Software Documentation

Course: [COP-2805C-86399 Java Advanced](https://hcc.instructure.com/courses/135902)

Assignment: Demonstrate OOP principles for encapsulation

Date: 2/4/24

Author Details: Jose A. Cañellas

# Requirements

* The program must include two classes, Car and Person, and a Main source file to run the code called CarAndPerson.
* The Car Class must have the following:
  + Setter and Getters for the Make.
  + Setter and Getters for the Model.
  + Setter and Getters for the Color.
  + Setter and Getters for the Year.
  + Setter and Getters for the Headroom.
  + Setter and Getters for the Transmission.
  + One toScript() method that prints the information of the object.
* The Person Class must have the following:
  + Setter and Getters for the driver’s Name.
  + Setter and Getters for the driver's Age.
  + Setter and Getters for the driver's Height.
  + Setter and Getters for the driver's Stick Shift driving ability.
  + One toScript() method that prints the information of the object.
* CarAndPerson must start by printing the information of the developer, the course, assignment, and the date.
* CarAndPerson must have three Car objects created with all their variables already set.
* CarAndPerson must ask the user to fill in the information for three Person Objects created in CarAndPerson.
* After the user is done filling out the information of each driver, CarAndPerson must print out a report with all the information for the driver and their assigned car.
  + In the case that a driver is too tall for their assigned car or they were assigned to a Stick Shift car without the ability to drive such a car, the proper warning must be displayed.

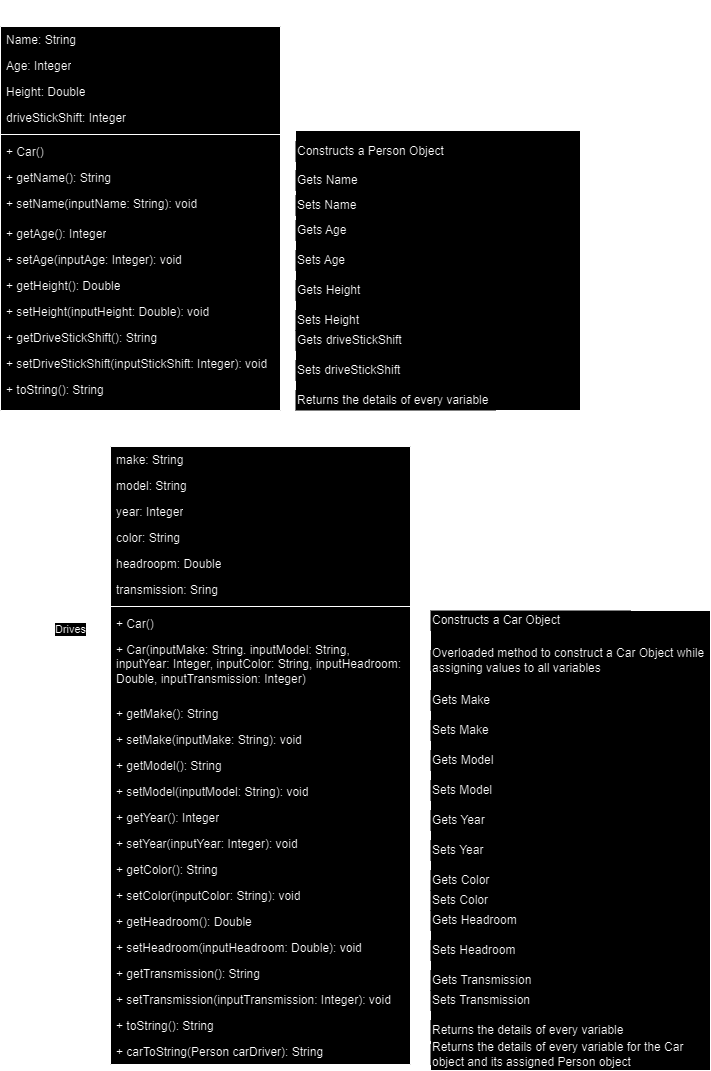
# Installation and Run Instructions

* The user must have any Operating System compatible with the latest version of Visual Studio Code (Windows 10 and 11, Linux, MacOS 10.15 or above)
* The user must install Visual Studio Code and install the “Coding Pack for Java” in the following link (https://code.visualstudio.com/docs/languages/java) . The “Extension Pack” is also recommended.
* Three .java source files will be provided, and the user must open a Java project where they can drag these files into the “src” tab in Explorer.
* On the upper left, click run and then “Start Debugging” or “Run without Debugging” to run CarAndPerson.java.

