# Lab 08
# Fleet of Vehicles

**Objective:**

Use classes and inheritance to create a vehicle system managing system.

**Lab Solution**

**Requirements:**

- Functionality. (80pts)
  - No Syntax Errors. (80pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
  - Set-Up the Project (5pts)
    - Include the tester code in your project.
    - Do not alter the provided code.
  - Write a class file called **Vehicle**
    - Instance Variables
      - Manufacturer's name: a non-null String that represents the manufacturer of the vehicle. Default value is "none".
      - Number of Cylinders: a non-zero, positive whole number value representing the number of cylinders. Default value is 6.
      - Owner's name: a non-null String that represents the owner's name of the vehicle. Default value is "none".
    - Constructors
      - Default: Must set all properties to their default values mentioned in the "Instance Variables" section.
      - Parameterized: Must take in a parameter for each instance variable in the order named above. This means the first instance variable is the first parameter, the second instance variable is the second parameter, and so on. This must set the instance variable values only if the given values are valid, but otherwise it must set the instance variables to their default values.
    - Methods
      - Accessors and Mutators for the instance variables
        - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
        - If the value that is being set is not valid, then set the instance variable to its default value.
      - Equals: This method takes in another instance of Vehicle and only returns true if all of the instance variables match.
      - ToString: This method returns a String with all of the instance variable values concatenated together with the format:

Manufacturer's Name: <<Manufacturer's Name>>\n
Number of Cylinders: <<Number of Cylinders>>\n
Owner's Name: <<Owner's Name>>\n

Where values in "<<>>" correspond to the instance variable values.

- All above must apply for full credit.

- Write a class file called **Truck** that inherits from **Vehicle**
  - Instance Variables
    - Load Capacity: A non-negative decimal value that represents number of tons the Truck can handle. Default value is 1.0.
    - Towing Capacity: A non-negative decimal value that represents the number of tons the Truck can move. Default value is 1.0.
  - Constructors
    - Default: Must set all properties to their default values mentioned in the "Instance Variables" section.
      - Must also set Vehicle's Default values as well.
    - Parameterized: Must take in a parameter for each instance variable in **Vehicle** and THEN **Truck** in the orders named above. This means the first instance variable is the first parameter, the second instance variable is the second parameter, and so on. This must set the instance variable values only if the given values are valid, but otherwise it must set the instance variables to their default values.
      - Must also set Vehicle's Parameterized values as well.
  - Methods
    - Accessors and Mutators for the instance variables
      - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
      - If the value that is being set is not valid, then set the instance variable to its default value.
    - Equals: This method takes in another instance of Truck and only returns true if all of the instance variables, both Vehicle and Truck, match.
    - ToString: This method returns a String with all of the instance variable values concatenated together with the format:

Manufacturer's Name: <<Manufacturer's Name>>\n
Number of Cylinders: <<Number of Cylinders>>\n
Owner's Name: <<Owner's Name>>\n
Towing Capacity: <<Towing Capacity>>\n
Load Capacity: <<Load Capacity>>\n

Where values in "<<>>" correspond to the instance variable values.

- All above must apply for full credit.

- Write a class file called **Car** that inherits from **Vehicle**
  - Instance Variables
    - Gas Mileage: A non-negative decimal value that represents number of gallons of gas per mile. Default value is 1.0.

- Number of Passengers: A non-negative whole number value that represents the number of passengers the car can contain. Default value is 1.

- Constructors
  - Default: Must set all properties to their default values mentioned in the "Instance Variables" section.
    - Must also set Vehicle's Default values as well.
  - Parameterized: Must take in a parameter for each instance variable in **Vehicle** and THEN **Car** in the orders named above. This means the first instance variable is the first parameter, the second instance variable is the second parameter, and so on. This must set the instance variable values only if the given values are valid, but otherwise it must set the instance variables to their default values.
    - Must also set Vehicle's Parameterized values as well.
- Methods
  - Accessors and Mutators for the instance variables
    - Make sure in the mutators check for valid values named in the "Instance Variables" Section.
    - If the value that is being set is not valid, then set the instance variable to its default value.
  - Equals: This method takes in another instance of Car and only returns true if all of the instance variables, both Vehicle and Car, match.
  - ToString: This method returns a String with all of the instance variable values concatenated together with the format:

Manufacturer's Name: <<Manufacturer's Name>>\n
Number of Cylinders: <<Number of Cylinders>>\n
Owner's Name: <<Owner's Name>>\n
Gas Mileage: <<Gas Mileage>>\n
Number of Passengers: <<Number of Passengers>>\n

Where values in "<<>>" correspond to the instance variable values.

- All above must apply for full credit.

