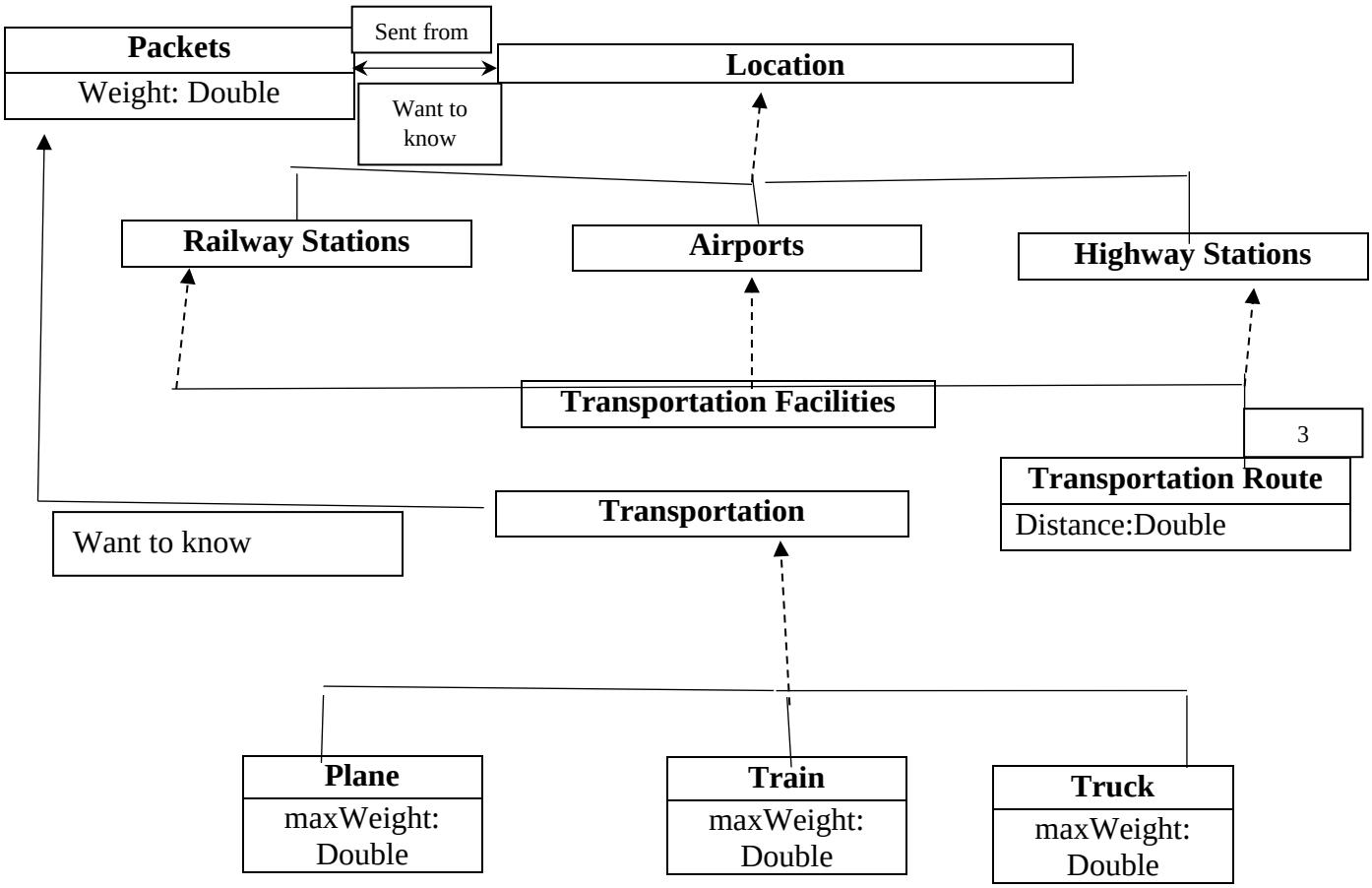
Julian Dickens
09/26/17
HW 2
CS 445

1. 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).

