**Functions**

* obtain vehicle manifest
* obtain list of packages scheduled for pickup
* scan on delivery, on acceptance and pickup

**Use Cases**

**Vehicle Manifest**

The driver indicates they wish to create a manifest of packages in their vehicle. After the driver logs in to the systems they are presented with a list of packages. The packages are found by sending the drivers vehicle id to the database (Data Adapter) which returns all the bins that are in the driver’s vehicle. The packages from the individual bins are then displayed on the list. The driver then has the ability to create a file of these packages that displays the same information that the list does.

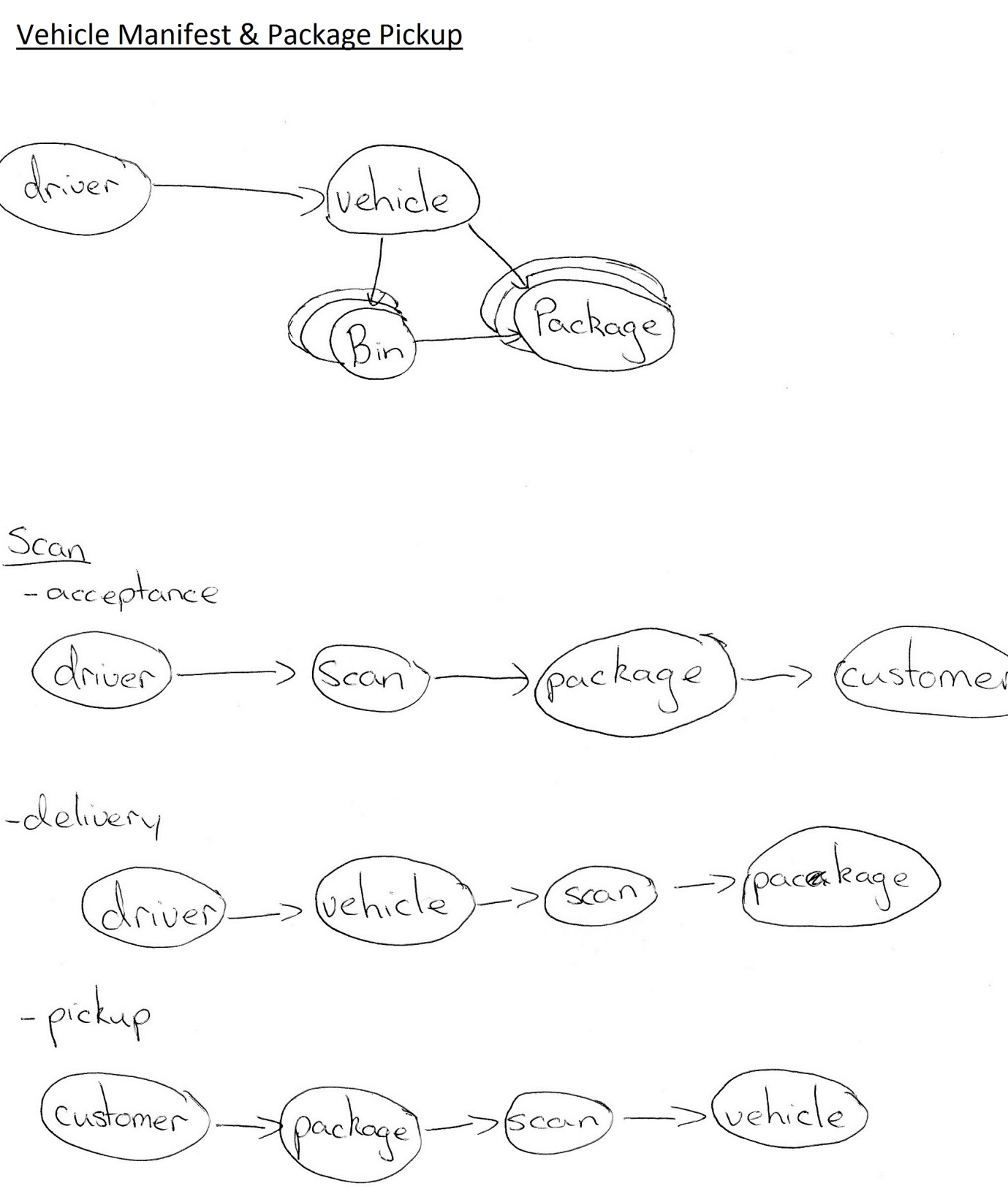
**Package Pick up**

The driver indicates that they wish to get a list of all the packages scheduled for pickup. After logging on the program sends a command to the database (Data Adapter) which returns all packages that have been marked with “awaiting pickup”. The driver has the ability to create a file of this information as well. The information of the packages to pick up is placed below any package to be delivered by the driver in both the list and file.

