

```
/**
 * The Car class
 * @author Richard White
 * @version 2018-10-14
 */
```

```
public class Car
{
```

```
    // instance variables
    private double gas;
    private double odometer;
    private double milesPerGallon;
```

Instance Variables

These variables are always declared as private, meaning the user doesn't have direct access to them. Users will only be able to interact with this information via the methods we write for them.

```
/**
 * Constructor for objects of class Car
 */
public Car()
{
    // initialise instance variables
    gas = 0;
    odometer = 0;
    milesPerGallon = 20;
}
```

Constructor Methods

One of these two methods will be used to construct a new Car when a program requests it. There are two constructors because there are two different ways to make a *new* Car: one default method, and one in which the initial characteristics of the car are specified.

```
/**
 * Overloaded Constructor for objects of class Car
 * @param initialGas amount of gas in the car
 * @param initialOdometer the initial odometer reading
 * @param milesPerGallon the capacity of the car's gas tank
 */
public Car(double initialGas, double initialOdometer, double milesPerGallon)
{
    // initialise instance variables
    gas = initialGas;
    odometer = initialOdometer;
    this.milesPerGallon = milesPerGallon;
}
```

```
/**
 * getGas method tells how much gas is left in the car
 * @return the amount of gas in the car's tank
 */
public double getGas()
{
    return gas;
}
```

Accessor Methods

These methods are *public* so that a program can access them. They "return" values to a program that uses them, giving the user access to current information about the Car object.

```
/**
 * addGas method adds an amount of gas to the gas tank
 * @param gasAdded the amount of gas being added to the tank
 */
public void addGas(double gasAdded)
{
    this.gas = this.gas + gasAdded;    // or just gas = gas + gasAdded
}
```

Mutator Methods

These methods are also public, and are used to alter the values of the private instance variables in our Car class.

```
/**
 * drive method drives the car a specified distance
 * @param distance the distance the car is driven
 */
public void drive(double miles)
{
    double gasNeeded = miles / milesPerGallon;
    odometer = odometer + miles;
    gas = gas - gasNeeded;
}
```

```
/**
 * getMiles method tells how many miles the car has traveled
 * @return the total miles the car has traveled ever (odometer reading)
 */
public double getMiles()
{
    return this.odometer;
}
```

```
}
```

```

/**
 * CarTester creates several objects of the class Car and tests them.
 *
 * @author Richard White
 * @version 2018-10-14
 */
public class CarTester
{
    public static void main(String[] args)
    {
        // Create two objects of the class Car
        Car myTruck = new Car();

        // 17 gallons in tank, 10,000 miles on odometer, 10 mpg gas mileage
        Car myHighlander = new Car(17, 10000, 10);

        myTruck.addGas(10);

        System.out.println(myTruck.getGas());
        System.out.println("Expected: 10");

        System.out.println(myHighlander.getMiles());
        System.out.println("Expected: 10000");

        myTruck.drive(150);
        System.out.println(myTruck.getMiles());
        System.out.println("Expected: 150");
        System.out.println(myTruck.getGas());
        System.out.println("Expected: 2.5");
    }
}

```

Tester Class

The CarTester class is the main program that will be used to run the Car class that we've created. It's a separate program that uses the Car class to establish two Car objects, and then manipulates those objects using their methods.

Often, we'll want to "test" the code that we've written, and demonstrate that it's working correctly by having it display calculated values, and compare them with expected values.

"Plumbing"

This line is used at the beginning of your main programs. The details of this syntax will be explained throughout the course of the year. For now, just know that you need to include this line at the beginning of your main program or tester.

Default Constructor

This part creates a new object of the class Car, which we'll refer to as "myTruck."

Another Constructor

This one creates a different object, and shows how we can use *parameters* to specify certain initial values.

Mutator Method call

Here we're calling the *addGas()* method for the *myTruck* object. Because it started out with 0 gallons of gas, we'd expect that it has 10 gallons now, but we're going to use the *getGas()* method to confirm that, and output the results along with what we expected to find.

Accessor Method call

We haven't changed any values in the *myHighlander* object yet, but let's confirm that the constructor method did what it was supposed to do. We'll call the *getMiles()* method.