Package Delivery System

A package consists of a unique, system created package ID (numeric), a receiver company name(alpha), delivery zone (A1 though Z9, see below for grid), the delivery date of the package (use the Date class from lecture), its weight and volume. We will not worry about full address, for our purposes the receiver company name and delivery zone will suffice for knowing where a package goes. Packages can either be regular delivery (anytime on the delivery date) or special delivery (delivered by the end of a specific military hour (9-16) deadline on the delivery date).

Your classes should have default and non-default constructors (must both call mutators to enforce encapsulation), accessor methods, mutator methods (with argument validation), and toString methods. For any attributes/arguments that are objects please use deep copy.

Default values: receiver company name "Default Company", delivery zone "A1", delivery date (use default Date constructor), weight 1, volume 1, time deadline 9

Delivery Zone Grid, rows A through Z, columns 1 through 9

A1	A2	A3	A4	A5	A6	A7	A8	A9
B1	B2	B3	B4	B5	B6	B7	B8	B9
C1	C2	C3	C4	C5	C6	C7	C8	C9
.	.	.						
.	.	.						
X1	X2	X3	X4	X5	X6	X7	X8	X9
Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9

A truck consists of a unique, system created truck ID (numeric), the weight of the packages on the truck, and the volume of the packages on the truck. Assume three truck types are available:

- Small - volume limit 1000, weight limit 2000
- Medium - volume limit 2000, weight limit 4000
- Large - volume limit 4000, weight limit 8000

The Package Delivery System needs to be able to assign package objects (read from an input file) to truck objects, with the overall goal of limiting the number of "truck hours" necessary to deliver the packages within the day/time limit of each package. A truck can stop at most 5 different companies (multiple packages delivered at each company is OK) in each hour within a single delivery zone for a standard 9am-5pm workday. A truck can also travel to an adjacent zone (above, below, left, right), but for each adjacent zone travelled to, you subtract one from the number of delivery stops possible in that zone. For example, one truck can make 30 delivery stops in the first 6 hours of a day in one zone, and then 8 delivery stops in the last two hours in an adjacent zone.

Each non "void main" class needs its own individual test program. Each non "void main" class needs its own individual exception class and should throw exceptions for any invalid action. Other classes need to catch these exceptions where appropriate and write out details to log.txt.

The “void main” for the PackageDeliverySystem class should prompt the user for an input file name, read all the data from the file (errors to log.txt), creates Trucks, optimally loads Packages on Trucks, and outputs the following:

- To the screen - output the number of trucks of each size used, and the total “truck hours” used. “truck hours” is equal to the total of (truck type small=1, medium=2, large=3) multiplied by (hours used for each truck) for all trucks used.
- To a file "deliveries.txt" - output for each truck, the truck information and the packages delivered by that truck (in the order delivered) and showing all package information.
- To a file “log.txt” - Any errors in Package, SpecialPackage, Date, and Truck methods should throw a user-defined exception specific to that class, and when caught the details of the exception should be written to a “log.txt” file.

Coding Requirements (80 points) - Resubmit your design for re-grading. You must submit all your java files including test programs for each non "void main" class.

1. (60 points) Package Delivery System functionality
    a. Input File reading (5 points)
    b. Package/SpecialPackage classes (10 points)
    c. Creating/managing collection of Package objects (10 points)
    d. Truck class (10 points)
    e. Implementation of your algorithm to optimally assign Packages to Trucks (20 points)
    f. Output to screen and files (5 points)
2. (5 points) Test programs for each non "void main" class
3. (5 points) Exception classes implemented for each non "void main" class
4. (10 points) One-Page Analysis - Please describe what you think is the optimal algorithm for placing packages on trucks with the overall goal of limiting the number of "truck hours" necessary to deliver the packages within the day/time limit of each package. Would your algorithm still be optimal if the number of packages delivered was predicted to grow by a factor of 10 in the next few years? Would we still need all three truck types?

Code and test your Package Delivery System. Each class needs its own individual test program. The Package, SpecialPackage and Truck classes need to all have their own exception classes and should throw exceptions for any invalid action. Other classes need to catch these exceptions where appropriate.