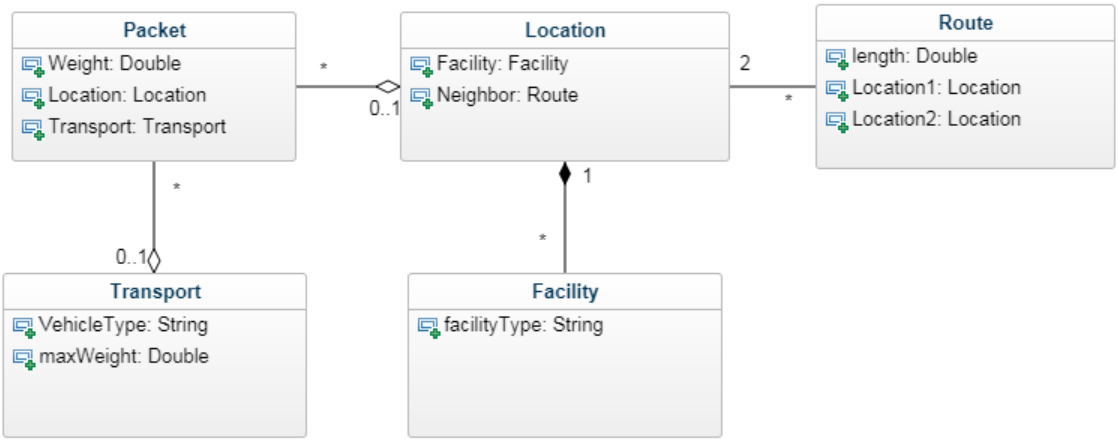
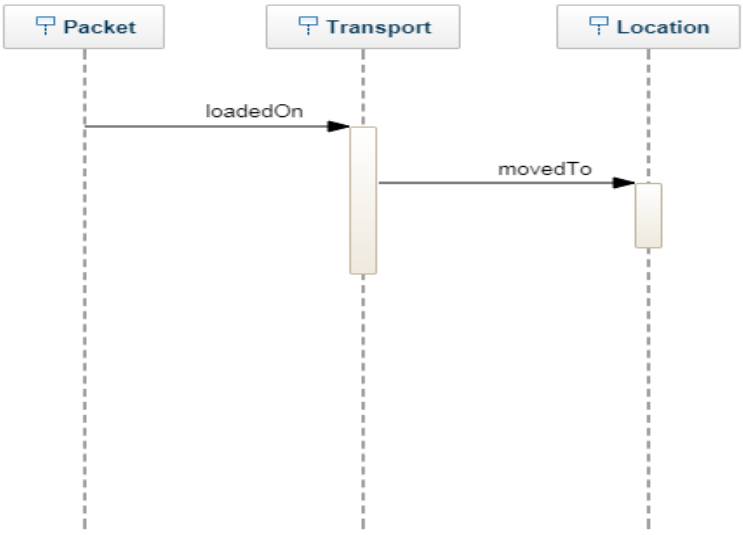
Problem 1

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)



Problem 2

A.java

```
Public class A {
Public void method(B b){}
}
```

B.Java

```
Public class B{}
```

C.java

```
Public class C extends A{
Public void method(D d){}
}
```

D.java

```
Public class D{
ArrayList<B> b = new ArrayList<B>();
Public F f1 = new F();
Public F f2 = new F();
Public F f3 = new F();
Public F f4 = new F();
Public F f5 = new F();
}
```

E.java

```
Public class E extends C{}
```

F.java

```
Public class F{
Public D d1 = new D();
Public D d2 = new D();
}
```

Problem 3

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)

You cannot extend the Integer class to another class because wrappers for primitive data types in java are declared final and thus are immutable.

(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)

If you were able to extend integer into another class, you could in essence overwrite any of the methods that integer has which would cause integer to not work the way you might think it does anymore.

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

Integer already has a method that does this. It is called toHexString. But I guess the way you would do it if you are trying to build this yourself would be to continue to do integer division on the given integer by 16, keeping track of the remainders. Once the integer division hits 0, then take all of the remainders and convert them to strings where 0-9 remain the same and 10-15 convert to A-F. Then take those remainders and concatenate them into one string ordering from the last remainder achieved until the first one.