Syncroness Work Sample
Author: Kenneth C. LaMarca
Date: 05/06/2017

--README--

I created a CLI for the program that I was only able to test in windows CMD so for the best experience please use a Windows CMD prompt to execute the code.
If you wish you can just execute the code without a command prompt and it will print out a list of example vehicles in the starting repoistory and which ones
need an oil change.

I used gradle to build the project on a Windows 10 OS machine. I will include an executable jar which can be run with the following command assuming you have
java installed on your computer properly. This jar is just from the gradle jar command which builds a I utilized to build a jar file after every compilation.

```
java -jar SyncronessVehicleInventory-1.0-SNAPSOT.jar
```

PS. I've noticed that the Windows CMD does not like Unicode characters so the © shows up as a question mark. When executing in an IDE the unicode
character \u00A9 (©) shows up properly.

--NOTES--
These are some notes I took prior to tackling the challenge.

Problem Statement:

Syncroness automotive associates needs a vehicle inventory management system. Vehicles are to be identified by make, model, year, color and mileage. Make information should include the manufacturer name and the country of origin of that manufacturer. The repository should allow adding vehicles, fetching, iterating, as well as searches by year and manufacturer. The repository service should keep track of mileage of a vehicle and allow the user to add mileage to a vehicle. The service functionality will also require the ability to retreive vehicles by VIN, model or year. This system should determine whether vehicles are due for an oil change and allow for custom oil change requirements. On execution the application should print to console the inventory of vehicles and which of them require oil change.

Design:

Classes:

Manufacturer:
- Name
- CountryOfOrigin
- List<Model>
- Slogan (ex. Tesla: "Batteries Included!")
- - function toString() ?Name
- - function oilChangeRequirements(model)
- - function addModel(Model, oilChange)

Oil: (keep track of oil change requirements)
- TypeOfOil (not required but this is part of my thought process)
- Months
- Miles

Model:
- Oil
- Year
- Name
- - function toString() ?Year + Name

Vehicle:
- Make? reference

```
        Model? reference
        VIN (unique id)
        Color
        Mileage
        Last Oil change (miles and date)
        -   function isDue() boolean ?call it's manufacturer to ask for oil change requirements
        -   function toString() string ?VIN + Color + Make + Model + Make.slogan
        -   function oilChange() void
        -   function addMileage(int milesToAdd)

    Repository:
        List<Vehicles> with iterator
        List<Manufacturer> (manufacturer has list of models)
        -   function addManufacturer()
        -   function getByYear()
        -   function getByManufacturer()
        -   function getByModel()
        -   function sortedByVIN()
        -   function sortedByModel()
        -   function sortedByYear()

    Service:
        Repository
        //Includes all CLI functionality
        function addMileage(vehicle)
        function findIsDue()
        function doOilChange(vehicle)
```

Discussion:

I want to be able to keep track of manufacturers and models seperately from vehicles. Each vehcile should reference it's manufacturer and model but not contain them. Each vehicle will keep track of when it needs an oil change and just report that to the service class when called upon. The repository acts as a fancy container which has several variations of sorts and iterators. The service class simply calls upon vehicles to report if an oil change is necessary.