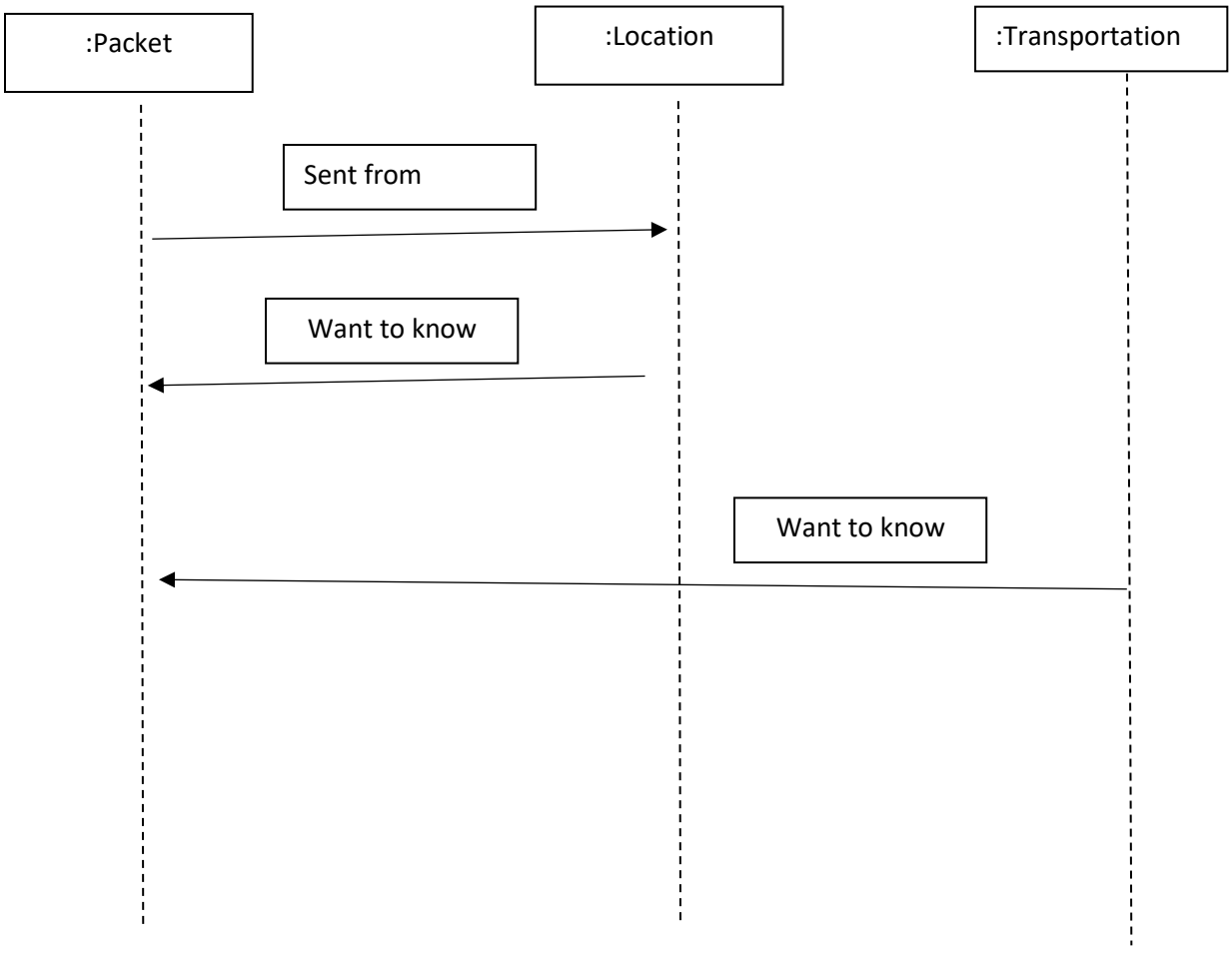
(i) Draw a class diagram for this problem; identify the semantic relationships and cardinalities. (10 points)



Semantic Relationships

Nouns	Verbs	Cardinalities
Packets	Sent from one location to another (Association)	1
Location	Have a certain weight (Attribute)	1
Railway Stations	Want to know (Association)	1
Airports		1
Highway Stations		1
Transpotation		1
Transportation Route		3
Plane		1
Train		1
Truck		1
Weight		(Attribute)
MaxWeight		(Attribute)
Distance		(Attribute)

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



2. Java (skeleton) code

```
public class E extends C
{

}

public C extends A
{
        Public void for (C c)
        {
        }
}

public class A
{
        Public void for (B b)
        {
        }
}

public class B
{

}

public class D //Define method that declares dependency aqrdsssssssda
{
        private B b;
        private F f1;
        private f f2;
}

public class F
{
        private D d1;
        private D d2;
        private D d3;
        private D d4;
        private D d5;
}
```

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)

There are several reasons as to why the programmer could have trouble implementing this method. They include:

- Implementing the conversion from decimal to hex without calling another method can be a complicated process. The programmer has to find the highest order of 2 that is not larger than the number being passed in the method, then subtract that number, and find the highest order of 2 of the difference, and then subtract that number. The programmer must continue this process until the difference is 0.
- The programmer may be attempting to override the method that is already implemented in Java, known as toHexString,() to do the conversion.

(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.

If the Integer class was not inherited, the new method would not be able to inherit the classes that are originally in the Integer class. If the programmer wanted to use methods in the Integer API, he would have to call the old Integer class.

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

A solution to the problem that does not involve subclassing includes:

- Using composition instead of inheritance. The new Integer class does not have to extend from the old Integer class, but rather, create methods inside of the old Integer class.