# Term 4 Assignment 2

## Quicka Self-driving

You are the Lead Systems Engineer for the Quicka ridesharing platform and are in charge of developing all the necessary software for drivers, riders, and self-driving cars.

For this assignment, you are asked to write the Control Operation System (aka COS-101) software for the Quicka Autonomous Vehicle. The computers onboard the QAV are outfitted with a JVM capable of running Java code.

When designing and writing your systems, you may use any IDE of your choice, but you must ensure that your code is able to run on the JVM.

### Assignment 4.2: Traffic Engineering

For this assignment you may use the supplied `RunAssignment.java` file to test your code. However, this is only a small example, and you will need to test your code more thoroughly yourself.

#### Task 1: Traffic Control

We need to design software for the tracking of all the self-driving vehicles across the world. To do this we will use the principle of Object Oriented Design to develop classes that represent our different types of vehicles.

#### Instructions

1. Create a class called `BasicVehicle` that has the following:
    - private attributes called `latitude`, `longitude`, and `altitude` that are of type double
    - a private attribute called `registration` of type String
2. Create a default constructor for the class that:
    - sets `registration` to "unknown"
    - `altitude`, `longitude`, and `latitude` to 0.0
3. Overload the default constructor to take a parameter for each attribute and set it.
4. Write getter and setter methods for each of the attributes

5. Write a public method called `printInfo()` that prints out the state of the object (see below)
6. Write a public method called `distanceTo()` that takes a `BasicVehicle` parameter and returns the straight line distance between the two spacecraft as a double

The straight line distance can be calculated as:

```
distance = squareRoot( (A.x - B.x)^2 + (A.y - B.y)^2 + (A.z - B.z)^2 )
```

Hint: You may want to import `java.lang.Math.*` and use `Math.sqrt` and `Math.pow`. See java.lang.Math (https://docs.oracle.com/en/java/javase/14/docs/api/java.base/java/lang/Math.

**Sample Outputs** Method `printInfo()`:

```
Registration: ZS-CS1
Current position:
Alt: 18000.0
Lat: 22.0
Long: 31.0
```

**Task 2: Delivery Vehicle**

We want to develop a delivery vehicle that can carry a limited number of equally weighted boxes. To do this we can build off of our previous task by extending the `BasicVehicle` class.

**Instructions**

1. Create a new class called `DeliveryVehicle` that is a subclass of `BasicVehicle`:
    - a private attribute `cargoList` of type `String[]`
    - a private attribute `numItems` of type `int`
2. Create a constructor that takes two parameters: a String for `registration`, and an int for `maxCapacity`:
    - pass on the `registration` parameter to the super constructor along with 0.0 for altitude, longitude and latitude.
    - use the `maxCapacity` parameter to create a new array for `cargoList`
3. Create a boolean method called `isFull` that returns if the `cargoList` is full.
4. Create a boolean method called `isEmpty` that returns if there are no boxes loaded.
5. Create a method called `loadCargo` that takes a String parameter `description` and adds the cargo description to the cargo list (if it is not already full).
6. Override the `printInfo()` method to print out the cargo list in addition to the BasicVehicle info or “No cargo yet.” if the list is empty.

Hint: The `cargoList` should be treated as a partially filled array using the `numItems` attribute.

**Sample Input/Output**

The following class can be used to test your assignment and produce the output below

```
public class RunAssignment {

  public static void main(String[] args) {

    BasicVehicle vehicleCS1 = new BasicVehicle();
    vehicleCS1.setRegistration("ZU-CS1");
    vehicleCS1.setAltitude(18000.0);
    vehicleCS1.setLatitude(22.0);
    vehicleCS1.setLongitude(31.0);

    vehicleCS1.printInfo();

    BasicVehicle vehicleCOS = new BasicVehicle("ZU-COS", 10.0, 18.0, 19000.0);

    vehicleCOS.printInfo();

    System.out.println("Distance between vehicles: " +
                       vehicleCS1.distanceTo(vehicleCOS));

    DeliveryVehicle deliveryCA1 = new DeliveryVehicle("ZU-CA1", 5);
    deliveryCA1.printInfo();
    System.out.println("Delivery vehicle is empty: " + deliveryCA1.isEmpty());
    System.out.println("Delivery vehicle is full: " + deliveryCA1.isFull());
    deliveryCA1.loadCargo("50000 units paracetamol 500mg");
    deliveryCA1.loadCargo("2000 units melatonin 5mg");
    deliveryCA1.printInfo();

    System.out.println("Delivery vehicle is empty: " + deliveryCA1.isEmpty());
    System.out.println("Delivery vehicle is full: " + deliveryCA1.isFull());
  }
}
```

Output:

```
Registration: ZU-CS1
Current position:
Alt: 18000.0
Lat: 22.0
Long: 31.0
```

```
Registration: ZU-COS
Current position:
Alt: 19000.0
Lat: 10.0
Long: 18.0
Distance between vehicles: 1000.1564877557911
Delivery vehicle: ZU-CA1
Max capacity: 5
No cargo yet.
Delivery vehicle is empty: true
Delivery vehicle is full: false
Delivery vehicle: ZU-CA1
Max capacity: 5
50000 units paracetamol 500mg
2000 units melatonin 5mg
Delivery vehicle is empty: false
Delivery vehicle is full: false
```

## Submission Instructions

Submit only your files `BasicVehicle.java` and `DeliveryVehicle.java` in a compressed `.zip` folder labeled with your student number an underscore and `assignment5`.

For example if your student number is `12345` submit a file `12345_assignment5.zip` that contains the files `BasicVehicle.java` and `DeliveryVehicle.java`.

Late submissions incur a 10% penalty per day maximum of 5 days late.

## Marking

| Section | Marks |
|---|---|
| Task 1: Traffic Control | 22 |
| Task 2: Delivery Vehicle | 20 |
| **Total** | **42** |