Today's tasks are all based on the following scenario.

You're responsible for designing an application to help a shipping & logistics company (think FedEX or USPS) with tracking. There are two types of items that can be shipped: mail, and packages. Both items have a source address, a destination address, a tracking number, a current location, a weight, and a shipping cost. Packages also have the size of the package and an insured value. The location of an item (regardless of whether it is mail or a package) can be either one of the company's shipping depots, or on a truck (which handles transportation to a new location; we'll assume for now that trucks don't have GPS which figures out real-time locations). Each item should keep track of which truck it is on (if it is located on a truck), if it's not on a truck, it should keep track of the depot it is at. When an item is placed on a truck, its location is updated accordingly. When a truck arrives at a depot, all items on it should be updated to the new depot location. A user of this application should be able to ship a new item, view the status of all of their items (their items being defined as ones that either they have shipped, or that have been shipped to them, or one for which they know the tracking number). It should also be possible to update the status of an item, marking it as either having been placed on a truck, or that the truck has arrived at a new depot, or finally that the package has been delivered to its destination.

Task 1:

Identify *at least three* promising classes for this system. For each class, identify which fields/attributes it will have (including a has-a relationship with other classes you've defined, such as the Infrastructure has-a (or has-multiple) Improvements from the slides).

Task 2:

Identify at least one promising inheritance (is-a) relationship for this system. Indicate which class is the superclass, and which class is the subclass. Indicate what extra data or features the subclass includes beyond what is in the superclass.

Task 3:

Identify *at least three* methods your classes will need (a total of three, not three per class) in order to fulfill the description above.