**Example Dialog:**

```
Welcome to the fleet manager
Enter 1: to add a Vehicle
Enter 2: to remove a Vehicle
Enter 9 to quit
1
Enter 1: if it is a car
Enter 2: if it is a truck
Enter 3: if it is unclassified
1
```

```
Enter the manufacturer's name
Nissan
Enter the number of cylinders
6
Enter the owner's name
QWERTY
Enter the car's gas mileage
29
Enter the number of passengers
5
The Fleet currently
Manufacturer's Name: Nissan
Number Of Cylinders: 6
Owner's Name: QWERTY
Gas Mileage: 29.0
Number of Passengers: 5

Enter 1: to add a Vehicle
Enter 2: to remove a Vehicle
Enter 9 to quit
1
Enter 1: if it is a car
Enter 2: if it is a truck
Enter 3: if it is unclassified
2
Enter the manufacturer's name
Chevy
Enter the number of cylinders
8
Enter the owner's name
ASDF01
Enter the truck's load capacity
1
Enter the truck's towing capacity
2
The Fleet currently
```

```
Manufacturer's Name: Nissan
Number Of Cylinders: 6
Owner's Name: QWERTY
Gas Mileage: 29.0
Number of Passengers: 5

Manufacturer's Name: Chevy
Number Of Cylinders: 8
Owner's Name: ASDF01
Towing Capacity: 2.0
Load Capacity: 1.0

Enter 1: to add a Vehicle
Enter 2: to remove a Vehicle
Enter 9 to quit
1
Enter 1: if it is a car
Enter 2: if it is a truck
Enter 3: if it is unclassified
3
Enter the manufacturer's name
Ford
Enter the number of cylinders
6
Enter the owner's name
ASDF02
The Fleet currently
Manufacturer's Name: Nissan
Number Of Cylinders: 6
Owner's Name: QWERTY
Gas Mileage: 29.0
Number of Passengers: 5

Manufacturer's Name: Chevy
Number Of Cylinders: 8
Owner's Name: ASDF01
```

```
Towing Capacity: 2.0
Load Capacity: 1.0

Manufacturer's Name: Ford
Number Of Cylinders: 6
Owner's Name: ASDF02

Enter 1: to add a Vehicle
Enter 2: to remove a Vehicle
Enter 9 to quit
2
Enter 1: if it is a car
Enter 2: if it is a truck
Enter 3: if it is unclassified
2
Enter the manufacturer's name
Chevy
Enter the number of cylinders
8
Enter the owner's name
ASDF01
Enter the truck's load capacity
1
Enter the truck's towing capacity
2
The Fleet currently
Manufacturer's Name: Nissan
Number Of Cylinders: 6
Owner's Name: QWERTY
Gas Mileage: 29.0
Number of Passengers: 5

Manufacturer's Name: Ford
Number Of Cylinders: 6
Owner's Name: ASDF02
```

```
Enter 1: to add a Vehicle
Enter 2: to remove a Vehicle
Enter 9 to quit
9
The Fleet currently
Manufacturer's Name: Nissan
Number Of Cylinders: 6
Owner's Name: QWERTY
Gas Mileage: 29.0
Number of Passengers: 5

Manufacturer's Name: Ford
Number Of Cylinders: 6
Owner's Name: ASDF02

Goodbye
```

**Solution Tests:**

1. Is your name written as a comment in all source files?
2. Does the solution compile (no syntax errors)?
3. Does your output match the example dialog?

**Lab Report**

1. Create a section named “Problem” and describe this lab’s problem in your own words. (10pts).
2. Create a section named “Solution Description” and describe how the code solves the problem in your own words. (10pts).
3. Create a section named “Problems Encountered” and describe the various syntax, run-time, and logic errors that were encountered while implementing the solution. (10pts).
4. Describe inheritance in OOP. (10pts).
5. When creating a class, what reserved word in Java creates the inheritance, “is a” relationship? (10pts).
6. When calling a method or accessing a property from an inherited / “parent” class, what reserved word in Java must be called? (10pts).
7. Is it possible to call a private method or access a private property from an inherited class? (10pts).
8. In Java, is it possible to inherit from multiple classes? (10pts).
9. Assuming that we have Classes “ParentClass” and “ChildClass”, where “ChildClass” inherits from “ParentClass”. Would the following code work without error? If there is an error explain why and how it can be fixed.

```
ChildClass child = new ChildClass();
ParentClass parent = child;
```

10. Assuming that we have Classes “ParentClass” and “ChildClass”, where “ChildClass” inherits from “ParentClass”. Would the following code work without error? If there is an error explain why and how it can be fixed.

```
ParentClass parent = new ParentClass();
ChildClass child = parent;
```

**Finally:**
Upload the source code (.JAVA File Extension) and written lab report (.DOC, .DOCX, or .PDF file extension) to the CSCE Dropbox.