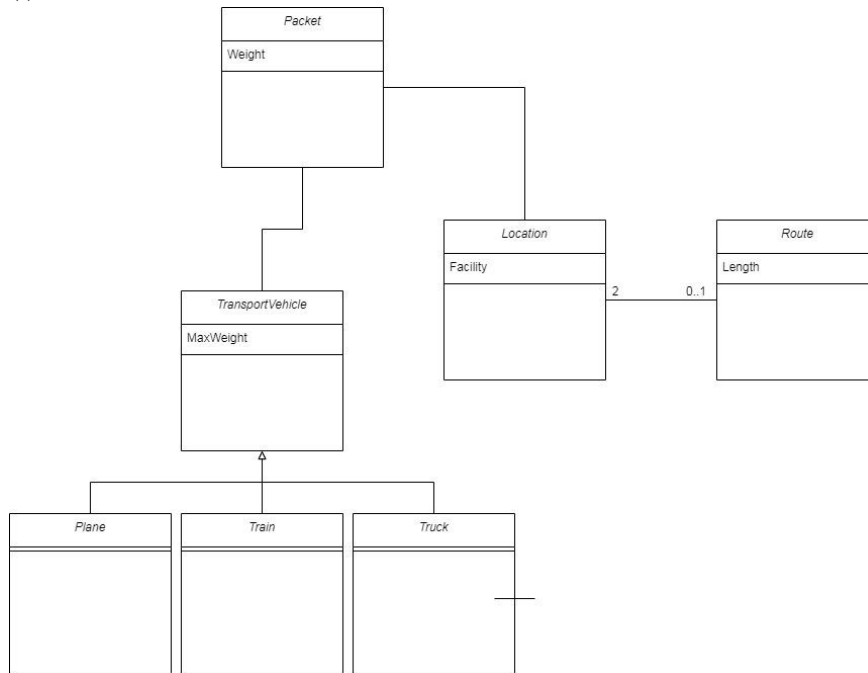
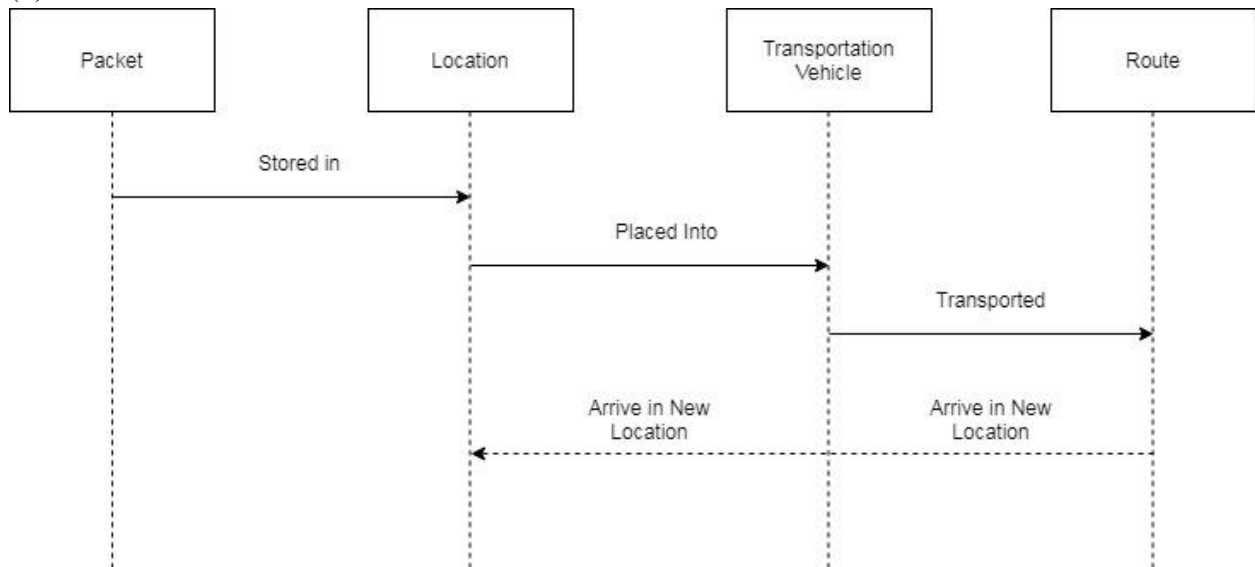


Problem 1:

(i)



(ii)



Problem 2:**A:**

```
public class A
{
    //Class A is associated with Class B
    B bObject = new B();
}
```

B:

```
public class B
{

}
```

C:

```
//Class C inherits from Class A
public class C extends A
{
    //Class C is associated with Class D
    D dObject = new D();
}
```

D:

```
//Class D inherits from Class B
public class D extends B
{
    //Aggregate relationship exists between Class D and Class F
    F fObject = new F();
}
```

E:

```
//Class E inherits from Class C
public class E extends C
{

}
```

F:

```
public class F
{
    //Aggregate relationship exists between Class F and Class D
    D dObject = new D();
}
```

Problem 3:

(i) – The class header for the Integer class is as follows:

public final class Integer extends Number implements Comparable<Integer>

Because the class is final, it cannot be extended in the first place to add our new method.

(ii) – Integer is one of many wrapper classes that is special. These classes are responsible for autoboxing/unboxing, and allowing subclasses to be created could complicate this process. The original functionality intended with these classes could be lost with the creation of subclasses.

(iii) – Instead of extending the Integer class, a new method can be written in another class. This method could replicate the function of the toString() method of the Integer class with special steps taken to return a String written in hexadecimal rather than a traditional String representation of an integer value.

Problems 4 & 5:

The code for problems 4 & 5 is included in my submission inside of the src folder.

Upon completing the assignment, I found the most difficult part of the assignment to be Problem 1. It was difficult trying to visualize how these diagrams should look, but it became clearer upon further thinking about it. It was also a little tricky and tedious to use the programs necessary to create these charts that are in this submission.

The following are some important statistics about my code:

ImprovedRandom.java:

Lines of code: 22

Cyclomatic Complexity of default constructor: 1

Cyclomatic Complexity of constructor with argument: 1

Cyclomatic Complexity of nextIntBetween(): 2

ImprovedStringTokenizer.java:

Lines of code: 25

Cyclomatic Complexity of constructor with one argument: 1

Cyclomatic Complexity of constructor with two arguments: 1

Cyclomatic Complexity of constructor with three arguments: 1

Cyclomatic Complexity of toArray(): 1

ImprovedRandomTest.java:

Lines of code: 24

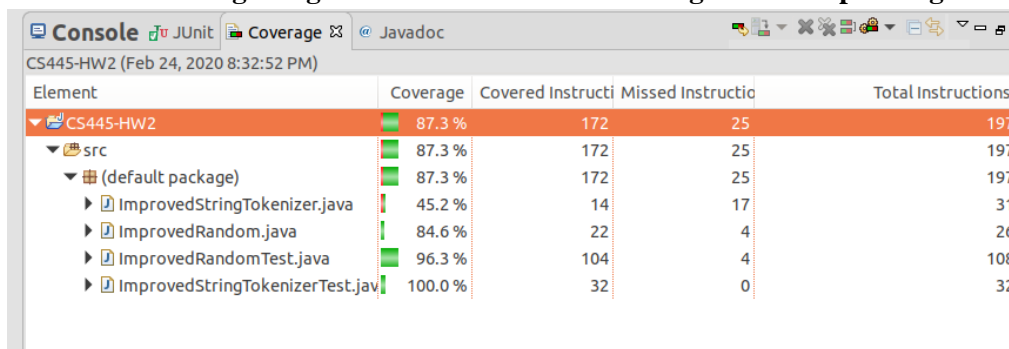
Cyclomatic Complexity of testBetween(): 3

ImprovedStringTokenizerTest.java:

Lines of code: 15

Cyclomatic Complexity of testArray(): 1

Unit Test Covering using EcEmma Java Code Coverage 3.1.2 Eclipse Plugin:



Element	Coverage	Covered Instructi	Missed Instructio	Total Instructions
CS445-HW2 (Feb 24, 2020 8:32:52 PM)	87.3 %	172	25	197
src	87.3 %	172	25	197
(default package)	87.3 %	172	25	197
ImprovedStringTokenizer.java	45.2 %	14	17	31
ImprovedRandom.java	84.6 %	22	4	26
ImprovedRandomTest.java	96.3 %	104	4	108
ImprovedStringTokenizerTest.jav	100.0 %	32	0	32