EDITORIAL ARTICLE

An object-oriented programming solution for managing inventory of used cars in Kenya using inheritance relationships.

Waweru Mwangi

[1]*Department of Computing, Jomo Kenyatta University of Agriculture and Technology (JKUAT) Kenya.*

Corresponding author: waweru_mwangi@icsit.jkuat.ac.ke

ABSRACT

The industry of buying and selling used cars in Kenya has continued to grow by leaps and bounds. There are many dealers who have opened shops in Nairobi, Mombasa, and many other towns locally. Therefore, having a software solution that can manage this data appropriately is important for the day-to-day running of these businesses. Inheritance is a key concept in object-oriented programming that uses the concepts of generalization and specialization. Inheritance allows a new class to extend an existing class. The new class inherits the members of the class it extends. This review paper explores the use of inheritance in developing software that used car dealers can find useful in managing their data.

Key words: Inheritance, superclass, objects.

1.0 Introduction

More than eighty percent of the cars bought and sold in Kenya are used (second-hand) cars. This is a thriving business that has provided many people with an income. Organizations such as the Kenya Autobazaar Association, Jiji, A-Plus Motors (A-plus Motors, n.d.), and many others have even built nice websites where they market these types of cars. Having a software tool that can aggregate and specialize vehicle data when necessary is vital, especially as stock numbers grow. Inheritance provides a useful object-oriented mechanism to achieve this end (Gaddis, 2019).

Inheritance uses the concepts of generalization and specialization. Many objects in the real world are specialized versions of other, more general objects. For example, the term "snake "describes a general type of creature with various characteristics. Because the puff adder and cobra are snakes, they have all the general characteristics of snakes. In addition, they have special characteristics of their own. For example, the puff adder has an average size of 1 meter, but the cobra's average length is 1.5 meters. Puff adders and cobras are specialized versions of snakes.

URL: https://ojs.jkuat.ac.ke/index.php/JAGST
ISSN 1561-7645 (online)
doi: 10.4314/jagst.v22i3.1

2.0 Program design and development

In developing a program that a car dealership in Kenya can use to manage its inventory, we focus on three types of automobiles: cars, pickups and lorries, and sport-utility vehicles (SUV). Regardless of the type, the dealership keeps the following data about each automobile:

- Make
- Year Model
- Mileage and
- Price

Each type of vehicle that is kept in inventory has these general characteristics plus its own specialized characteristics. For cars, the dealership keeps the following additional data:

- Number of doors (2 or 4)

For pickups and trucks, the dealership keeps the following additional data:

- Drive type (two-wheel type or four-wheel type)

For SUVs, the dealership keeps the following additional data:

- Passenger capacity

In designing this program, we wrote an automobile superclass to hold all the general data about an automobile. We then write subclasses for each specific type of automobile. The programs have been written using Python 3.9 and tested on the Spyder framework.

```
#The Automobile class holds general data
#about an automobile in inventory.
class Automobile:
    #The __init__ method accepts arguments for the
    #make, model, milieage, and price. It initializes
    #the data attributes with these values.
    def __init__(self, make, model, mileage, price):
        self.__make = make
        self.__model = model
        self.mileage = mileage
        self.__price = price
    #The following methods are mutators for the
    #class's data attributes.
    def set_make(self, make):
        self.__make = make
    def set_model(self, model):
        self.__model = model
    def set_mileage(self, mileage):
        self.__mileage = mileage
    def set_price(self, price):
        self.__price = price

    #The following methods are the accessors
    #for the class's data attributes.
    def get_make(self):
        return self.__make
    def get_model(self):
        return self.__model
```


URL: https://ojs.jkuat.ac.ke/index.php/JAGST
ISSN 1561-7645 (online)
doi: 10.4314/jagst.v22i3.1

```
    def get_mileage(self):
        return self.mileage
    def get_price(self):
        return self.__price

class Car(Automobile):
    #The __init__ method accepts arguments for the
    #car's make, model, price, and doors.
    def __init__(self, make, model, mileage, price, doors):
        #Call the superclass's _init__ method and pass
        #the required arguments. Note that we also have
        #to pass self as an argument.
        Automobile.__init__(self, make, model, mileage, price)
        #Initialize the __doors attribute.
        self.__doors = doors
    #The set_doors method is the mutator for the
    #__doors attribute.
    def set_doors(self, doors):
        self.__doors = doors
    #The get_doors method is the accessor for the
    #__doors attribute.
    def get_doors(self):
        return self.__doors

class PickupAndLorry(Automobile):
    #The __init__ method accepts arguments for the
    #PickupAndLorry's make, model, mileage, price, and drive type.

    def __init__(self, make, model, mileage, price, drive_type):
        #Call the superclass's __init__ method and pass
        #the required arguments. Note that we also have
        #to pass self as an argument.
        Automobile.__init__(self, make, model, mileage, price)

        #Initialize the drive_type attribute.
        self.__drive_type = drive_type

    #The set_drive_type method is the mutator for the
    #__drive_type attribute.

    def set_drive_type(self, drive_type):
        self.__drive_type = drive_type

    #The get_drive_type  method is the accessor for the
    #__drive_type attribute.

    def get_drive_type(self):
        return self.__drive_type

class SUV(Automobile):
    #The __init__ method accepts arguments for the
    #SUV's make, model, mileage, price, and passenger
    #capacity.
    def __init__(self, make, model, mileage, price, pass_cap):
        #Call the superclass's __init__ method and pass
        #the required arguments. Note that we also have
        #to pass self as an argument.
        Automobile.__init__(self, make, model, mileage, price)

        #Initialize the __pass_cap attribute.
        self.__pass_cap = pass_cap

    #The set_pass_cap method is the mutator for the
```


URL: https://ojs.jkuat.ac.ke/index.php/JAGST
ISSN 1561-7645 (online)
doi: 10.4314/jagst.v22i3.1

```
    #__pass_cap attribute.

    def set_pass_cap(self, pass_cap):
        self.__pass_cap = pass_cap

    #The get_pass_cap method is the accessor for the
    # __pass_cap attribute.

    def get_pass_cap(self):
        return self.__pass_cap
```

Figure 1.0 shows the UML diagram for this relationships (Inheritance)

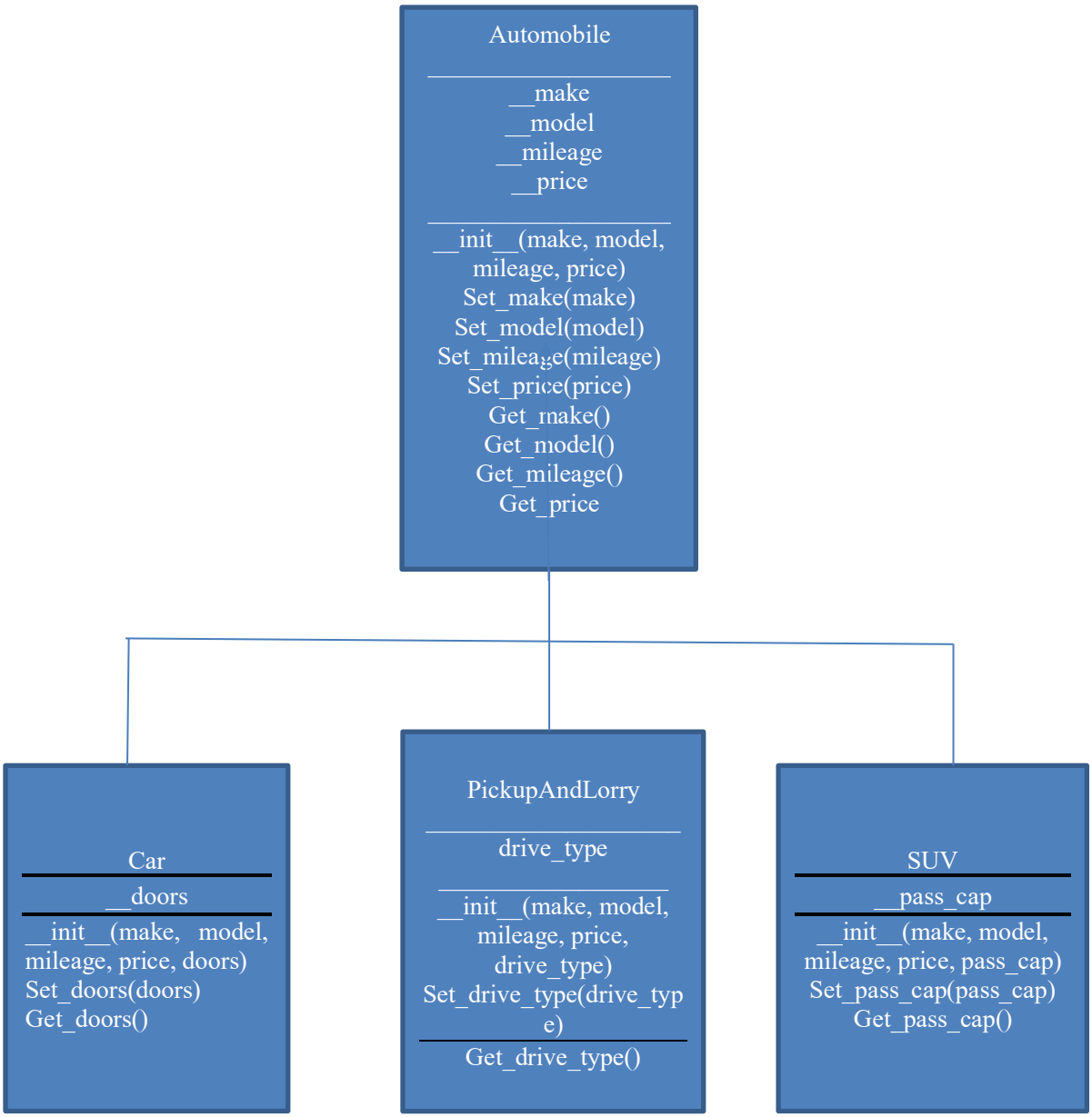

Figure 1.0: UML diagram for inheritance relationships in the Python program

URL: https://ojs.jkuat.ac.ke/index.php/JAGST
ISSN 1561-7645 (online)
doi: 10.4314/jagst.v22i3.1

It is possible to extend this design to includes other details like dealers and/or suppliers, name, location, address and contacts.

3.0 Conclusion

Combining the use of open-source software and object-oriented tools remains a highly cost-effective approach to building software solutions for small and micro enterprises (SME) in developing countries. The developed program can be deployed on any machine and requires minimal resources to run. It can also be scaled up to handle large volumes of data. It can be enhanced to be able to fetch data from a given database.

4.0 Acknowledgement

This study has benefited from support given by Africa-ai-Japan Phase II project.

5.0 References

A-Plus Motors. (n.d.). Aplusmotors.co.ke. Retrieved May 2, 2023, from http://aplusmotors.co.ke

Gaddis, T. (2019). *GLOBAL EDITION FOURTH EDITION Starting Out with Python ®*. http://www.kalfaoglu.com/ceng113/Python-Programming/Starting%20Out%20with%20Python%5B4th%20Globa%20lED%5DTony%20Gaddis.pdf

URL: https://ojs.jkuat.ac.ke/index.php/JAGST
ISSN 1561-7645 (online)
doi: 10.4314/jagst.v22i3.1