Classes and File operations

Programming in MATLAB

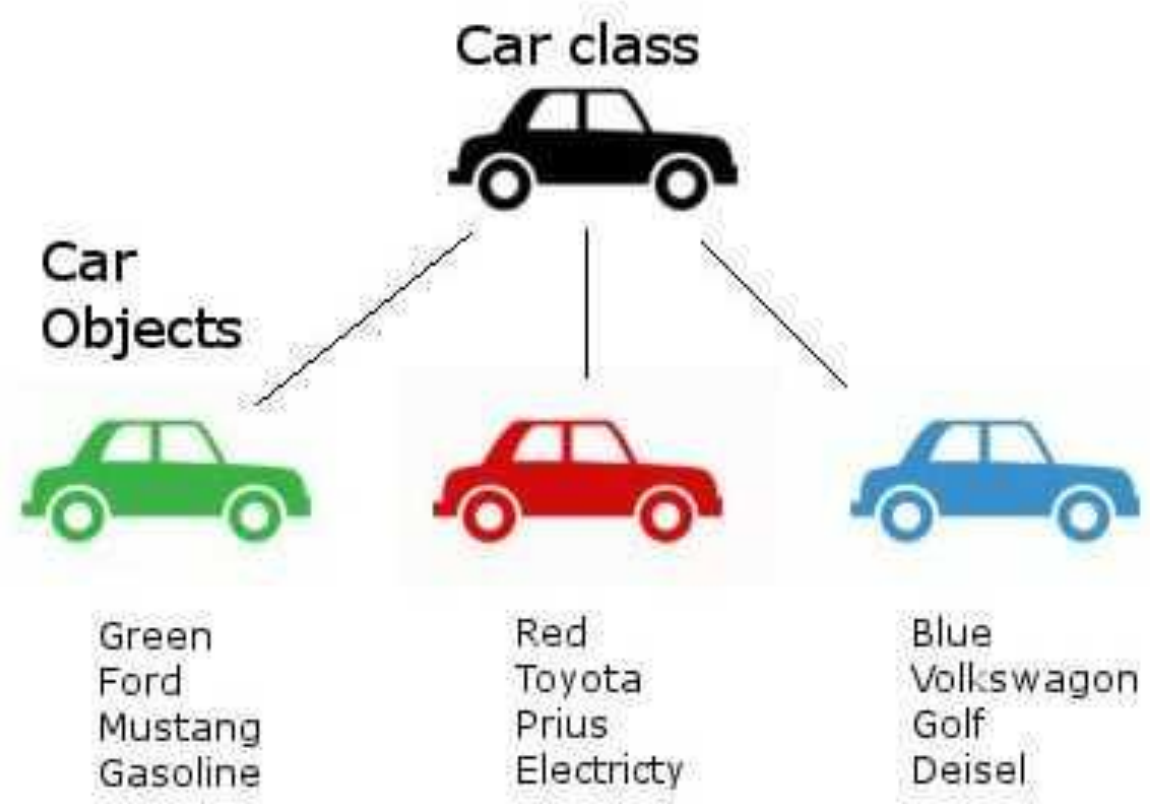
Classes

- Creating new type of **objects** using concepts of object-oriented programming (OOPs).
- Software = Data + Operations on Data.
- **Procedural programs** pass **data** to **functions**, which perform the necessary **operations** on the **data**.
- In **OOPs**, the **data** and the **functions** that operate on them are **bound together** so that **no other part of the code** can **access** this data **except** that **function**.

Classes and Objects

- **Class** is the **blueprint** of an **object**.
  - It is used to declare and create objects.

- A **class** describes a set of **objects** with **common characteristics**. **Objects** are **specific instances** of a **class**.



Creating a Simple Class

```
classdef BasicClass
  properties
    Value {mustBeNumeric}
  end
  methods
    function r = roundOff(obj)
      r = round([obj.Value],2);
    end
    function r = multiplyBy(obj,n)
      r = [obj.Value] * n;
    end
  end
end
```

Value — Property that contains the numeric data stored in an object of the class

Try >> help mustBeNumeric
Try >> lookfor('mustbe')

roundOff — Method that rounds the value of the property to two decimal places

multiplyBy — Method that multiplies the value of the property by the specified number

classdef is a keyword used to define MATLAB classes.

Using a class

- To use the class:
- Save the class definition in a .m file with the same name as the class.

- Create an object of the class.

  a = BasicClass

- Access the properties to assign data.

  a.Value = pi/3;

- Call methods to perform operation on the data. (dot operator)

  a.roundOff(a)
  a.multiplyBy(a,3)

Constructor

- Constructor is a special method to create objects of a class.
- It has the same as that of a class.
- Pass property values as arguments to the constructor

```
methods
    function obj = BasicClass(val)
        if nargin == 1
            obj.Value = val;
        end
    end
End
```

nargin - Number of input arguments to the function

Example : a = BasicClass(pi/3)

BasicClass Code

BasicClass definition after adding the constructor.

```
classdef BasicClass
    properties
        Value {mustBeNumeric}
    end
    methods
        function obj = BasicClass(val)
            if nargin == 1
                obj.Value = val;
            end
        end
        function r = roundOff(obj)
            r = round([obj.Value],2);
        end
        function r = multiplyBy(obj,n)
            r = [obj.Value] * n;
        end
    end
end
```

File operations

Find, view, and change files and folders

File operations

- To list folder contents : type **>> dir** or **>> ls**
- >> list = **ls**(‘*my*’) - List all the files and folders with names that contain my.
- >> list = **ls**(‘*.m’) List all the files and folders with a .m extension.
- >> **pwd** - Identify current folder
- >> **isfile**(fileName) - returns 1 if fileName is a file located on the specified path or in the current folder. Otherwise, isfile returns 0.

Create, Change, and Delete Files and Folders

- **cd** - Change current folder
- **copyfile** - Copy file or folder (returns status logical 1/0)
- **delete** - Delete files or objects
- **mkdir** - Make new folder (returns status logical 1/0)
-
- **movefile** - Move or rename file or folder (returns status logical 1/0)
- **rmdir** - Remove folder (returns status logical 1/0)

File Compression

- Zip - Compress files into zip file

- zippedfiles = **zip**('tmwlogo.zip',{'membrane.m','logo.m’})
  - Compress the files **membrane.m** and **logo.m** into a file named **tmwlogo.zip**.

- **zip**('backup',{'*.m','*.mlx'});
  - Compress all **.m** and **.mlx** files in the current folder to the file **backup.zip**.

Compress a Folder

- Create a folder myfolder containing a subfolder mysubfolder and the files membrane.m and logo.m.

- **mkdir** myfolder;
- **movefile**('membrane.m','myfolder');
- **movefile**('logo.m','myfolder');
- **cd** myfolder;
- **mkdir** mysubfolder;
- **cd** ..

- zippedfiles = **zip**('myfiles.zip','myfolder’);
  - Compress the contents of myfolder, including all subfolders.

Compress Files Into Specified Folder

- zip(zipfilename, filenames, rootfolder)
- zip(**‘../backup.zip'**,**{‘notes.doc’,’tutorial.ppt'}**,**’d:/Subjects’**);
- The files **notes.doc** and **tutorial.ppt** located in the folder **d:/Subjects**.
- Compress these files into **backup.zip**, **one level up from the current folder**.

Extract contents of zip file

- filenames **= unzip**(zipfilename)
  - extracts the archived contents of zipfilename into the current folder.
  - unzip can extract files from your **local system** or from an **Internet URL**

- filenames = **unzip**(zipfilename, outputfolder)
- extracts zipfilename into outputfolder. If outputfolder does not exist, MATLAB creates it.

- Download and extract a zip file from a URL to a local folder.
- url = **'http://example.com/example_file.zip'**;
- **unzip**(url, **'example_folder'**);