CSIS215 – Project – Fall 2023

In this project we will simulate the work of an application that manages the deliveries made by a delivery company.

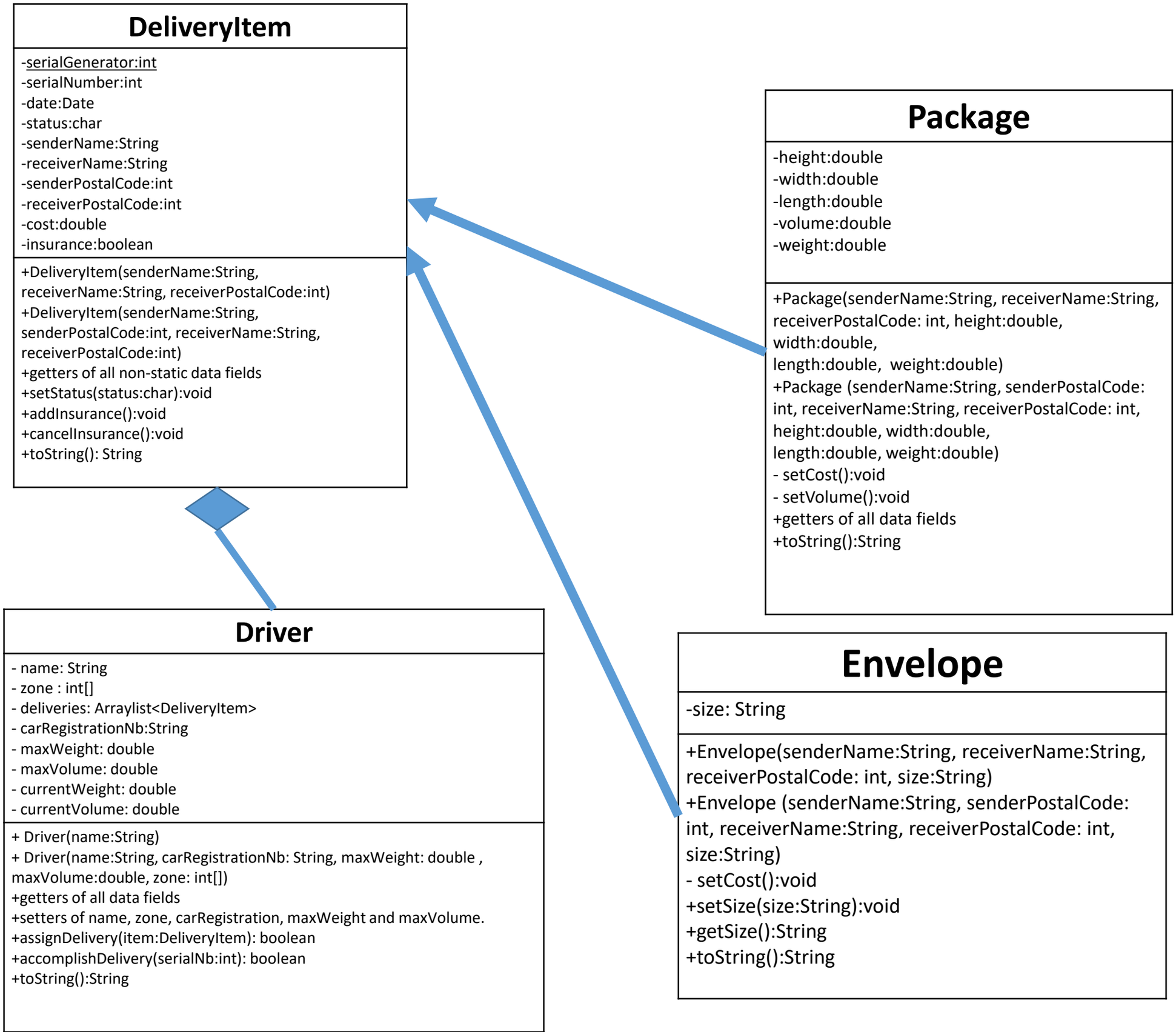
The company has multiple drivers, each using his car to deliver items. And each driver is able to cover a specific zone.

