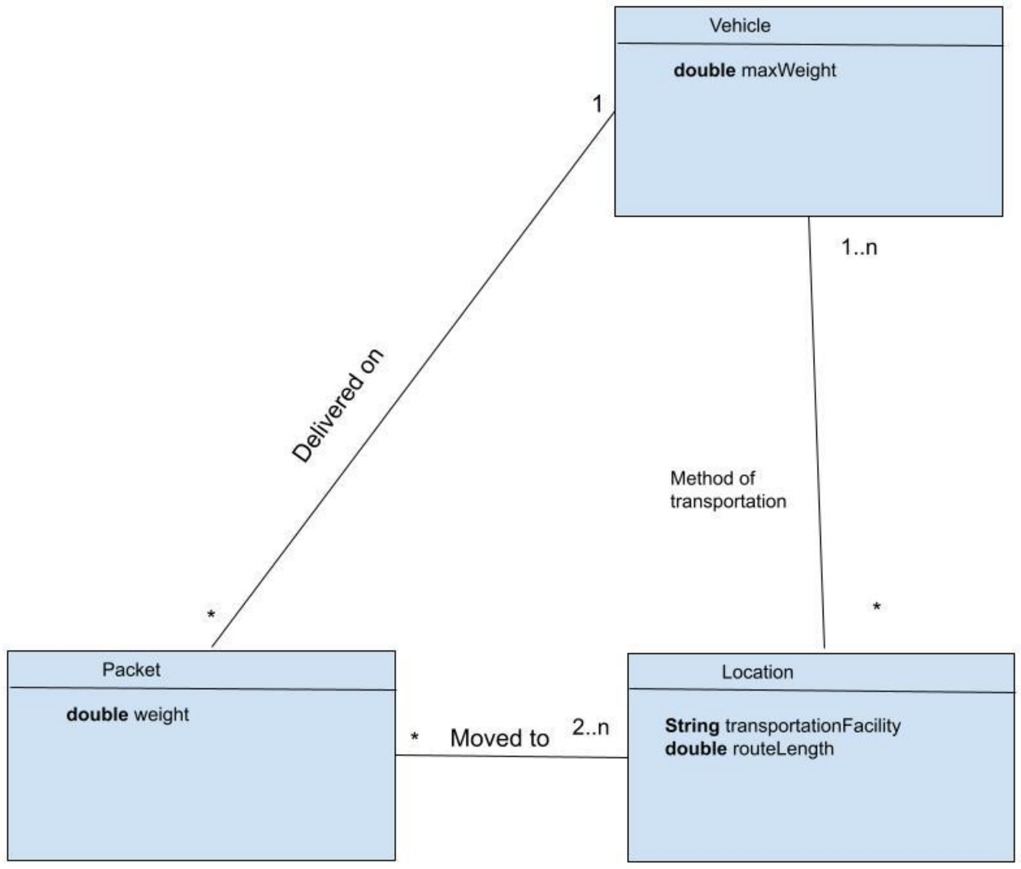
Problem 1:

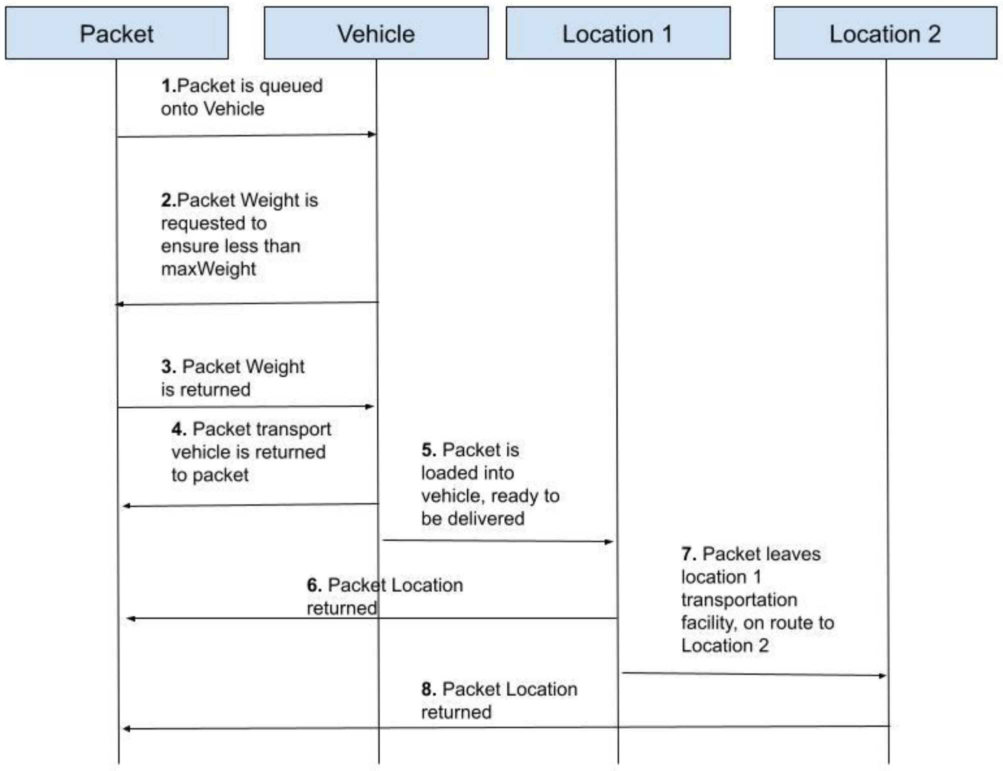
1a)



- **There are three classes:**
    - Packet
        - Has one attribute, weight (double)
    - Location
        - Has two attributes, transportationFacility (String) and routeLength (double)
    - Vehicle
        - Has one attribute, maxWeight (double)
- **Relationships:**
    - Packet - Location:

        - A packet moves from starting location to destination
            - Must be 2 locations per packet
        - A location may contain 0 to many packets at a time
    - Location - Vehicle:
        - A location may have 1 vehicle in the transportation route if its direct
        - May use multiple vehicles if not direct:
            - Ex: first shipped by truck to airport, then shipped to location via airplane
        - Vehicle can be on multiple location routes
    - Packet - Vehicle:
        - A packet can be in one vehicle at a time
        - A vehicle can have 0 to multiple packets as long as the total weight of all packets are less than the maxWeight

1b)


Packet
Vehicle
Location 1
Location 2
1.Packet is queued onto Vehicle
2.Packet Weight is requested to ensure less than maxWeight
3. Packet Weight is returned
4. Packet transport vehicle is returned to packet
5. Packet is loaded into vehicle, ready to be delivered
6. Packet Location returned
7. Packet leaves location 1 transportation facility, on route to Location 2
8. Packet Location returned

Problem 2:

A has an dependency relationship with B, therefore the should be objects constructed from the B class.

```
Class A {

        B bObject;

}
```

B doesn't have a association with any other class

```
Class B {


}
```

C knows about A, has an inheritance relationship and has a dependency relationship with the D class.

```
Class C extends A{

        D dObject;

}
```

D knows about B, has an inheritance relationship and the D class has five instances of the F class.

```
Class D extends B {

        F fObject1 = new F();

        F fObject2 = new F();

        F fObject3 = new F();

        F fObject4 = new F();

        F fObject5 = new F();

}
```

E knows about the C class, has an inheritance relationship.

```
Class E extends C {


}
```

The F class has 2 instances of the D class.

```
Class F {
        D dObject1 = new D();
        D dObject2 = new D();
}
```

Problem 3:

i) Java Wrapper classes (such as Integer) are final. Therefore you cannot change and.or manipulate the state of variables within the class by simply extending it. In other words, the Integer class is immutable. This method cannot be constructed in the extended class.

ii) Integer wrapper classes were implemented this way because Integer class variables/methods were never intended to be changed. Subclasses can change the behavior of the Wrapper class due to polymorphism.

For example: If I made a child class that adds 2.5 to all integers when instantiated, this would change the state of the Integer variable.

```
Integer i = new MyInteger(3);
System.out.println(i);


Console: 5.5
```

iii) A solution to this problem that doesn't involve subclassing is making a new class and constructing the method using the Integer Wrapper class. This method would take the Integer as a parameter and return it as a hexadecimal String.