Lab 4 150 Points
You Ship
Readings 7-11

In our fourth lab of the semester we will create a Java application that uses **GUI** input (use **JOptionPane**) which enables a shipping company to enter package information in order to output final shipping costs for a single package.

The shipping clerk (**our user in this application**) can can choose to ship a package by air, truck or mail, as well as insure the package or not insure the package. Depending on the clerk's choices combined with package weight, total cost of shipping will vary.

The application will also need to handle potential exceptions such as entering a string when an integer is expected, etc.

Lab Parameters

We will use **Exercise 7 on pages 531-532** of the textbook as our starting point. Do use the table values for shipping rates and insurance costs in your application.

Make sure to get the customer's name as well. This is not discussed in the exercise.

You should also add specific Exception handling to your application as you deem appropriate. For example, if the application is expecting a number for the weight and the shipping clerk enters nothing or "heavy" you application should be able to recover.

However, do not automatically instantiate objects as the exercise directs. Instead allow the shipping clerk (our user) to set the parameters in terms of customer name, package weight, shipping method, and whether to add insurance. **Use the select parameters to determine from which superclass or subclass to instantiate your object.**

For this lab, you will **have three (3) classes**. They should be named as such (and are in the class repository):

UsePackage.java

Your main application class. You determine what logic and exceptions should be included here.

Package.java (superclass)
This class creates a package object according to the shipping cost table.

InsuredPackage.java (subclass)
This class extends the package object to create an insured package object according to the insurance cost table.

In terms of output, make sure to output the following via the **CLI**:

- name of customer
- weight of package
- shipping method
- cost of shipping method
- total cost of shipping
- cost of insurance (**only if selected**)
- total cost of shipping with insurance (**only if selected**)

Make sure to add some design and include your company's name and other important visial information in the output.

Challenge Parameters

For those who want to create a more robust application, consider the following in this order.

1. **[Moderate Challenge]** Instead of outputting via the CLI, output via the GUI.
2. **[Somewhat Difficult Challenge]** Maintain and calculate rates with a multidimensional array.
3. **[Difficult Challenge]** Allow for more than one package object to be created using an array of objects (not two separate objects) and displayed on one invoice. Allow for more than one insured package object to be created using an array of objects (not two separate objects) and displayed on one invoice.

It is better to have a completely coded, documented, and functional program without challenge parameters than it is to have a non-working or partially working lab with challenge parameters.

As a reminder: The tutor is instructed to not help with any challenges.

Important items to note

- Use NetBeans for this lab.
- Follow variable and file naming conventions.
- Use appropriate data types.
- Document every file that you create and/or change. You must explain the new concepts and approaches (e.g., classes, etc.) in great detail. Other items such as **System.out.println()** are older concepts and do not need to be documented in detail but should be noted.
- Make sure to discuss the various interactions among class files.

- Refer to the **Documentation Guide** at for guidance on comments and lab preparation.