LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021-22 Spring - 2

Student Name: Pramit Badgami

Group: C14

London Met ID: NP01CP4S220137

College ID: NP01CP4S220137

Assignment Due Date: 20th May, 2022

Assignment Submission Date: 20th May, 2022

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

Table of Figures

Table of tables

1. Introduction

Programming is one of the main modules which is very important to understand. Programming is the core of any IT related topic. This Coursework teaches us how to interact with the computer and give set of instructions to perform a task. Programming can be performed in multiple programming languages such as Java, Python, C, C++, and many more. These languages are the platform where the coding is executed using different style of coding in each one. Programming involves writing codes in a step wise way which gives the output in a chronological order as per need. (Wilkins, 2021)

1.1 Java

Java is a vastly used programming language which allows the programmers to implement the code. It was first released in 1995 by James Gosling at Sun Microsystems. Users can run a program anywhere after the code has been compiled from a programming language to a machine readable language. Java also helps to reduce costs and improves application services. Syntax used in Java is based on C++ and also very swayed by C++. Java is platform independent as it can be used on various operating systems such as Windows, MAC, Linux, etc. Java can execute two or more tasks of a program at once which is also known as multi-threading. Java is a simple and secured programming language which keeps all the programs in a safe manner. (IBM Cloud Education, 2019)

1.2 BlueJ

BlueJ is a Java integrated development environment (IDE) used for java programming language. BlueJ is used to learn the object oriented programming and it helps to execute the java programming language. It was developed by Michael Kölling and John Rosenberg in 1999. BlueJ is a free and easy-to-use environment which provides multiple features for the programmers to implement their codes in Java Programming Language. Java programs are quickly developed in BlueJ. (GeeksforGeeks, 2020)

1.3 MS Word

Microsoft Word is a built-in software program which allows to execute multiple tasks. It is a word processor used for creating reports, letters, documents, etc. It was first released in 1981 by the developing organization known as Microsoft. MS Word is the software program that we used to making the report for this particular module Programming. It has its own advantages as it allows for wider range of spectrum to make different documents as per the requirement.

1.4 Draw.io

Draw.io is a web based diagram applications that allows the users to create different diagrams, flowcharts, pie-charts, etc. In this coursework, we used draw.io for creating a class diagram for representing multiple variables, methods and data types of all classes.

2. Class Diagram

Class Diagram is a visual representation that describes the structure of classes, their attributes, and methods. Class Diagram is represented in tabular form where there is 1 column and 3 rows. The first row includes the name of the class, the second row includes all the attributes and the third row includes all the methods.

