Car Manager

A company needs an application to manage their list of cars. Each **Car** has as attributes: *manufacturer's name*, *model, year of fabrication* and *colour*. The car is uniquely identified by the model and the year of fabrication. See below some examples of such cars (are separated by "|"):

Fiat | Bravo | 2007 | red
Fiat | Idea | 2003 | black
Audi | A5 | 2007 | blue
BMV | Coupe | 2013 | pink
Ford | Fiesta | 1976 | yellow

Write a car manager application with a console based user interface which allows to:

1. Add a new car. A message will be shown if the car was successfully added (**1p**). If another car with the same model and the same year of fabrication exists, the application **should not** add the car and show a message (**1p**).
2. Remove a car (**0.5p**).
3. Show all cars, with all their information (**1p**), sorted by manufacturer and model **(1.5p)**.
4. Show all vintage cars: those having a fabrication year older than 45 years (**1p**), sorted by colour **(1p)**.

Input at least 5 cars to your initial list of cars (from the source code).

Write specifications and white-box tests for the following functions:

- Function which adds a car to the car list (the repository/service function). (**0.5** – specification, **0.5** – test)
- Function which retrieves all vintage cars, sorted by manufacturer. (**0.5** – specification, **0.5** – test)

The application must use layered architecture in order for functionalities to be graded.

1p – of.

Time: 60 minutes.