Inheritance and Polymorphism

Create a base class called `Car`. It should have attributes that are appropriate for a generic car class (things that might be present for all cars like engine, doors, wheels etc.). Add more attributes according to your imagination.

Constructor should initialize number of `doors` and `name` of the car. Set `wheels` to 4. Passed parameters to the constructor should only be name, doors and engine.

Create appropriate `getters` and `setters`.

Create some generic methods, like `startEngine`(it can print something like “Engine Started”), `accelerate` and `brake`. Add one or two more methods according to your imagination.

Now create 3 subclasses (child classes) for three different cars. Override the appropriate methods to show polymorphism. The parent class has some methods that can be overridden by the child classes. Only override some methods, not all. There should be some methods of the parent class that are overridden and some methods that are not overridden. The child classes should have their own unique attributes and methods in addition to whatever they inherit from the parent class.

(a) Explain how you showed inheritance.
(b) Separately explain how you showed polymorphism.

For (a) and (b) two just add comments at the end of your code.
A comment block can start with `/*`
A comment block ends when you type `*/`

Reading Material

https://www.freecodecamp.org/forum/t/java-inheritance-explained-in-detail-with-examples/16723

Read through the entire thing, and try to understand it completely. It is written in a very easy to understand way.

(More on next page)

Scope

Predict the output of following Java program. Do not type it in your IDE. Try and see how much you understand. After thinking about the output and predicting, you can test it in your IDE.

```
public class Test
{
    static int x = 11;
    private int y = 33;
    public void method1(int x)
    {
        Test t = new Test();
        this.x = 22;
        y = 44;

        System.out.println("Test.x: " +
Test.x);
        System.out.println("t.x: " + t.x);
        System.out.println("t.y: " + t.y);
        System.out.println("y: " + y);
    }

    public static void main(String args[])
    {
        Test t = new Test();
        t.method1(5);
    }
}
```

Access Modifiers

Previously, we discussed access modifiers for methods and attributes. But access modifiers can be used with classes as well. For example, we can have a static class just like we have a static method. find out what it means when we define a **class** as static. Explain with examples.