2.1 Class Diagram of Vehicle Class

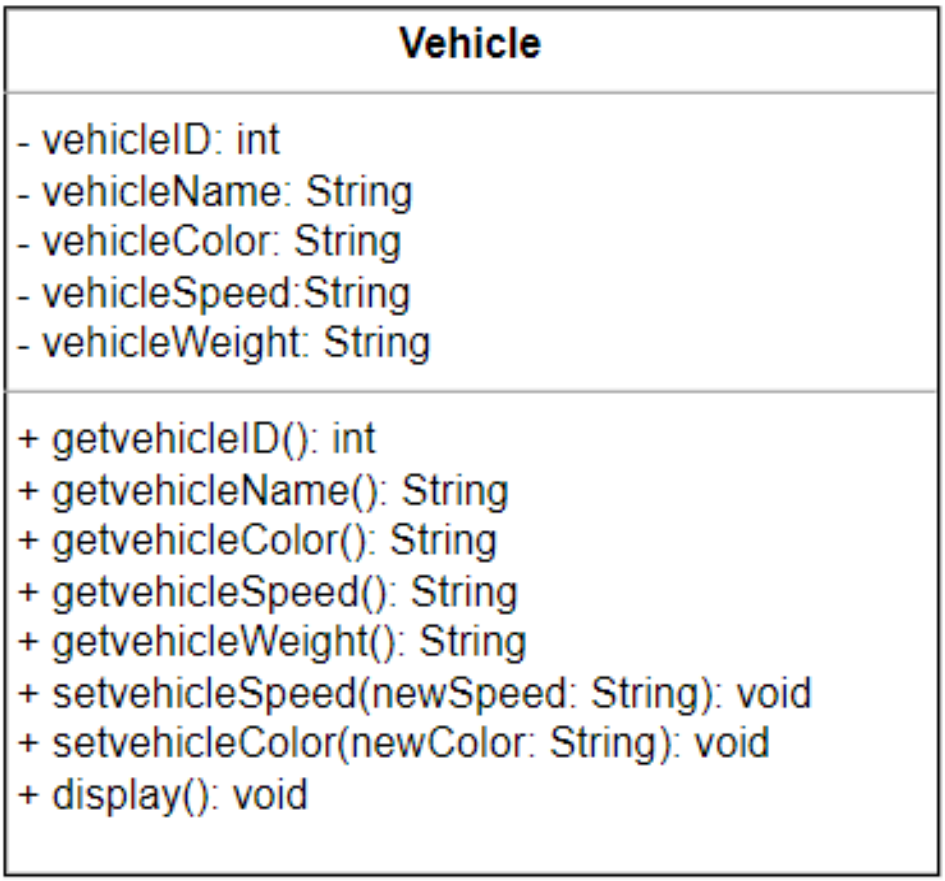

Figure 1: Class Diagram of Vehicle Class

2.2 Class Diagram of AutoRickshaw

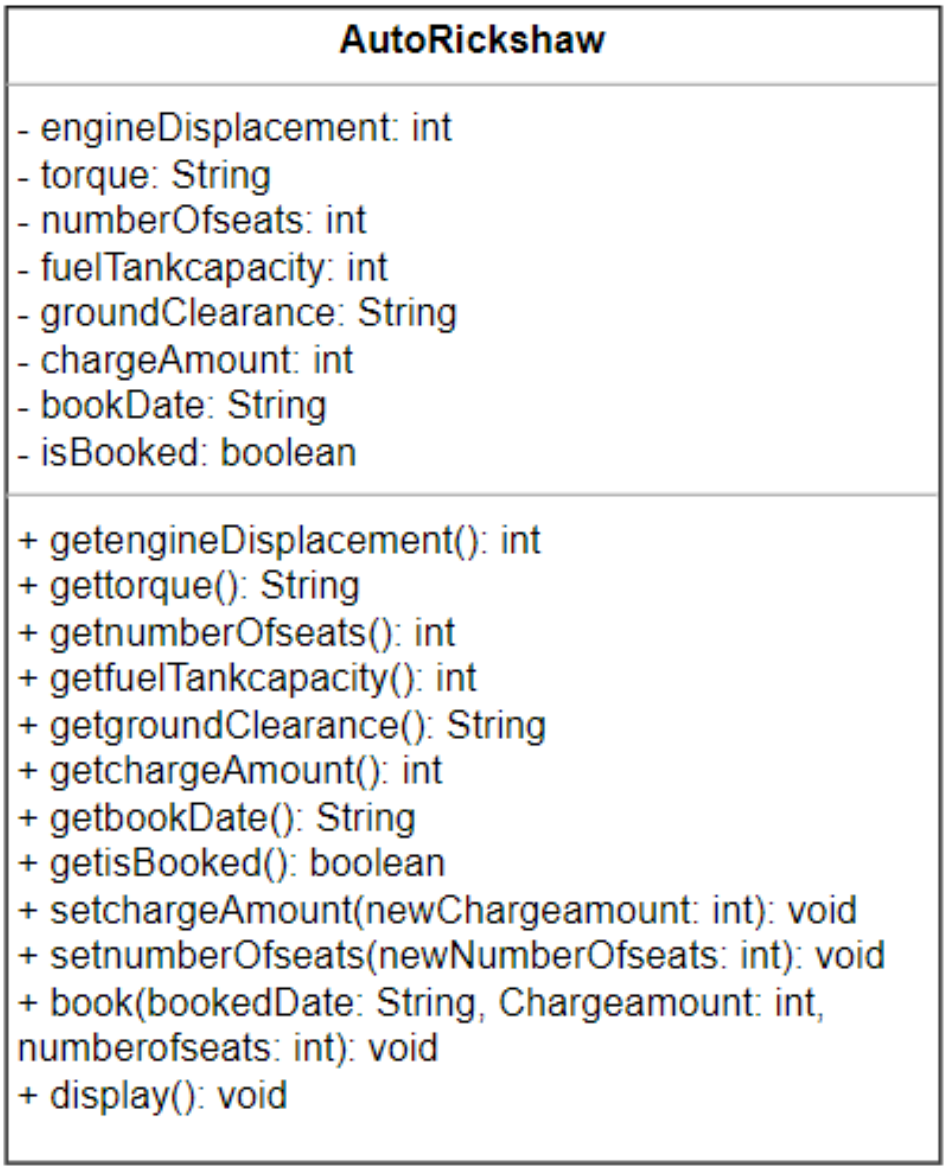

Figure 2: Class Diagram of AutoRickshaw Class

2.3 Class Diagram of ElectricScooter

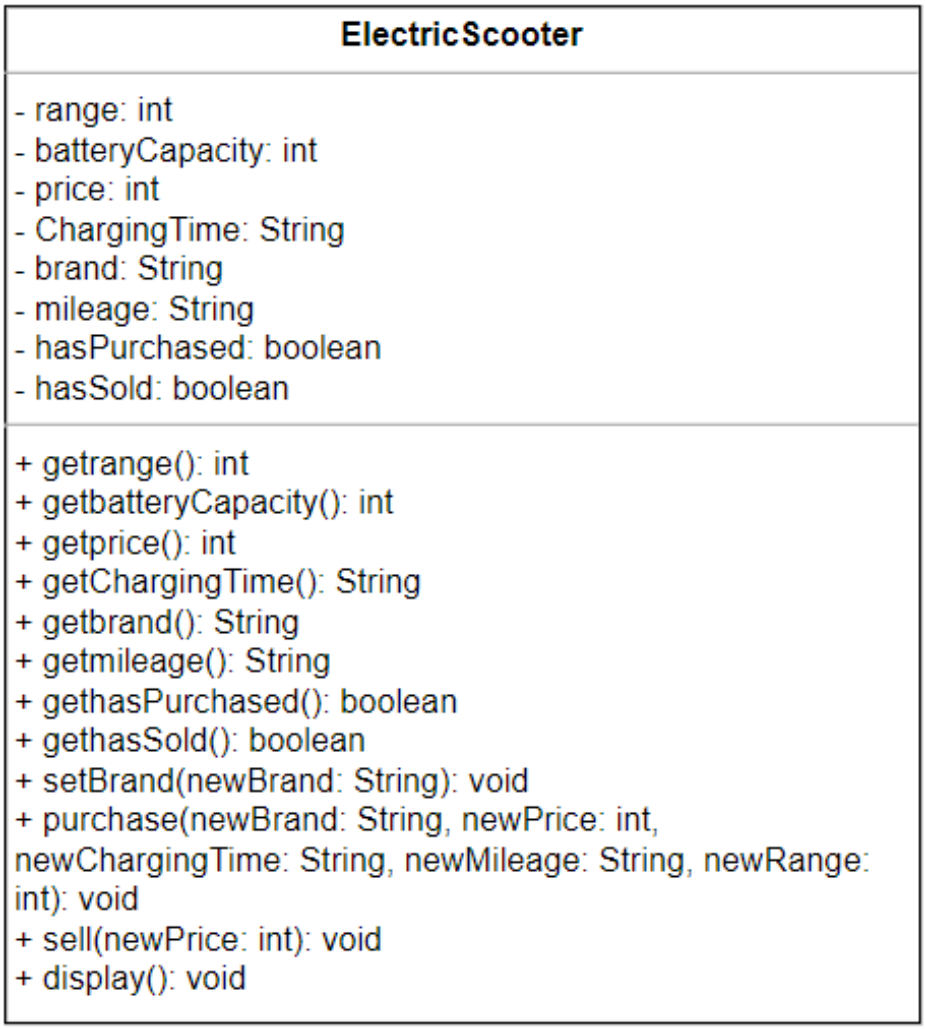

Figure 3: Class Diagram of ElectricScooter Class

2.4 Inheritance

Inheritance is the process where a class inherits the properties of another class. A class has its own methods and fields which can be acquires through new classes. The class which acquires the properties of others is known as subclass and the class whose properties are inherited is called superclass. Inheritance is used here in this class diagram where the superclass is Vehicle and subclasses are AutoRickshaw and ElectricScooter. (Schildt, 2017)

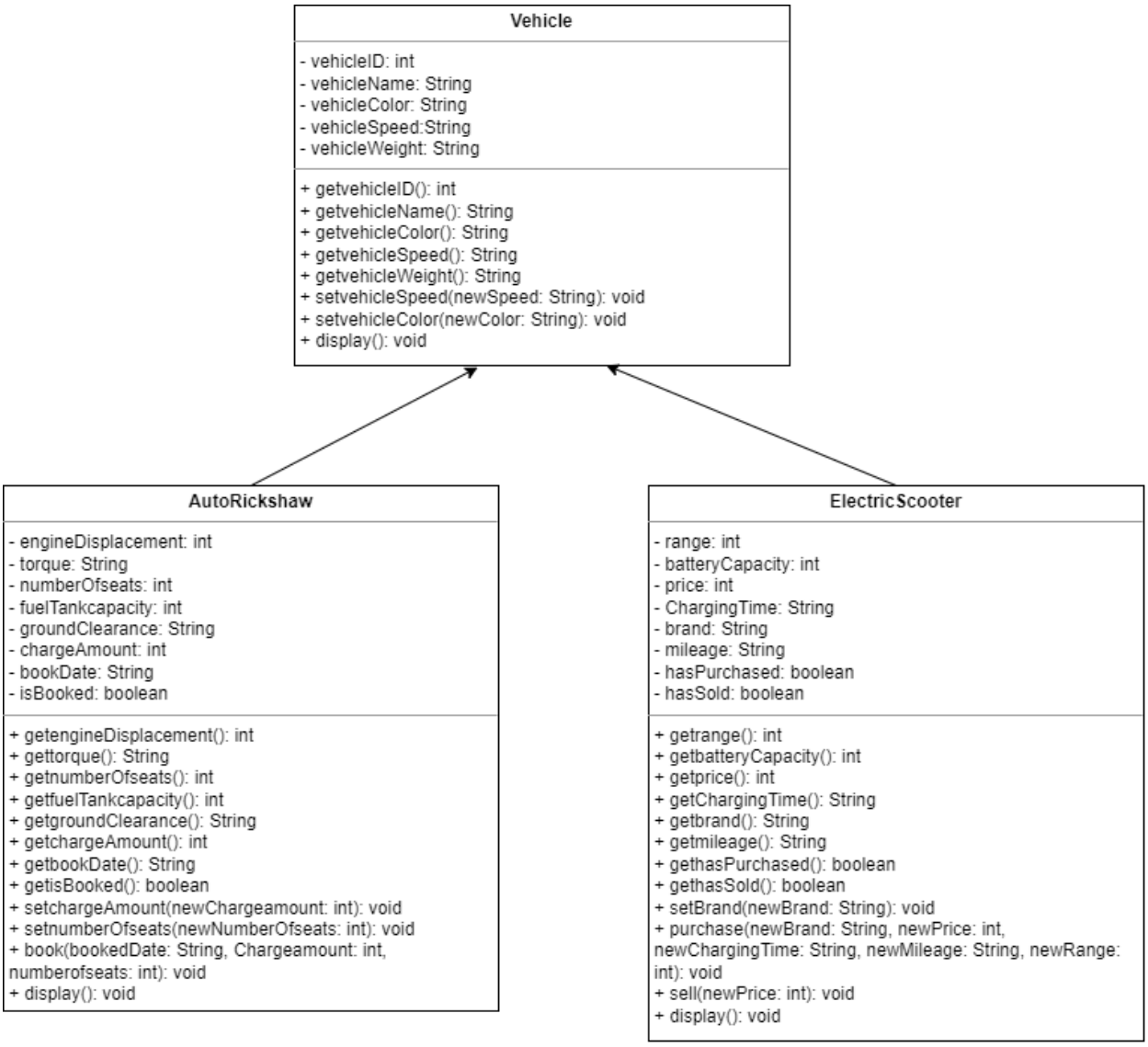

Figure 4: Class Diagram

3. Pseudocode

Pseudocode is an informal high-level representation of the code itself. Pseudocode turns any code into plain English and helps to understand the code. It helps to form algorithms using an artificial language which is easily readable and understandable. It also doesn't require any syntax to write a Pseudocode. Pseudocode gives a brief summary of the code's proper flow and insures that everyone understands it. Pseudocode ensures fixing of errors and bug which may be harder for programmers to detect in the program itself. (Metwalli, 2021)

3.1 Pseudocode of Vehicle Class

CREATE a parent class Vehicle

DO

DECLARE an instance variable vehicleID as int datatype

DECLARE an instance variable vehicleName as String datatype

DECLARE an instance variable vehicleColor as String datatype

DECLARE an instance variable vehicleSpeed as String datatype

DECLARE an instance variable vehicleWeight as String datatype

CREATE a constructor Vehicle which accepts four parameters: vehicleID as int, vehicleName as String, vehicleColor as String, and vehicleWeight as String

DO

INITIALIZE the value of Instance variable vehicleID

INITIALIZE the value of Instance variable vehicleName

INITIALIZE the value of Instance variable vehicleColor

INITIALIZE the value of Instance variable vehicleWeight

END DO

CREATE an accessor method getvehicleID() with return type int

DO

RETURN vehicleID

END DO

CREATE an accessor method getvehicleName() with return type String

DO

RETURN vehicleName

END DO

CREATE an accessor method getvehicleColor() with return type String

DO

RETURN vehicleColor

END DO

CREATE an accessor method getvehicleSpeed() with return type String

DO

RETURN vehicleSpeed

END DO

CREATE an accessor method getvehicleWeight() with return type String

DO

RETURN vehicleWeight

END DO

CREATE a mutator method setvehicleSpeed() as void with parameter newSpeed as String

DO

INITIALIZE the value of vehicleSpeed as newSpeed

END DO

CREATE a mutator method setvehicleColor() as void with parameter newColor as String

DO

INITIALIZE the value of vehicleColor as newColor

END DO

CREATE a display method as void

DO

PRINT vehicleID

PRINT vehicleName

PRINT vehicleColor

PRINT vehicleSpeed

IF vehicleWeight is empty

DO

PRINT vehicle weight is empty

END DO

```
        IF END
        ELSE
        DO
            PRINT Vehicle weight
            END DO
        ELSE END
        END DO
    END DO
```

3.2 Pseudocode of AutoRickshaw Class

CREATE AutoRickshaw which is a child class of Vehicle

DO

DECLARE an instance variable engineDisplacement as int datatype

DECLARE an instance variable torque as String datatype

DECLARE an instance variable numberOfseats as int datatype

DECLARE an instance variable fuelTankcapacity as int datatype

DECLARE an instance variable groundClearance as String datatype

DECLARE an instance variable chargeAmount as int datatype

DECLARE an instance variable bookDate as String datatype

DECLARE an instance variable isBooked as boolean datatype

CREATE a constructor AutoRickshaw which accepts nine parameters: vehicleID as int, vehicleName as String, vehicleColor as String, vehicleWeight as String, vehicleSpeed as String, engineDisplacement as int, torque as String, fuelTankcapacity as int and groundClearance as String

DO

CALLING constructor of superclass which accepts four parameters: vehicleID, vehicleName, vehicleColor, vehicleWeight

INITIALIZE the value of instance variable engineDisplacement

INITIALIZE the value of instance variable torque

INITIALIZE the value of instance variable fuelTankcapacity

INITIALIZE the value of instance variable groundClearance

CALLING a mutator method setvehicleSpeed() from superclass Vehicle with vehicleSpeed as the parameter

CALLING a mutator method setvehicleColor() from superclass Vehicle with vehicleColor as the parameter

ASSIGN isBooked as false

END DO

CREATE an accessor method getengineDisplacement() with return type int

DO

RETURN engineDisplacement

END DO

CREATE an accessor method gettorque() with return type String

DO

RETURN torque

END DO

CREATE an accessor method getnumberOfseats() with return type int

DO

RETURN numberOfseats

END DO

CREATE an accessor method getfuelTankcapacity() with return type int

DO

RETURN fuelTankcapacity

END DO

CREATE an accessor method getgroundClearance() with return type String

DO

RETURN groundClearance

END DO

CREATE an accessor method getchargeAmount() with return type int

DO

RETURN chargeAmount

END DO

CREATE an accessor method getbookDate() with return type String

DO

RETURN bookDate

END DO

CREATE an accessor method getisBooked() with return type Boolean

DO

RETURN isBooked

END DO

CREATE a mutator method setchargeAmount() as void with parameter newChargeamount as int