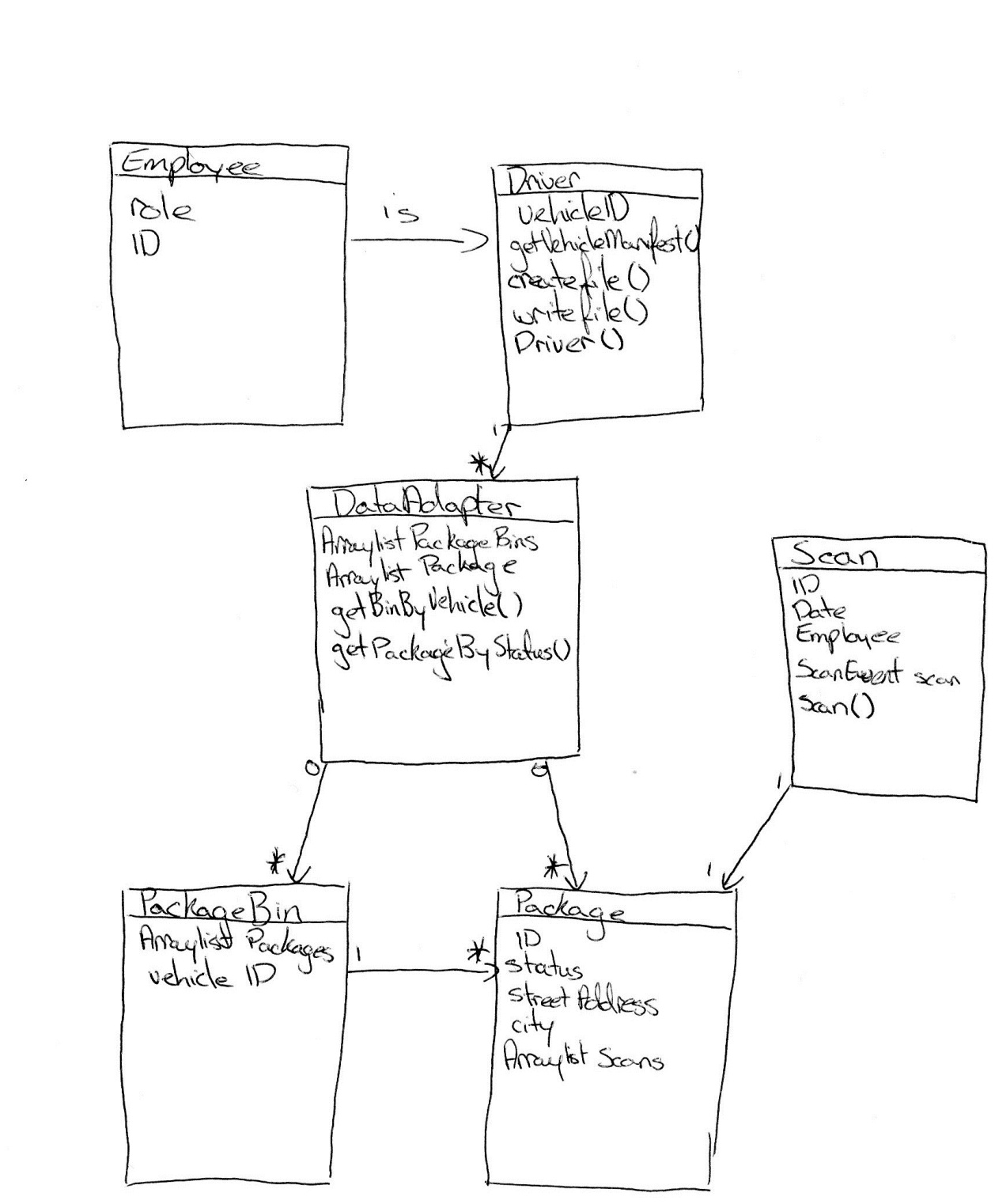
**Scan on delivery, acceptance and pickup**

Upon removal from the vehicle the package is scanned to indicate the driver has attempted to deliver the package. Before giving the package to the recipient the package is scanned again to indicate it has been successfully delivered. If a package is to be picked up, once the driver has gotten the package from the client they will then scan the package to indicate the package has been successfully picked up. This information is then recorded in the packages log. After each scan the information is recorded in the packages log.

**Object Models**



**Class Diagram**



**Missing Requirements**

* Database – this would be easier to keep track of all required information, but since we are only interested in the information it returns it is not needed here.
* Scanning Device – allowing for packages to be scanned.

**Testing Plan**

**General Testing**

* Make sure all variables are not null
* Make sure all number id’s are greater than 0

**Manifest & Pickup Testing**

* Check returned lists aren’t null
* Check PackageBins are returned by vehicleID
* Check Packages are returned by pickup status

**Process**

We started this project by looking at the brief and trying to determine which the main functions that were required. We then broke these functions down into sub functions. Then next step was to come up with a general idea of how the functions worked and interacted with each other, this was done by creating object models that we thought represented the overall system. Out of this process we were able to determine what would be required by the system the only problem we encountered now was how to store the relevant information about all the clients, employees, packages and bins. We came up with two options the first was to implement a simple database to store the information, but would require a lot of work. The second option was to only worry about the information we would be getting from the database and store this in an easy to access way for example using a group of array lists that stored all the relevant information. We decided that the second option would be the best as we were only interested in what the database would return.

For my part of the project I was interested only in what the driver was doing and what they needed to accomplish their tasks. The process of creating a vehicle manifest and a list of packages for pickup were very similar. This was done by using bins and packages. Since the bins held a list of packages in them and a vehicle identifier it was easy to gather the vehicle manifest by collecting up all the bins in the vehicle and getting their packages. For the package pickup the package class would hold a variable that indentified whether it was needed to be picked up. For the scanning processes the driver makes a list of different scan events were created these were then added to the package information every time a scan occurred.

There are several things that could have helped us to improve the project and its implementation. One of which was to have a database attached to the program, this would allow us to store and retrieve our data and information a lot easier with large volumes of clients and packages than our implementation of the data adapter. Another thing that could of helped was more documentation of junit testing as it seemed poorly documented and we had to search internet blogs for the proper way of doing this.

As a group we had very little problems. The only problem we had was that one of our group members was a bit difficult to motivate, this did not impact the rest of the work the group was doing. Since his part of the project was dealing with the client information and UI, this was pretty much separate from the rest of the group so the impact on the rest of us was minimal.