The company will receive items (packages and envelopes) from its client and assign the delivery to one of its drivers.

Following are the description of classes and their data members, needed for this application.

Implement all the following classes, respecting standards and java coding conventions. Then write the driver program that is used to manage the activities of the company.

Note: Your application must be user friendly and your code must respect java coding convention and well documented.


DeliveryItem
-serialGenerator:int
-serialNumber:int
-date:Date
-status:char
-senderName:String
-receiverName:String
-senderPostalCode:int
-receiverPostalCode:int
-cost:double
-insurance:boolean
+DeliveryItem(senderName:String, receiverName:String, receiverPostalCode:int)
+DeliveryItem(senderName:String, senderPostalCode:int, receiverName:String, receiverPostalCode:int)
+getters of all non-static data fields
+setStatus(status:char):void
+addInsurance():void
+cancelInsurance():void
+toString(): String
Package
-height:double
-width:double
-length:double
-volume:double
-weight:double
+Package(senderName:String, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double)
+Package (senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double)
- setCost():void
- setVolume():void
+getters of all data fields
+toString():String
Driver
- name: String
- zone : int[]
- deliveries: Arraylist<DeliveryItem>
- carRegistrationNb:String
- maxWeight: double
- maxVolume: double
- currentWeight: double
- currentVolume: double
+ Driver(name:String)
+ Driver(name:String, carRegistrationNb: String, maxWeight: double , maxVolume:double, zone: int[])
+getters of all data fields
+setters of name, zone, carRegistration, maxWeight and maxVolume.
+assignDelivery(item:DeliveryItem): boolean
+accomplishDelivery(serialNb:int): boolean
+toString():String
Envelope
-size: String
+Envelope(senderName:String, receiverName:String, receiverPostalCode: int, size:String)
+Envelope (senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, size:String)
- setCost():void
+setSize(size:String):void
+getSize():String
+toString():String

DeliveryItem Class

DeliveryItem
-serialGenerator:int -serialNumber:int -date:Date -status:char -senderName:String -receiverName:String -senderPostalCode:int -receiverPostalCode:int -cost:double -insurance:boolean
+DeliveryItem(senderName:String, receiverName:String, receiverPostalCode:int) +DeliveryItem(senderName:String, senderPostalCode:int, receiverName:String, receiverPostalCode:int) +getters of all non-static data fields +setStatus(status:char):void +addInsurance():void +cancelInsurance():void +toString(): String

Class Delivery Item

A- Data Fields:

1. serialGenerator: static int used to generate a new, unique, and automatically incremented serial number. serialGenerator must start at 1000 and get incremented for each new item. When a new object is created the serialGenerator will provide the correct serial number for the object.
2. serialNumber: the serial number of the item. Serial number is set to a new serial number when the item is created using the serialGenerator.
3. date: the date of creation of the item (the date when the item is received)
4. status: the status of the delivery: 'R' for received, 'A' for assigned to a driver, and 'D' for delivered. Default is 'R'
5. senderName: the name of the sender. Default is null.
6. receiverName the name of the receiver. Default is null.
7. senderPostalCode: the postal code of the sender. Default is 10000.
8. receiverPostalCode: the postal code of the receiver. Default is 10000.
9. cost: the cost of delivery. Default is 2.
10. insurance: true if the sender wants to have insurance on his item and false otherwise. Default is false

B. Methods:

1. DeliveryItem(senderName:String, receiverName:String, receiverPostalCode: int): creates a new object with the given sender name, receiver name and postal code and sets all other values to default.
2. DeliveryItem(senderName:String, senderPostalCode: int, receiverName: String, receiverPostalCode: int) creates a new object with the given sender name and postal code, receiver name and postal code and sets all other values to default.
3. getters of all non-static data fields
4. setStatus(status:char): sets the status of the DeliveryItem object.
5. addInsurance(): sets insurance to true and update the cost by adding 3$.
6. cancelInsurance(): sets insurance to false and update the cost by deducing 3$.
7. toString(): return a string object with all the information about the delivery Item in the following format.

   [SerialNumber] – [Date]
   Sender : [name] – [Postal Code]
   Receiver: [name] – [Postal Code]
   [With / without insurance]
   Status: [Receiver / Assigned / Delivered]

Subclasses of Delivery Item

Envelope
-size: String
+Envelope(senderName:String, receiverName:String, receiverPostalCode: int, size:String) +Envelope (senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, size:String) - setCost():void +setSize(size:String):void +getSize():String +toString():String

Package
-height:double -width:double -length:double -volume:double -weight:double
+Package(senderName:String, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double) +Package (senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double) - setCost():void - setVolume():void +getters of all data fields +toString():String

Class Envelope

A- Data Fields:

1. size: represents the size of the envelope. Possible values: “A2”, “A6” , “A7” , “A9”, ”3 square” , “4 square” , “5 square”. Default is “A2”.

B. Methods:

1. Envelope(senderName:String, receiverName:String, receiverPostalCode: int, size:String): creates a new object with the given sender name, receiver name and postal code and size and sets all other values to default.
2. Envelope(senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, size:String): creates a new object with the given sender name and postal code, receiver name and postal code and size and sets all other values to default
3. setCost():private method that sets the cost of the envelope’s delivery according to the sender and receiver address, size and insurance – check the table below – The cost should be set at the moment of creation of a new envelope object. The cost must be updated if needed when the data fields of an envelope object are changed.
4. setSize(size:String): sets the size of the envelope and update the cost.
5. getSize(): returns the size of the envelope.
6. toString():returns a string with all information about the envelope.

Envelope: [serialNumber] – [date]
Size : [size]
Sender : [name] – [Postal Code]
Receiver: [name] – [Postal Code]
[With / without insurance]
Status: [Receiver / Assigned / Delivered]
Cost: [Cost]

Type	Price
A2	2.0
A6	1.6
A7	1.5
A9	1.2
4 square	1.8
5 square	1.6
Add insurance	3.0

Class Package

A- Data Fields:

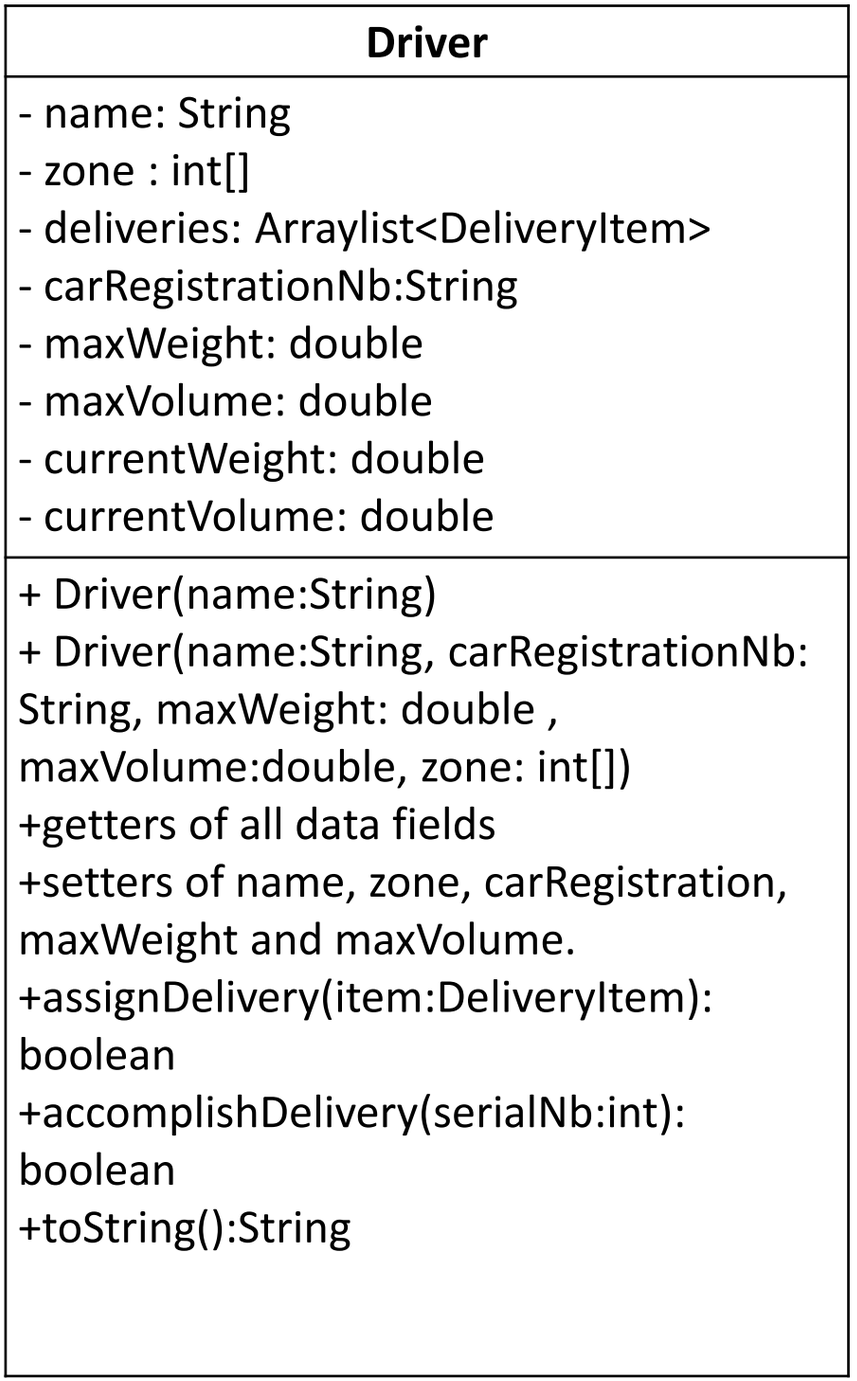
1. height: represents the height of a package. Default is 0.2
2. width: represents the width of a package. Default is 0.2
3. length: represents the length of a package. Default is 0.2
4. volume: represents the volume of a package. The volume must be updated whenever one of the dimensions is set of updated
5. weight: represents the weight of a package. Default is 0.1

B. Methods:

1. Package(senderName:String, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double)
2. Package(senderName:String, senderPostalCode: int, receiverName:String, receiverPostalCode: int, height:double, width:double, length:double, weight:double)
3. setCost():private method that sets the cost of the package's delivery according to the sender and receiver address, weight, volume and insurance – check the table below – The cost should be set at the moment of creation of a new envelope object. The cost must be updated if needed when the data fields of an envelope object are changed.
4. setVolume(): private method that sets the volume of the package according to its dimensions. The method should be called whenever the volume is needed to be set.
5. getters of all data fields
6. toString():returns a string with all information about the envelope.

Package: [serialNumber] – [date]
Dimension: [height]*[width]*[length]
Volume:[volume] – Weight : [weight]
Sender : [name] – [Postal Code]
Receiver: [name] - [Postal Code]
[With / without insurance]
Status: [Receiver / Assigned / Delivered]
Cost: [Cost]

Condition	Price
Volume <=2	2.0+3*weight
2 < Volume <=5	2.8 + 3*weight
Volumer>5	2.8 + 3*weight +5*(Volume-5)
Add insurance	20*Weight


Driver
- name: String
- zone : int[]
- deliveries: Arraylist<DeliveryItem>
- carRegistrationNb:String
- maxWeight: double
- maxVolume: double
- currentWeight: double
- currentVolume: double
+ Driver(name:String)
+ Driver(name:String, carRegistrationNb: String, maxWeight: double , maxVolume:double, zone: int[])
+getters of all data fields
+setters of name, zone, carRegistration, maxWeight and maxVolume.
+assignDelivery(item:DeliveryItem): boolean
+accomplishDelivery(serialNb:int): boolean
+toString():String

Class Driver

A- Data Fields:

1. name: represents the driver's name
2. zone : int[] array of 2 integer representing the zone covered by the driver . Example if zone equals [30000, 50000] it means that the driver covers the zone of all postalCode between 30000 and 50000. Default is [10000-50000]
3. deliveries: Arraylist containing all the items assigned to this driver. When an item is assigned to the driver, it must be added to this list. When the delivery is completed the item must be removed from the list
4. carRegistrationNb: represents the registration number of the driver's car. Default is empty string.
5. maxWeight: represents the maximum weight that the driver can handle in the same time. If the maximum weight is reached the driver can not accept any more delivery items. Default is 500.
6. maxVolume: represents the maximum volume that the driver can handle in the same time. If the maximum volume is reached the driver can not accept any more delivery items. Defaut is 50.
7. currentWeight: represents the total weight of all items currently handled by this driver.
8. currentVolume: represents the total weight of all items currently handled by this driver.

B. Methods:

1. Driver(name:String): creates a new driver with the given information and sets all other values to default.
2. Driver(name:String, carRegistrationNb: String, maxWeight: double , maxVolume:double, zone:int[]): creates a new driver with the given informations and sets all other values to default.
3. getters of all data fields
4. setters of name, zone, carRegistration, maxWeight and maxVolume.
5. assignDelivery(item:DeliveryItem): assign a new delivery item to the driver. Returns true if assignment is successful and false otherwise. Assigning a new delivery must respect the zone of delivery and insure the maximum weight and volume are not exceeded.
6. accomplishDelivery(serialNb:int): removes a delivery item from the list of assigned items of this driver and returns true. If item is not found returns false.
7. toString(): returns a string with all information about this driver in the following format:

   [name]
   Active Zone: [zone]
   Car: [carRegistrationNb]
   Max Weight: [max Weight] – Max Volume [Max Volume]

Driver Program:

Write the main class that will create an Arraylist for drivers. And an ArrayList for all delivery items.
The program displays and executes the functions of the following menu:

1. **Add new driver.**
2. **Receive new item.**
3. **Display all items (received - assigned – delivered)**
4. **Display all received items.**
5. **Display all assigned items.**
6. **Display all delivered items.**
7. **Display all driver.**
8. **Assign item to a driver**
9. **Set item to received**
10. **Check a driver load**
11. **Display total cost**

Examples for drivers:

Driver1:

John Doe
Active Zone: [10000-20000]
Car: A23948
Max Weight: 700 – Max Volume 150

Driver2:

George Attallah
Active Zone: [25000-35000]
Car: F03941
Max Weight: 200 – Max Volume 120

Examples for envelopes:

Envelope 1:

Envelope: 1000 – Mon Nov 13 14:00:00 GMT 2023
Size : A2
Sender : Rimas Doe – 10000
Receiver: Loulou Doe – 15200
With insurance
Status: Receiver
Cost: 5.0$

Envelope 2:

Envelope: 1001 – Mon Nov 13 18:38:08 GMT 2023
Size : 4 square
Sender : Loulou Doe – 15200
Receiver: Rimas Doe – 10000
Without insurance
Status: Assigned
Cost: 1.8$

Examples for packages:

Package1:

Package: 1003 – Mon Nov 13 14:50:02 GMT 2023
Dimension: 0.2 *0.2 * 1
Volume: 0.04 – Weight : 2
Sender : Rimas Doe – 10000
Receiver: Loulou Doe – 15200
Without insurance
Status: Receiver
Cost: 8.0$

Package2:

Package: 1004 – Mon Nov 13 15:01:00 GMT 2023
Dimension: 0.8 *0.2 * 3
Volume: 0.48 – Weight : 5
Sender : Rimas Doe – 10000
Receiver: Loulou Doe – 15200
With insurance
Status: Delivered
Cost: 117$