# Design Notes and UML Diagram

* The Car Class needed Setter and Getter methods for each of its variables (Make, Model, Year, Color, Headroom, and Transmission) so they were created first. During development, it was decided that to set the Transmission, the setTransmission method will use an If statement to verify that the inputTransmission is either 1(for Stick Shift) or 2 (for Automatic) and depending on the input the Transmission will be set. Accordingly, two private/final/static variables called carTransmissionStick(set to 1) and carTransmissionAutomatic(set to 2) were created to compare the input in the If Statements.
* Since the Project requirements asked for the variables of the Car objects were set when CarAndPerson was run, two Constructor methods were made. One that creates an empty Car Object, and an Overloaded Constructor method to make the process faster and simple. For testing, both were used in CarAndPerson.
* Two toString() methods were created, one that simply returns the variables set, and one called carToString() which returns the variables set for both the Car object and the Person object assigned to it. The method carToString() was also made to compare the headroom and height of the Car and the assigned Person, plus to make sure a Person that can’t drive Stick Shift wasn’t assigned to one. If carToString() detects an error, it will print out the proper warning alongside the details for both objects.
* The Person Class needed Setter and Getters, just like Car, so they were created to fill out the variables (Name, Age, Height, and driveStickShift). Similar to Car, setTransmission() uses an if statement to compare the input to two private/static/final variables called stickShiftYes (set to 1) and stickShiftNo (set to 2) and depending on the input the Transmission will be set to “Yes” or “No”.
* The Person Class only has one Constructor method, Person(). It also has one toString() method that returns the values set for each variable. This toString() method is called from within Car to assign a Person object to a Car object.
* For CarAndPerson, the two Constructor methods for Car are tested to assign the variables of three different cars. It begins by printing out the Course, Assignment, Name, and Date before creating the Car objects.
* For the user to enter the values of each Person object, CarAndPerson prompts the user to enter each detail separately. This is repeated for all three Person objects.
* For simplicity, instead of verifying for incompatibility between a Person’s height and Stick Shift driving ability, it was decided to move that process into the Car Class itself, as described in the third paragraph of this section (Design Notes and UML Diagram)

**UML Diagram:**



# Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement | Input/Action | Expected | Output | Pass/Fail |
| Car() must create a Car object | Car() and the details of each variable | Car object is created | A Car object is created with empty field, or all fields filled (If Overloaded Method is used | PASS |
| Car getter and setter methods add or return assigned variables | When called, Setters set the variable and Getters return the variable | Variables are either set or returned | Setter methods properly set the variables and Getter methods properly return the variables | PASS |
| carToString() is assigned a driver and verifies compatibility | A Person object is assigned, then the method verifies compatibility, then throws a warning if incompatible | Returns the details of the assigned driver and their car | Returns the details of both objects, and throws a warning since driver 2 was too tall and driver 3 was assigned a Stick Shift car they can’t drive | PASS |
| Person() must create a Person object | Person() is called, and a Person object is created | Person() is called, and a Person object is created | When called, Person() created a Person object to be filled in CarAndPerson | PASS |
| Person Getter and Setter methods | When called, the setters will set a variable while the getters will return a variable | Variables are set or returned depending on the method | All setter methods set variables while the getters return the variables | PASS |
| CarAndPerson must request the user for the details of all three drivers | When prompted, the user will input each detail to be assigned | After each input, the setter methods take the input and assign them to the proper variable | Input is correctly assigned to their variables through the setter methods | PASS |

# Screenshots

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated