CS 445

Memo

To: Professor Bistriceanu

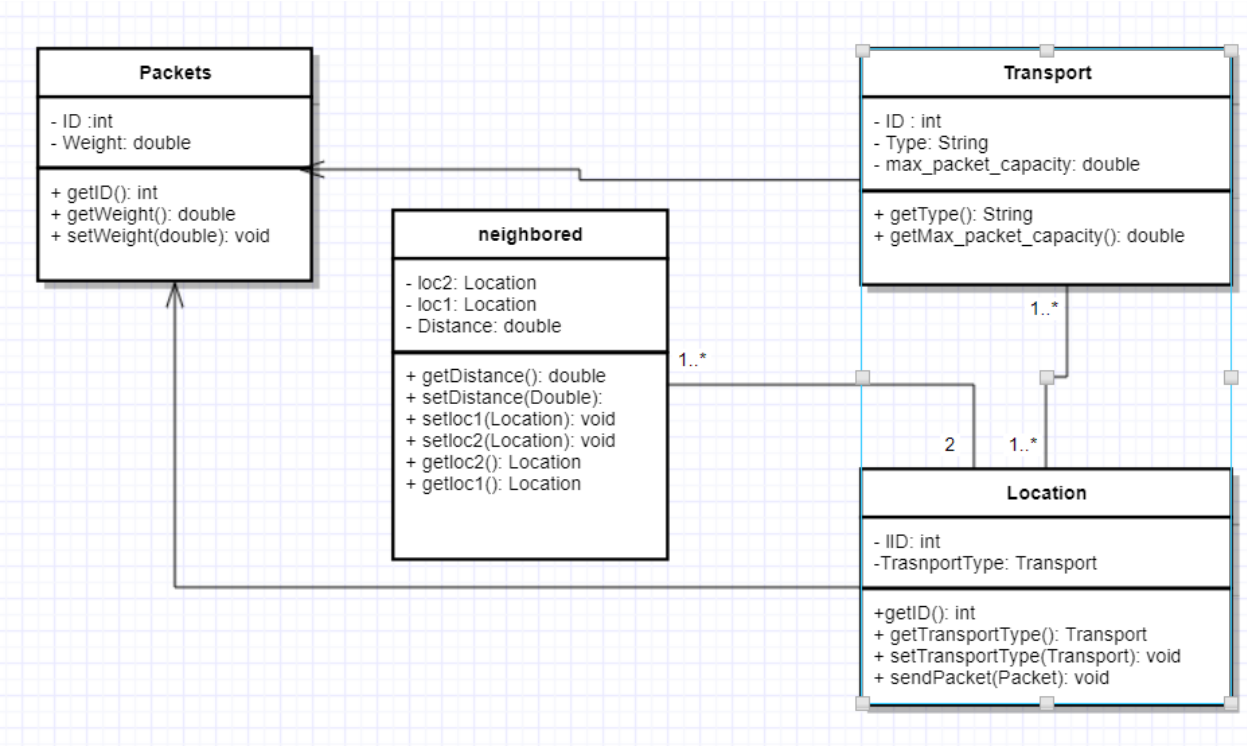
From: Amer Alsabbagh

cc: TA

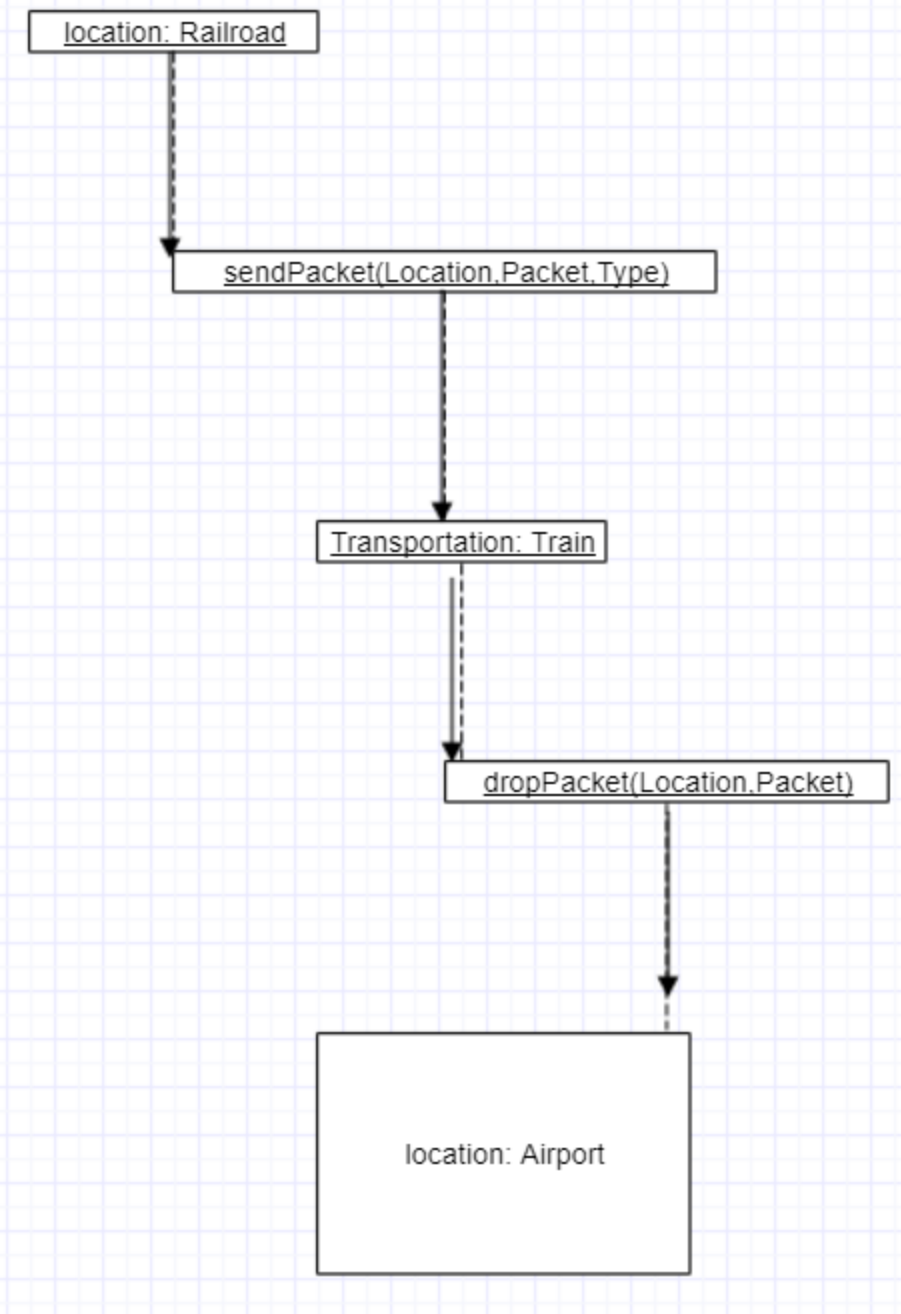
Date: 9/26/2017

Re: HW2

Problem 1:a



B



Problem 3:

There are good reasons to makes the primitive wrappers final.

First, note that these classes get special treatment in the language itself - unlike any normal classes, these are recognized by the compiler to implement auto(un)boxing. Simply allowing subclasses would already create pitfalls (unwrapping a subclass, performing arithmetic and wrapping back would change the type).

Also, wrapper types can have shared instances (e.g. look at Integer.valueOf(int)) requiring that an instance of a wrapper is strictly immutable. Allowing subclasses would open a can of worms where immutability can no longer be assured, forcing robust code to be written as if the wrappers were mutable, leading to defensive copies, wasting memory and CPU time and also creating issues with their usefulness in multi-threaded scenarios.

Essence: wrapper types need to be immutable to ensure uniform capabilities across all instances; immutability being an important part of their known properties. And to guarantee immutability, the types need to be final.

If you need added functionality, implement it as utility class like you would do for primitives (see java.lang.Math for example).

Source: https://stackoverflow.com/questions/28237542/why-wrapper-classes-in-java-are-final

This assignment wasn't particularly difficult. It was very straight forward as it was just a refresher on basic java concept such as inheritance, interfaces, abstract classes etc... The only tricky part I would say that in problem 3 which I wasn't familiar or knowledgeable of the issue at all but after I did some digging and research online the answer straight forward.

I have completed the assignment so it is a full submission. There is a table below showing code coverage for all my classes and all Junit tests

Element	Class	Methods	lines
ImprovedStringTokenizer	100% (1/1)	100% (5/5)	100% (12/12)
ImprovedStringTokenizerTest	100% (1/1)	100% (4/4)	100% (31/31)
ImporvedRandom	100% (1/1)	66% (2/3)	60% (3/5)
ImporvedRandomTest	100% (1/1)	100% (1/1)	100% (9/9)
problem5	100% (2/2)	100% (9/9)	100% (43/43)
problem4	100% (2/2)	75% (3/4)	85% (12/14)

The Cyclomatic complexity is 5