DO

INITIALIZE the value of chargeAmount as newChargeamount

END DO

CREATE a mutator method setnumberOfseats() as void with parameter newNumberOfseats as int

DO

INITIALIZE the value of numberOfseats as newNumberOfseats

END DO

CREATE a method book with return type void which accepts parameters: bookedDate as String, Chargeamount as int, and numberofseats as int

DO

IF isbooked is false

DO

ASSIGN bookDate as bookedDate

PRINT Your autorickshaw is now booked

CREATE a method setchargeAmount() with return type void which has a parameter as chargeamount

CREATE a method setnumberOfseats() with return type void which has a parameter as numberofseats

ASSIGN isBooked is true

PRINT your autorickshaw is already booked

END DO

IF END

ELSE

DO

```
            PRINT the status of isBooked is
            PRINT your autorickshaw is already booked
            END DO
        END ELSE
        END DO
    CREATE a display method as void
    DO
        CALLING display method from super class
        IF isBooked is true
        DO
            PRINT Engine Displacement
            PRINT torque
            PRINT Fuel tank capacity
            PRINT Ground clearance
            PRINT Booked Date
            END DO
        IF END
        IF chargeAmount is 0
        DO
            PRINT charge amount is not recorded
            END DO
```

```
        IF END
        ELSE
        DO
            PRINT Charged amount:
            END DO
        ELSE END
        IF numberOfseats is 0
        DO
            PRINT Number of seats are not recorded
            END DO
        IF END
        ELSE
        DO
            PRINT number of seats:
            END DO
        ELSE END
        END DO
    END DO
```

3.3 Pseudocode of ElectricScooter

CREATE ElectricScooter which is a child class of Vehicle

DO

DECLARE an instance variable range as int datatype

DECLARE an instance variable batteryCapacity as int datatype

DECLARE an instance variable price as int datatype

DECLARE an instance variable ChargingTime as String datatype

DECLARE an instance variable brand as String datatype

DECLARE an instance variable mileage as String datatype

DECLARE an instance variable hasPurchased as boolean datatype

DECLARE an instance variable hasSold as boolean datatype

CREATE a constructor ElectricScooter which accepts six parameters: vehicleID as int, vehicleName as String, vehicleWeight as String, vehicleSpeed as String, vehicleColor as String and batteryCapacity as int

DO

CALLING constructor of superclass Vehicle which accepts four parameters: vehicleID, vehicleName, vehicleColor, vehicleWeight

ASSIGN range as 0

ASSIGN price as 0

ASSIGN brand as empty

ASSIGN mileage as empty

ASSIGN ChargingTime as empty

INITIALIZE batteryCapacity as batteryCapacity

ASSIGN hasPurchased as false

ASSIGN hasSold as false

CALLING a mutator method setvehicleSpeed() of superclass Vehicle with vehicleSpeed as the parameter

CALLING a mutator method setvehicleColor() of superclass Vehicle with vehicleColor as the parameter

END DO

CREATE an accessor method getrange() with return type int

DO

RETURN range

END DO

CREATE an accessor method getbatteryCapacity() with return type int

DO

RETURN batteryCapacity

END DO

CREATE an accessor method getprice() with return type int

DO

RETURN price

END DO

CREATE an accessor method getChargingTime() with return type String

DO

RETURN ChargingTime

END DO

CREATE an accessor method getbrand() with return type String

DO

RETURN brand

END DO

CREATE an accessor method getmileage() with return type String

DO

RETURN mileage

END DO

CREATE an accessor method gethasPurchased() with return type Boolean

DO

RETURN hasPurchased

END DO

CREATE an accessor method gethasSold() with return type Boolean

DO

RETURN hasSold

END DO

CREATE a mutator method setBrand() as void with parameter newBrand as String

DO

IF hasPurchased is false

DO

INITIALIZE the value of brand as newBrand

END DO

IF END

ELSE

DO

PRINT Brand cannot be changed

END DO

ELSE END

END DO

CREATE a method purchase() with return type void which accepts parameters: newBrand as String, newPrice as int, newChargingTime as String, newMileage as String and newRange as int

DO

IF hasPurchased is false

DO

CREATE a method setBrand() with return type void which has a parameter brand

INITIALIZE the value of price as newPrice

INITIALIZE the value of ChargingTime as newChargingTime

INITIALIZE the value of mileage as newMileage

INITIALIZE the value of range as newRange

INITIALIZE the value of hasPurchased as true

END DO

IF END

ELSE

DO

PRINT has already been purchased

END DO

ELSE END

END DO

CREATE a method sell() with return type void which accepts the parameter newPrice as int

DO

IF hasSold is false

DO

INITIALIZE the value of price as newPrice

INITIALIZE the value of range as 0

INITIALIZE the value of ChargingTime as empty

INITIALIZE the value of mileage as empty

INITIALIZE the value of batteryCapacity as 0

```
            INITIALIZE the value of hasSold as true
            INITIALIZE the value of hasPurchased as false
            END DO
        IF END
        ELSE
        DO
            PRINT The scooter is already purchased
            END DO
        ELSE END
        END DO
    CREATE a display method as void
    DO
        CALLING display method from superclass
        IF hasPurchased is true
        DO
            PRINT Brand
            PRINT Battery Capacity
            PRINT Mileage
            PRINT Range
            PRINT Recharge time
            END DO
```

```
        IF END

        END DO

    END DO
```

4. Method

Method is a series of code that only runs when it is called. It is a collection of instructions which performs a specific task in java. Method can help to reuse the code when and where needed in multiple places. Method is only executed when called for it. Method have void as its return type and can return the result to the caller when needed. Method in java can also perform some task without returning anything. A verb in single word or multi-word beginning with a verb is typically used as a method name. Method name always starts a lowercase letter while specifying it. Every method has its own function while calling it. (GeeksforGeeks, 2022)

There are two types of methods:-

- **Predefined Method:** Predefined methods are built-in methods which are already defined in Java standard libraries. These methods can be directly used when called in the program at any time.
- **User-defined Method:** User-defined methods are methods created by the user as per own requirements and needs. These methods can also be modified per the user's requirements while predefined method cannot be modified.

4.1 Method Description of Vehicle Class

Description of every method used in super class Vehicle is as follows:

Method	Return Type	Description
getvehicleID()	int	getvehicleID() is an accessor method which returns the value of instance variable vehicleID.
getvehicleName()	String	getvehicleName() is an accessor method which returns the value of instance variable vehicleName.
getvehicleColor()	String	getvehicleColor() is an accessor method which returns the value of instance variable vehicleColor.
getvehicleSpeed()	String	getvehicleSpeed() is an accessor method which returns the value of instance variable vehicleSpeed.
getvehicleWeight()	String	getvehicleWeight() is an accessor method which returns the value of instance variable vehicleWeight.
setvehicleSpeed()	void	setvehicleSpeed() is a mutator method which sets the value of instance variable vehicleSpeed. It has newSpeed as its parameter whose return type is String.
setvehicleColor()	void	setvehicleColor() is a mutator method which sets the value of instance variable vehicleColor. It has newColor as its

		parameter whose return type is String.
display()	void	display() is a method which shows the output. It prints Vehicle ID, Vehicle Name, Vehicle Color, Vehicle Speed, vehicle Weight and gives an appropriate message.

Table 1: Methods of Vehicle Class

4.2 Method Description of AutoRickshaw Class

Description of every method used in sub class AutoRickshaw is as follows:

Method	Return Type	Description
getengineDisplacement()	int	getengineDisplacement() is an accessor method which returns the value of instance variable engineDisplacement.
gettorque()	String	gettorque() is an accessor method which returns the value of instance variable torque.
getnumberOfseats()	int	getnumberOfseats() is an accessor method which returns the value of instance variable numberOfseats.
getfuelTankcapacity()	int	getfuelTankcapacity() is an accessor method which returns the value of instance variable fuelTankcapacity.
getgroundClearance()	String	getgroundClearance() is an accessor method which returns the value of instance variable groundClearance.
getchargeAmount()	int	getchargeAmount() is an accessor method which returns the value of instance variable chargeAmount.
getbookDate()	String	getbookDate() is an accessor method which returns the value of instance variable bookDate.

isBooked()	boolean	isBooked() is an accessor method which returns the value of instance variable isBooked.
setchargeAmount()	void	setchargeAmount() is a mutator method which sets the value of instance variable chargeAmount. It has newChargeamount as its parameter whose return type is int.
setnumberOfseats()	void	setnumberOfseats() is a mutator method which sets the value of instance variable numberOfseats. It has newNumberOfseats as its parameter whose return type is int.
book()	void	book() is a method which has parameters: bookedDate as String data type, Chargeamount and numberofseats as int data type. This method is used to book the Auto Rickshaw.
display()	void	display() is a method which shows the output. It calls the display() method of the parent class Vehicle. This method is mainly used to display the required details of the Auto Rickshaw.

Table 2: Methods of AutoRickshaw Class

4.3 Method Description of ElectricScooter Class

Description of every method used in sub class ElectricScooter is as follows:

Method	Return Type	Description
getrange()	int	getrange() is an accessor method which returns the value of instance variable range.
getbatteryCapacity()	int	getbatteryCapacity() is an accessor method which returns the value of instance variable batteryCapacity.
getprice()	int	getprice() is an accessor method which returns the value of instance variable price.
getChargingTime()	String	getChargingTime() is an accessor method which returns the value of instance variable ChargingTime.
getbrand()	String	getbrand() is an accessor method which returns the value of instance variable brand.
getmileage()	String	getmileage() is an accessor method which returns the value of instance variable mileage.
gethasPurchased()	boolean	gethasPurchased() is an accessor method which returns the value of instance variable hasPurchased.

gethasSold()	boolean	gethasSold() is an accessor method which returns the value of instance variable hasSold.
setBrand()	void	setBrand() is a mutator method which sets the value of instance variable Brand. It has newBrand as its parameter whose return type is String.
purchase()	void	purchase() is a method which has parameters: newBrand, newMileage and newChargingTime as String data type and newPrice and newRange as int data type. This method is used to purchase the Electric Scooter.
sell()	void	sell() is a method which has parameter: newPrice as int data type. This method is used to sell the Electric Scooter.
display()	void	display() is a method which shows the output. It calls the display() method of the parent class Vehicle. This method is mainly used to display the required details of the Electric Scooter.

Table 3: Methods of ElectricScooter Class

5. Testing

5.1 Test 1 – To inspect AutoRickshaw class, book the Auto-Rickshaw and re-inspect the AutoRickshaw class

Test No.	1
Objective	To inspect AutoRickshaw class, book the auto-rickshaw and re-inspect the AutoRickshaw class.
Action	➢ The AutoRickshaw is called with the following arguments: vehicleID = 1 vehicleName = “Benz” vehicleColor = “Red” vehicleWeight = “380 kg” vehicleSpeed = “35 kmph” engineDisplacement = 145 Torque = “290 nm” fuelTankcapacity = 18 groundClearance = “210 mm” ➢ Inspection of the AutoRickshaw class ➢ void book is called with the following arguments: bookDate = “19th May 2022” chargeAmount = 1500 numberOfseats = 4 ➢ Re-inspection of the AutoRickshaw class
Expected Result	The Auto-Rickshaw would be booked.
Actual Result	The Auto-Rickshaw was booked.
Conclusion	The test is successful.

Table 4: To inspect AutoRickshaw class, book the AutoRickshaw and reinspect the AutoRickshaw class

Output Result:

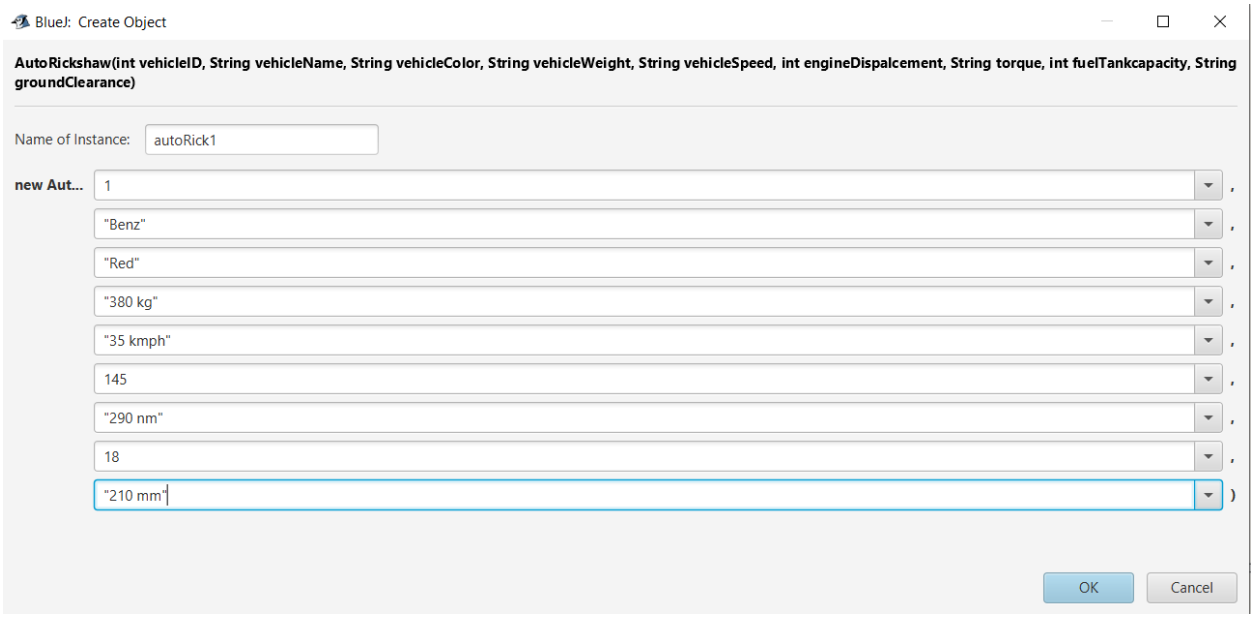

Figure 5: Assigning the data in AutoRickshaw class

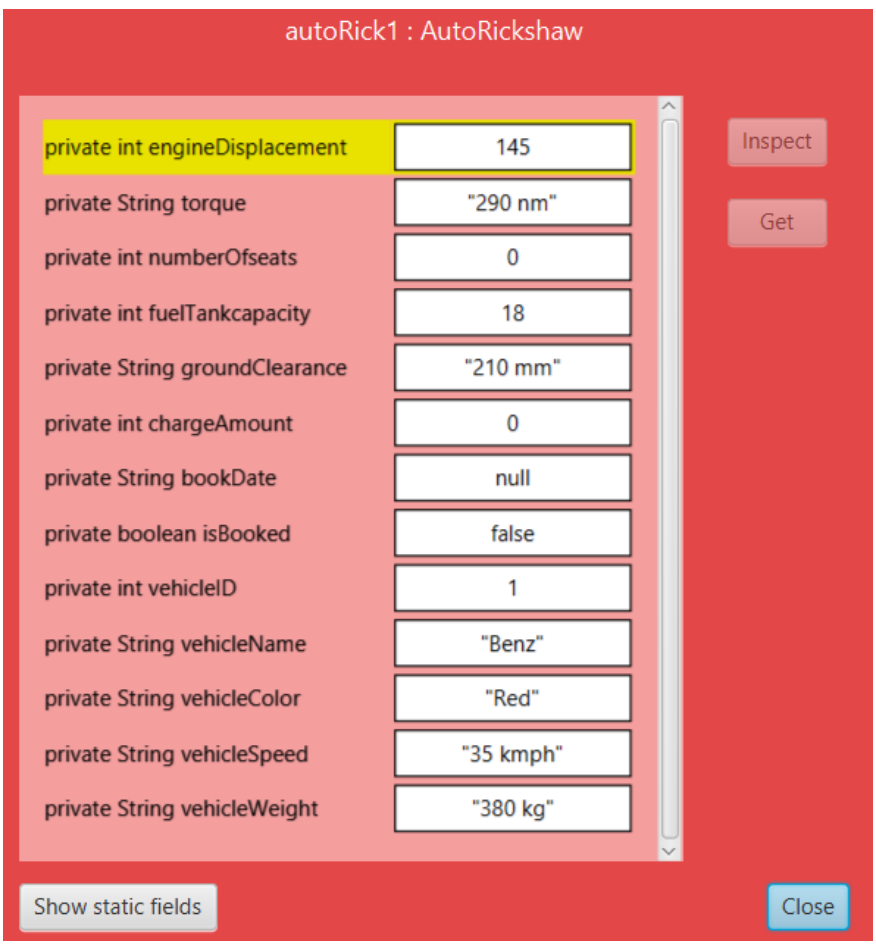

Figure 6: Inspection of AutoRickshaw class

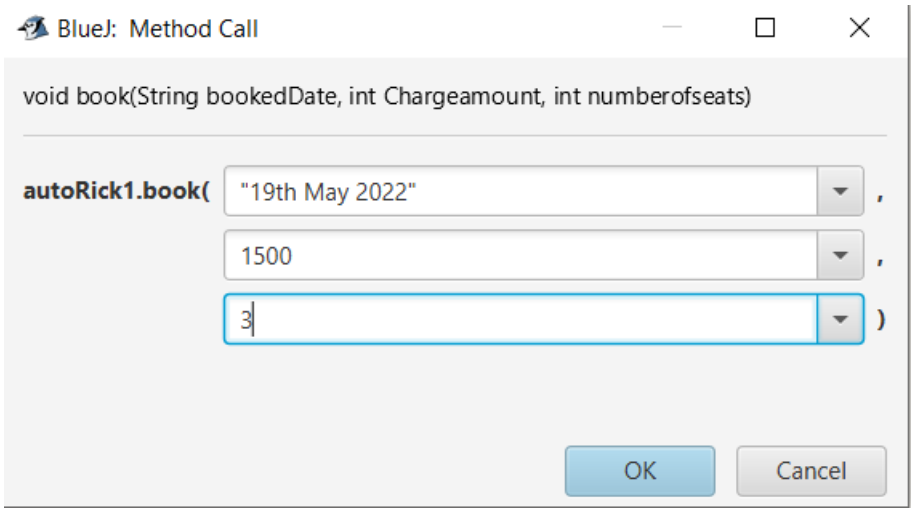

Figure 7: Assigning values in book() method

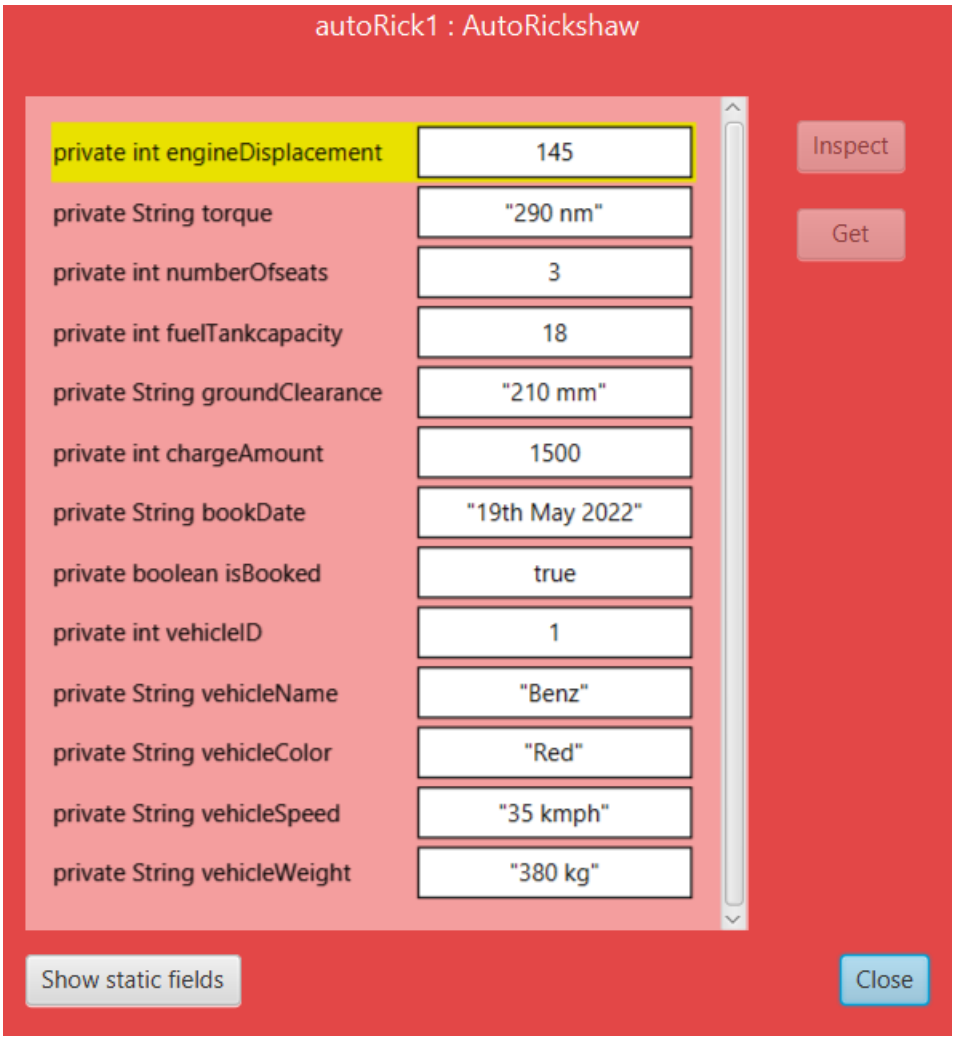

Figure 8: Reinspection of AutoRickshaw class

5.2 Test 2 – To inspect ElectricScooter class, purchase an electric scooter and re-inspect the ElectricScooter class

Test No.	2
Objective	To inspect ElectricScooter class, purchase an electric scooter and re-inspect the ElectricScooter class.
Action	➢ The ElectricScooter is called with the following arguments: vehicleID = 2 vehicleName = “Ronny” vehicleColor = “Silver” vehicleWeight = “13 kg” vehicleSpeed = “45 kmph” ➢ Inspection of the ElectricScooter class ➢ hasPurchased is called with the following arguments: brand = “Super” price = 250000 charging time = “6hrs’ mileage = “35 km/bar” range = 45 ➢ Re-inspection of the ElectricScooter class
Expected Result	The Electric Scooter would be purchased.
Actual Result	The Electric scooter was purchased.
Conclusion	The test is successful.

Table 5: To inspect ElectricScooter class, purchase an electric scooter and re-inspect the ElectricScooter class

Output Result:

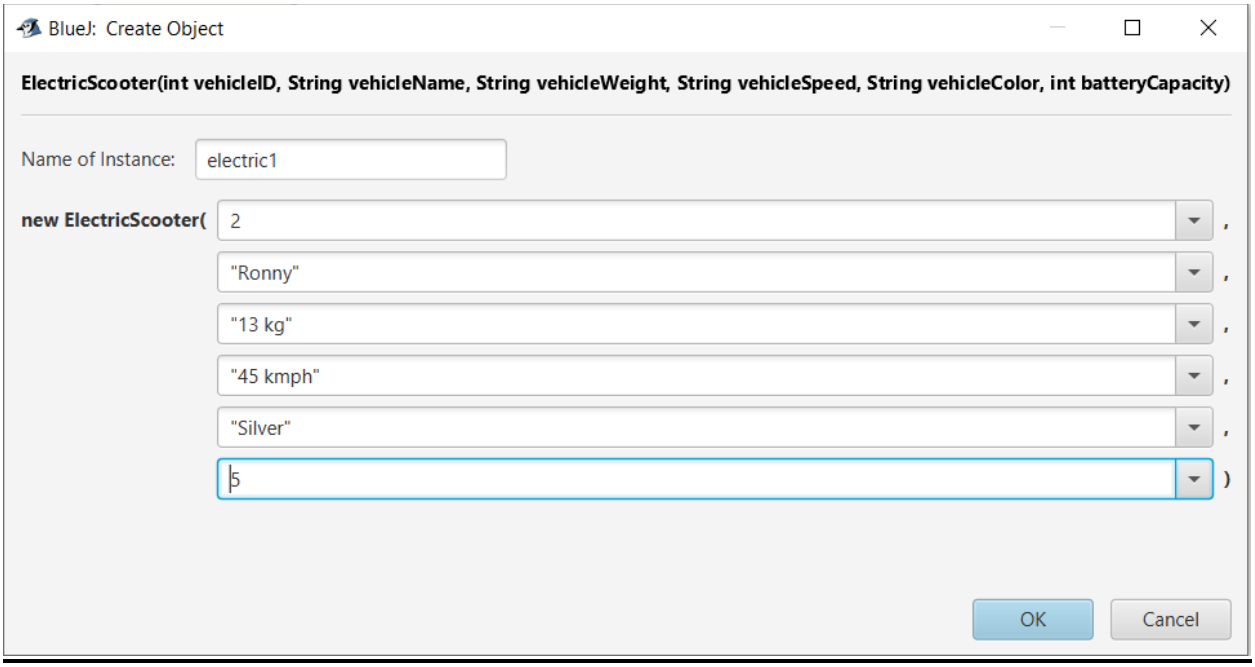

Figure 9: Assigning the data in ElectricScooter class

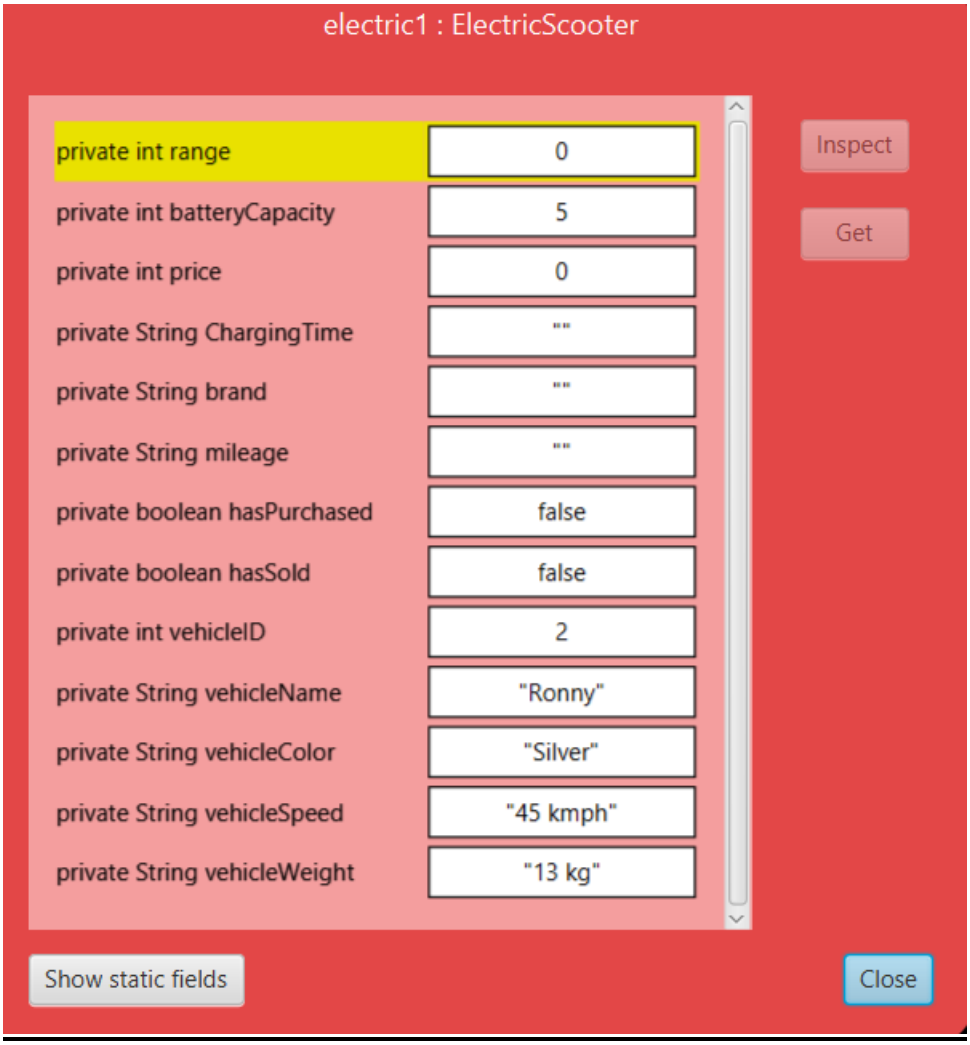

Figure 10: Inspection of ElectricScooter class

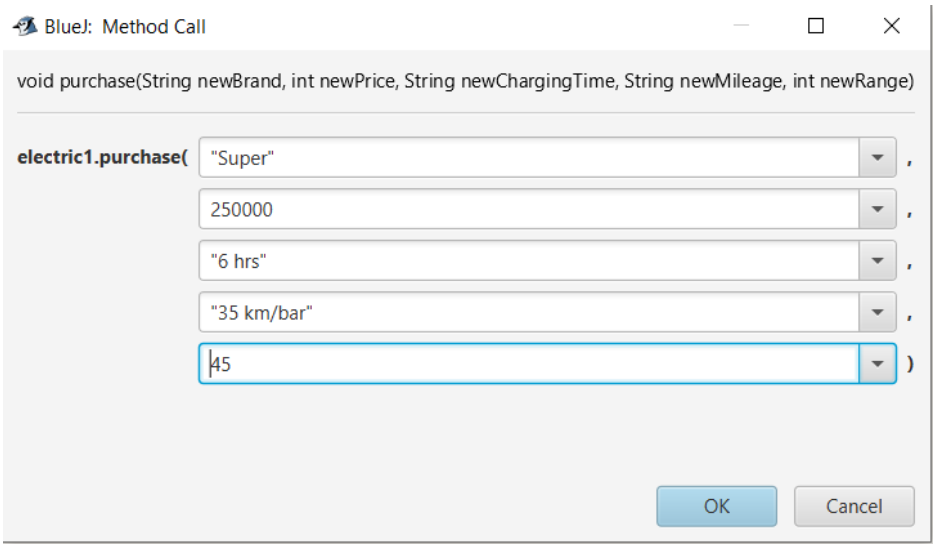

Figure 11: Assigning values in purchase() method

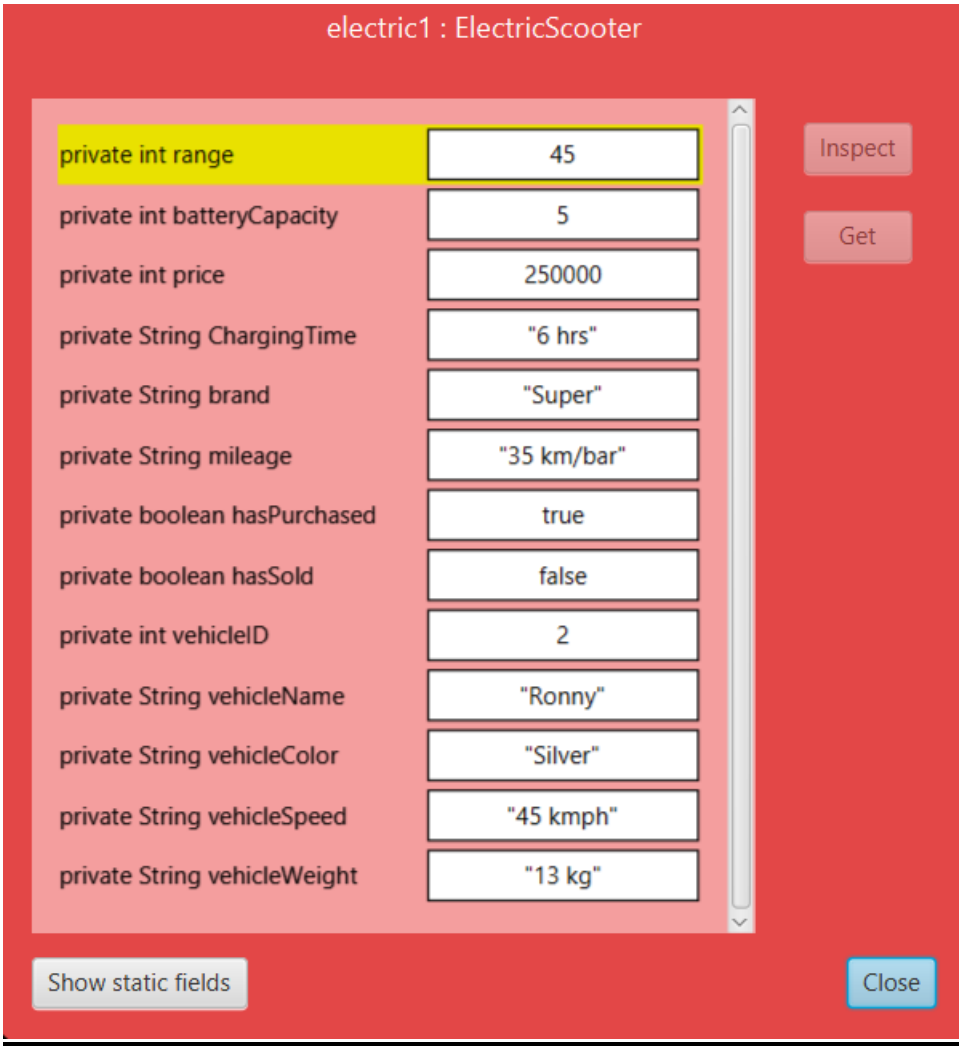

Figure 12: Reinspection of ElectricScooter class

5.3 Test 3 – To inspect ElectricScooter class again, change hasPurchased to false and re-inspect the ElectricScooter class

Test No.	3
Objective	To inspect ElectricScooter class again, change hasPushased to false and re-inspect the ElectricScooter class.
Action	➢ Inspection of the ElectricScooter class after electric scooter has been purchased. ➢ void sell is called with argument: price = 200000 ➢ Re-inspection of the ElectricScooter class
Expected Result	hasPurchased should be changed to 'false'
Actual Result	hasPurchased was changed to 'false'
Conclusion	The test is successful.

Table 6: To inspect ElectricScooter class again, change hasPurchased to false and re-inspect the ElectricScooter class

Output Result:

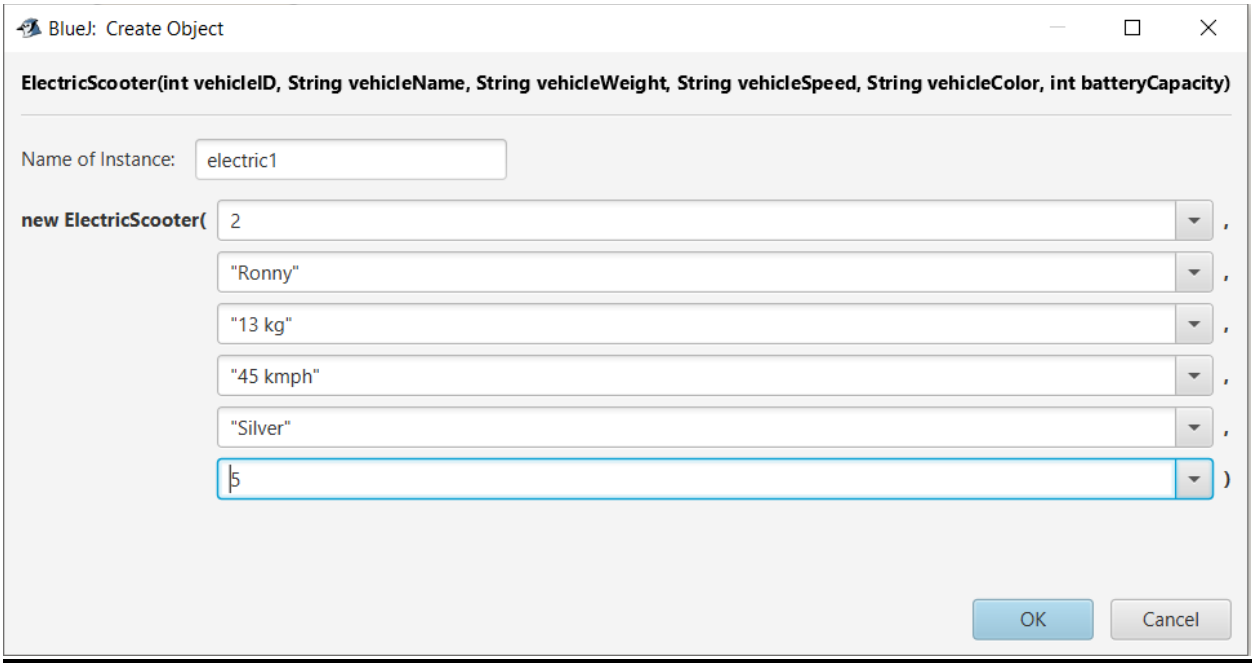

Figure 13: Assigning the data in ElectricScooter class

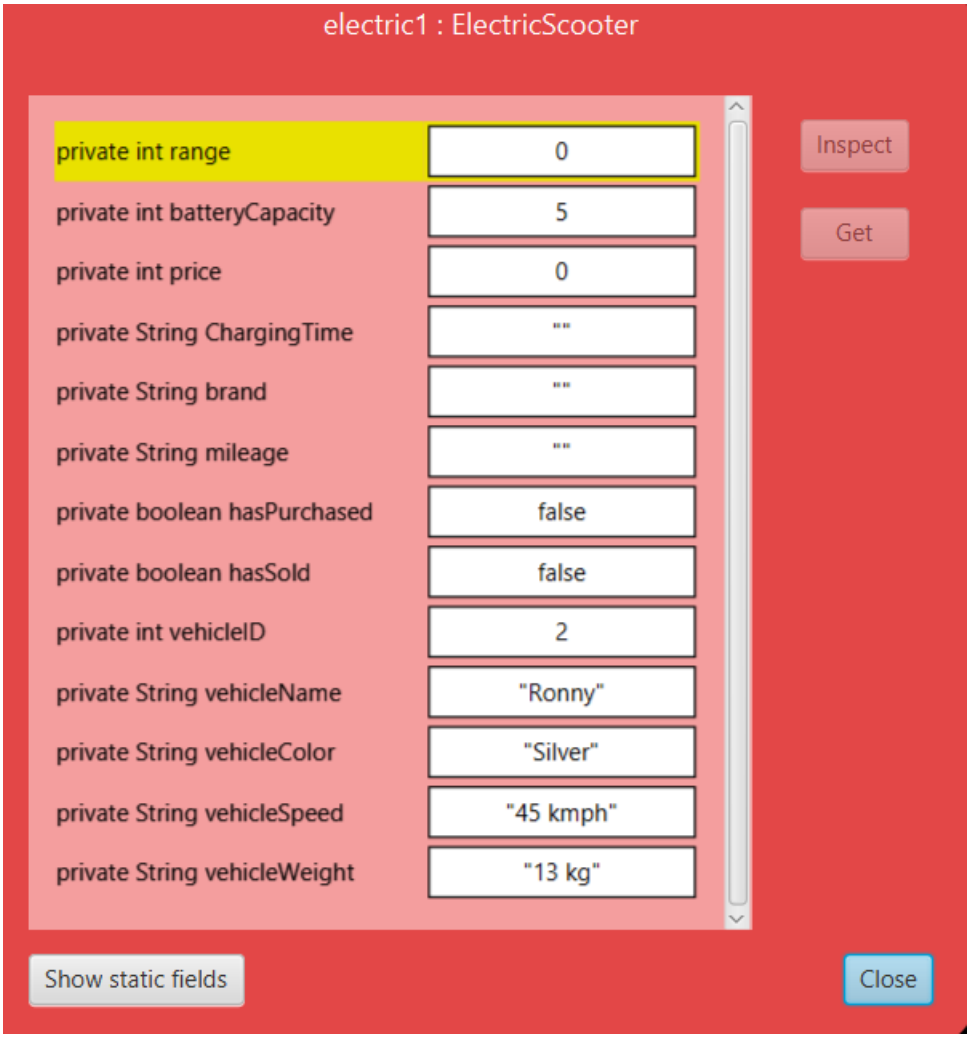

Figure 14: Inspection of ElectricScooter class

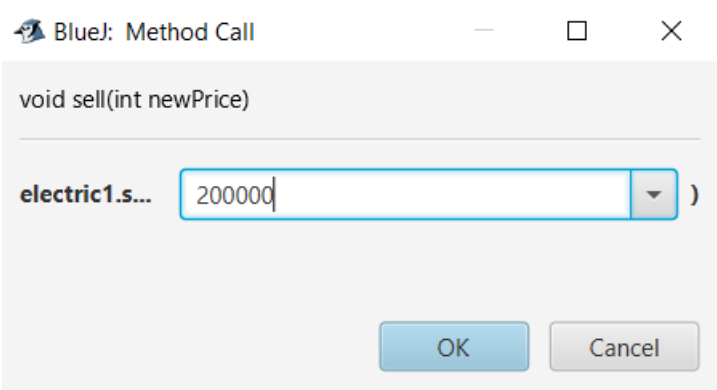

Figure 15: Assigning value in sell() method

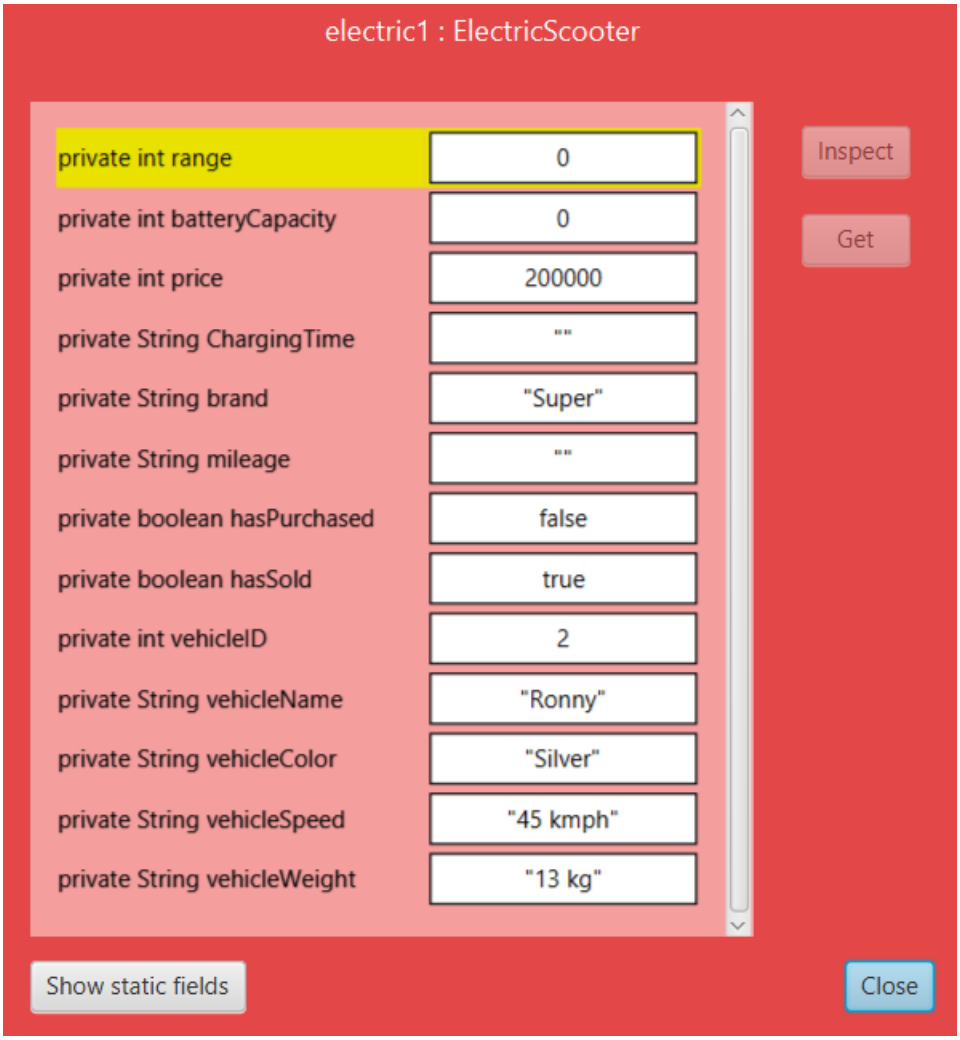

Figure 16: Reinspection of ElectricScooter class as hasPurchased is changed to "false"

5.4 Test 4 – Display the details of AutoRickshaw and ElectricScooter classes

Test No.	4
Objective	To display the details of AutoRickshaw and ElectricScooter classes.
Action	➢ The AutoRickshaw is called again with the same values along with following arguments: vehicleID = 1 vehicleName = “Benz” vehicleColor = “Red” vehicleWeight = “380 kg” vehicleSpeed = “35 kmph” engineDisplacement = 145 Torque = “290 nm” fuelTankcapacity = 18 groundClearance = “210 mm” ➢ void book is called with the following arguments: bookDate = “19th May 2022” chargeAmount = 1500 numberOfseats = 4 ➢ display() method is called to display the distails of AutoRickshaw class. ➢ The ElectricScooter is called again with the same values along with following arguments: vehicleID = 2 vehicleName = “Ronny” vehicleColor = “Silver” vehicleWeight = “13 kg” vehicleSpeed = “45 kmph” ➢ void purchase is called with the following arguments: brand = “Super” price = 250000 charging time = “6hrs’ mileage = “35 km/bar”

	range = 45 ➢ void sell is called with the following argument: price = 200000 ➢ display() method is called to display the details of ElectricScooter class.
Expected Result	The details of AutoRickshaw and ElectricScooter classes
Actual Result	The details of AutoRickshaw and ElectricScooter classes is displayed.
Conclusion	The test is successful.

Table 7: To display the displays of AutoRickshaw and ElectricScooter classes

Output Result:

