CSCD 211
Lab 9

Introduction: This assignment will simulate the activities of a delivery truck transporting packages from one city to another. Certain simplifying assumptions have been made to allow you to concentrate on the programming concepts of Java rather than the "complete" realism of the simulation.

Requirements: Create Truck, Package(abstract), Letter, Box, Crate(abstract), MetalCrate, and WoodCrate classes to simulate the long-distance transportation of packages by a truck.

Specifications: Your solution must conform to the following specifications:

- The **CSCD211Lab9** class contains the main and cannot be changed.
- The **Truck** class should contain the driver's name, the maximum number of packages it can carry, additional information about each package that it carries (via an array of Package references). Trucks will be assigned a driver, loaded with packages, then driven and unloaded. This class will contain the bulk of the code necessary for the tasks that must be performed in this program.

  It also must contain the following three methods:

  - load – This method loads the packages from the manifest to the truck. It will print all loading information to the log file, including tracking what packages are not loaded.
  - drive – This method is to simulate driving the truck. It will sort the packages by their natural order. It will also print the date and time the truck was loaded (ie the time the program was ran) to the log file as well. (Information on how to do this is available in both your book and the java API, look in the Date class).
  - unload – This method unloads the packages from the truck and prints the information from each package to the log file. It will also report the total number of packages transported and the total combined weight of those packages.
- **Package** is the base class for a 'packages' hierarchy -- you will never have something that is just a Package. Each package has a tracking number and weight. The tracking number encodes the type of package. All packages are Comparable, comparisons are done first by the name of the type of package (ie letter, box, metal crate etc) and then by weight.
- **Letter** is a type of Package. The least significant digit of the tracking number is a 0. The weight is given in ounces. The maximum weight is 24 ounces. Letters have a length and width associated with them that are given in inches. The maximum length or width of a letter is 18 inches.
- **Box** is a type of Package. The least significant digit of the tracking number is a 1. The weight is given in pounds. The maximum weight of a box is 100 pounds. Boxes have length, width, and height associated with them that are all given in inches. The maximum length, width, or height is 36 inches.
- **Crate** is a type of Package. Crates have length, width, and height associated with them that are all given in inches. You will never have just a Crate

- **Metal Crate** is at type of Crate. The least significant digit of the tracking number is a 2. The weight is given in pounds. The maximum weight of a metal crate is 500 pounds. The maximum length, width, or height is 60 inches. Metal Crates are used to ship cargo or live animals. You will need a designator in metal crate for what is being shipped.
- **Wooden Crate** is a type of Crate. The least significant digit of the tracking number is a 3. The weight is given in pounds. The maximum weight of a metal crate is 200 pounds. The maximum length, width, or height is 48 inches. Wooden Crates are used to ship cargo which could be fragile or not. You will need to keep track if the cargo is fragile.

Additional Guidelines:

- Any least significant digit not yet specified (ie 4-9) represents a package your truck does not carry. The package should not be loaded, but information about the unknown package type needs to be written to the log file (see example output below). For such packages, you are guaranteed there will be a 4 numbers following the package number.
- Data to test your program should be stored in an input file called **manifest.txt** whose format is specified as follows:
  Driver Name
  Maximum packages truck can carry (integer - guaranteed)
  Tracking Number (integer - guaranteed)
  Package Weight (integer)
  Package Length (integer)
  Package Width (integer)
  (possibly) Package Height (integer)
  (possibly) Package Contents (String)

  You are guaranteed the driver name will be the on the first line of the input file. You are guaranteed the second line of the file will contain the maximum number of packages the truck can carry. You are also guaranteed there will be at least one tracking number followed by the information associated with that particular package.

- All actions taken in loading the truck should be written to an output file called **log.txt**. The log file should contain a record of everything that happened with the truck and its packages.
- As you read a tracking number and prepare to "load" that package on the truck, you must check for packages that are too heavy or too large. For such packages, write to the output file the problem (package type, tracking number, weight, and dimensions) along with why it was not loaded. Of course, don't load that package.
- If the truck is full, write the information to the log file (package type, tracking number, weight, and dimensions) along with why the package was not loaded.
- If there are multiple errors, ie package is too large and the truck is full or the contents are invalid, then you need only print one of the reasons why the package was not loaded. Choose whatever is easiest.
- Hint: Try and avoid setting yourself up to need to place dummy data members in Classes. For instance, you can put a toString in Package that works for every subclass, but then you

would be forced to put a height, and contents in the Letter class. Instead define class specific toStrings in each subclass that add to their parent's toString.

- You will need to write a **FEW** helper methods not specified in my API. These methods will be of the proper visibility, meaning mostly protected, and a few public. You will be substantiated graded on how you write these methods and follow the rules of encapsulation so be careful!
- Hint: It may be beneficial to track if a *Package* is too large, say with a boolean data member.
- I have provided a jar file for opening the input file

--> NOTE: Your log file should essentially contain a history of what occurred during the execution of the program. It should specifically describe each thing that occurred

To Turn In:

A zip file containing

- your source files including package system
- the input file you used to test your program
- the output file created from running your program
- We should be able to unzip your file and compile and run your code

Named your last name first letter of your first name lab9.zip (Example: steinerslab9.zip)