**Core Java Practical Assessment**

## 1. Problem Statement

The Department of Highways of a particular county is installing a toll collection system on one of its major roads. Trucks pulling up to a tollbooth are required to pay a toll of $5 per axle plus $10 per half-ton of the truck's total weight. A display in the booth shows the toll receipts and the number of truck arrivals since the last collection.

You will design an object oriented program in Java that simulates the operation of the tollbooth.

## Sample Scenario

To aid understand the problem statement let us imagine how such a toll collection system might work. A toll agent sits in a tollbooth that is equipped with a computer screen and a bar code reader. When a truck arrives at the booth, the agent scans a bar code on the windshield of the truck; it contains information about the truck, including its number of axles. The weight of the truck is obtained by scanning a bar code contained on the bill of lading presented by the driver. The truck information and toll due are then displayed on the computer screen:

***Truck arrival - Axles: 5 Total weight: 12500 Toll due: $145***

When a button on the side of the screen is pressed, the booth's totals are displayed:

***Totals since last collection - Receipts: $205 Trucks: 2***

When the cash drawer is removed from its cradle, the following is displayed on the screen. The totals are displayed and then reset to zero:

***\*\*\* Collecting receipts \*\*\****

***Totals since the last collection - Receipts: $253 Trucks: 5***

This scenario will be simulated by the object oriented system that you will build.

## Primary Objects in the Problem Statement

The primary objects of the tollbooth model problem are the relevant noun phrases of the problem statement. In this case we list: *trucks, tollbooth, axle, weight, receipts.* Of these, tollbooth and truck are the most important. Axles and weight are properties of a truck, and receipts are secondary to the tollbooth. Therefore, we chose truck and tollbooth as our primary types.

## Trucks Behaviors and Attributes

Toll depends on the number of axles and weight of a truck. Tollbooths need to get both of these from a truck. Therefore, a truck should keep track of the number of *axles* it has, its *total weight*, and allow other objects to get access to these but not modify these attributes. In addition a truck is of a particular *make*.

## Tollbooth Behaviors and Attributes

The main behavior required of a tollbooth is the ability to *calculate the toll* due. We also want to be able to *display the data* for the total receipts and number of trucks since the last receipt collection. This implies that these totals can be reset by the tollbooth  *on receipt of collection.*

## Directions to complete the Assessment

1. Create and compile the two interfaces (*Truck* and *TollBooth)*
2. Create two classes that implement the specified interface Truck. Each class should be named after a truck make of your choice (obviously the two truck makes must be different).   
   These classes should have the member variables (attributes) identified earlier. Carefully choose the access level modifier and make sure that you take into account the fact that some variables change over time and others don't.
3. Create a class that implements the specified interface TollBooth. For example: **AlleghenyTollBooth**  
   This class keeps track of a) the total number of trucks that have gone through the tollbooth and b) total receipts since collection.  
   The methods of your class specified in the interface will at least perform the following:a) display the booth's totals simply by printing those values to System.out and b) display the totals maintained by the booth and then reset them to zero because receipts and truck. Remember that totals are maintained only since the most recent collection. This corresponds to the supervisor of the tollbooth, emptying the cash box in the booth and resetting the meters.

## Using the tollbooth

The following represents sample usage code for the classes above. You may use it to test your overall implementation.

Use the class names that you have defined for your implementations of the truck interface. Also, the method names on your tollbooth may be called differently. You may also have different types of constructors depending on how much initialization of variables you choose when instantiating a class.

Class TestTollBooth {

Public static void main(String [] args){

TollBooth booth = new AlleghenyTollBooth();

Truck ford = new FordTruck(5, 12000); // 5 axles and 12000 kilograms

Truck nissan = new NissanTruck(2, 5000); // 2 axles and 5000kg  
 Truck daewoo = new DaewooTruck(6, 17000); // ….

booth.calculateToll(ford);

booth.displayData();  
 booth.calculateToll(nissan);

}

}

**2. Java Program to find smallest and largest element in an Array.**

**3. Java Program to get the following output.**

\*  
\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*  
\* \* \*  
\* \*  
\*