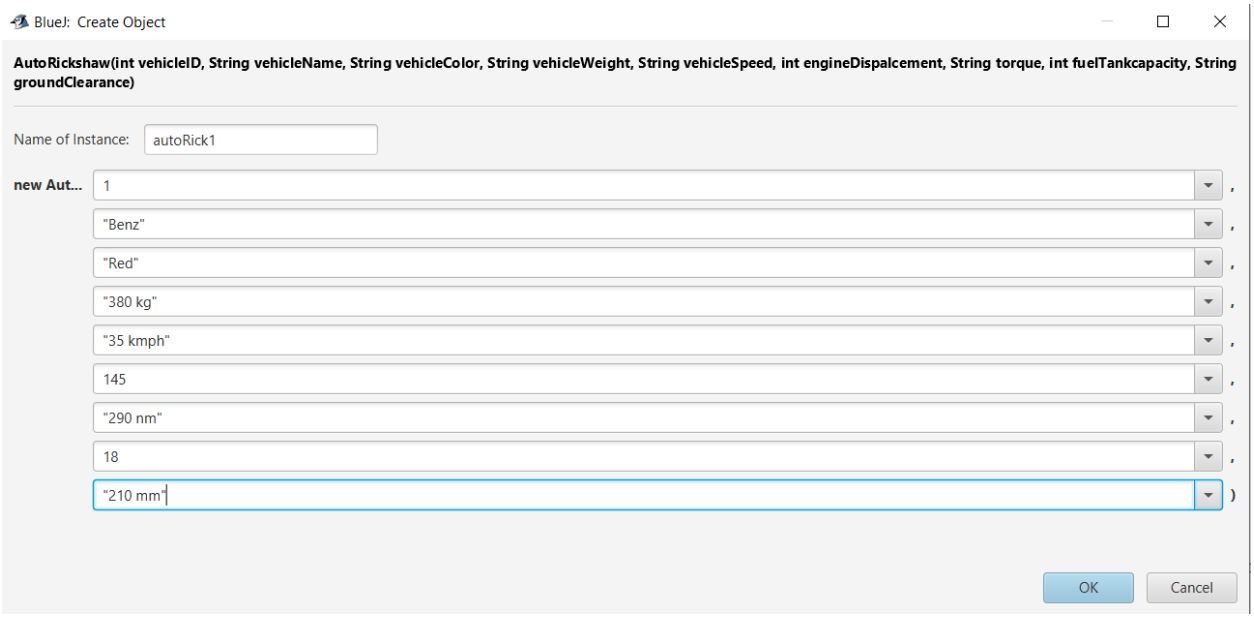

Figure 17: Assigning the data in AutoRickshaw

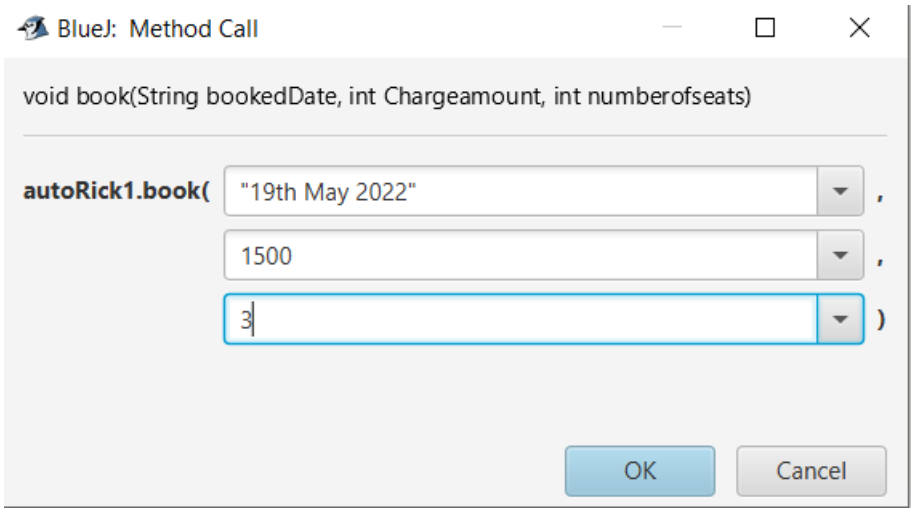

Figure 18: Assigning values in book() method

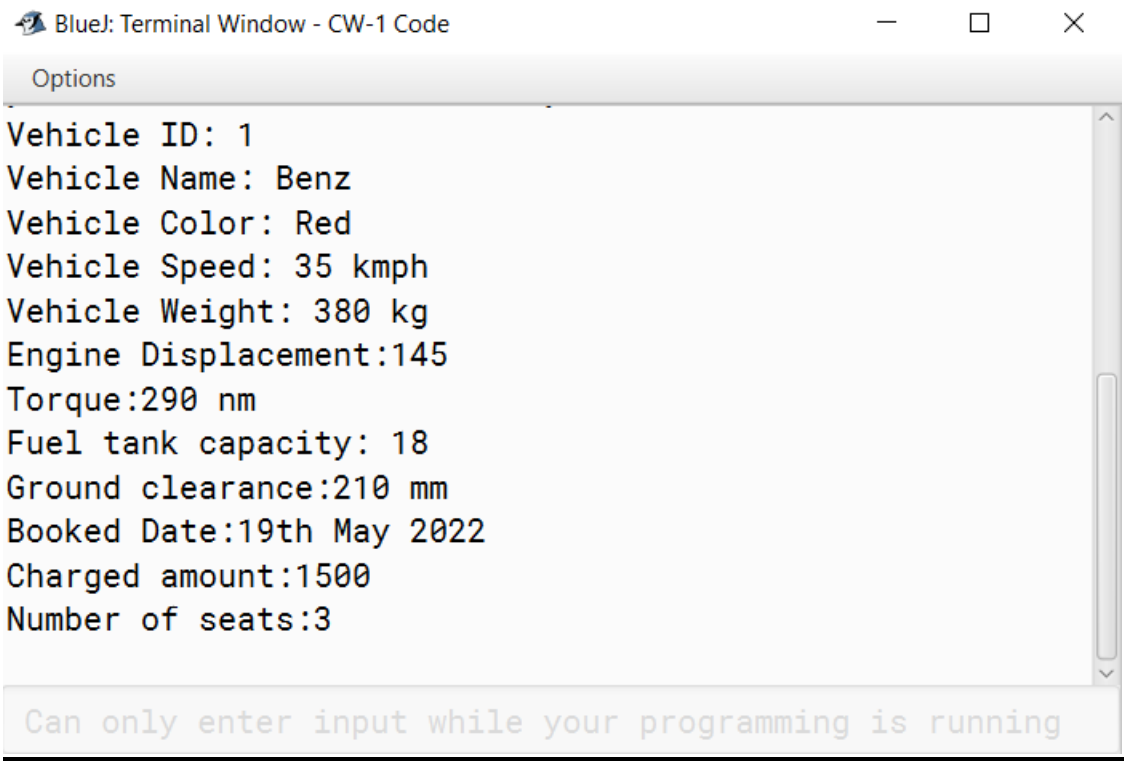

Figure 19: Displaying the details of AutoRickshaw class

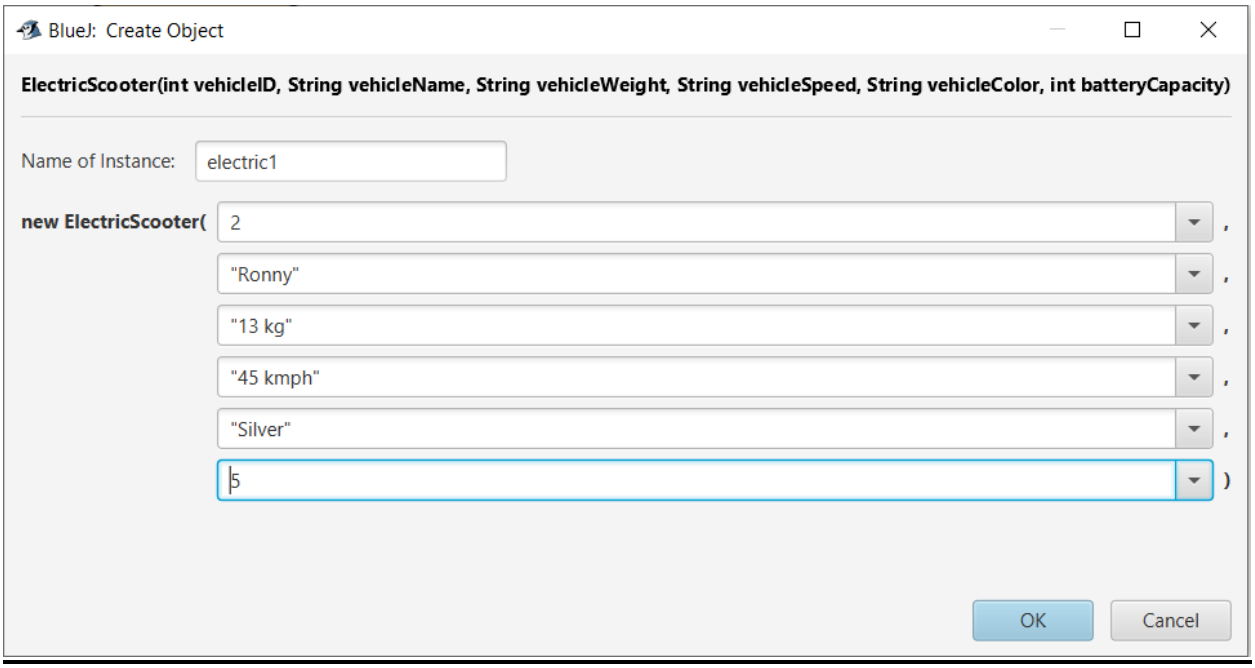

Figure 20: Assigning the data in ElectricScooter

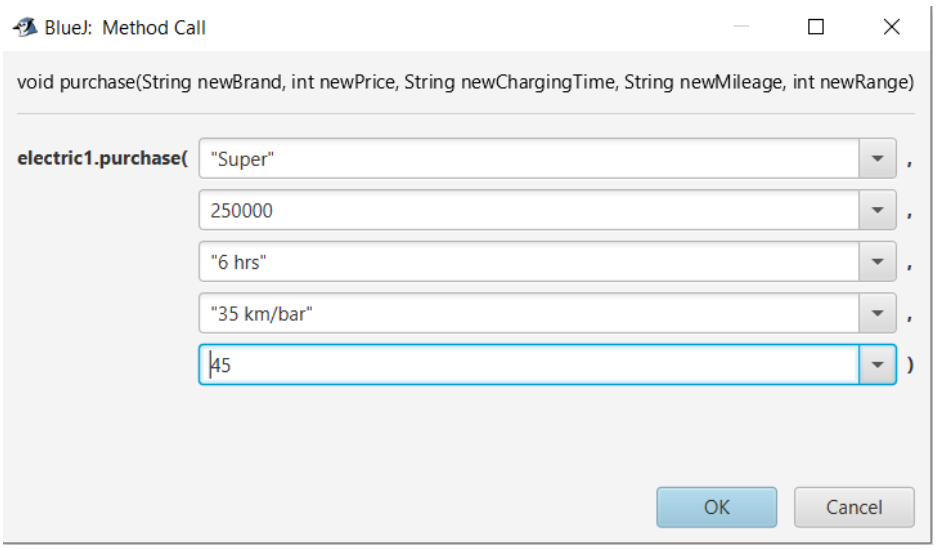

Figure 21: Assigning values in purchase() method

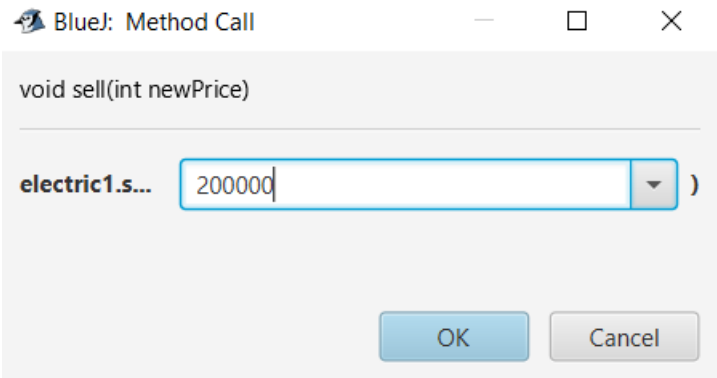

Figure 22: Assigning values in sell() method

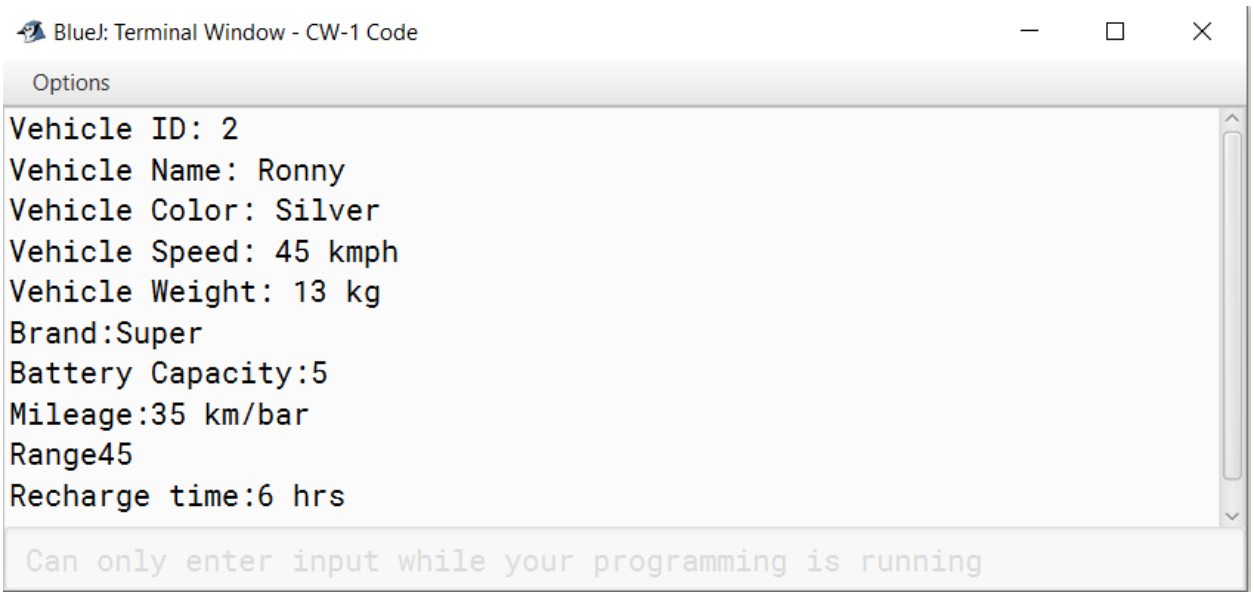

Figure 23: Displaying the details of ElectricScooter class

6. Error Detection and its Correction

Error is a problem that occurs in a program when the code has mistakes in it. In Java, an error is a mistake that occurs in the program. Programmers usually make mistakes while coding which causes for errors in the code and can be detected manually or the system shows the error in the program itself. Error is an foul operation performed by the programmer which results in the abnormal working of the program. While compiling the program, system detects errors which need to be fixed and executed accordingly.

There are three types of errors in Java:-

- **Syntax Error:-** Syntax error is the most common error in Java programming language. It occurs when programmers misuse the Java reserved words. Syntax error can also occur when variable and function names are misspelled, semicolons are missing and mismatching parenthesis.
- **Semantics Error:**- Semantics error is the error which occurs when the syntax of the code is correct but the usage is incorrect. When the variable used in the code isn't properly initialized then semantics error occurs. This type of error is mostly detected while compiling the code by the system itself.
- **Logical Error:-** Logical error is a bug which is usually caused by incorrect idea used by the programmer. Logical error returns incorrect result as the program is compiled but the errors are not detected. This type of error is very dangerous as the system is unable to find the error while compiling and needs to be detected manually.

6.1 Error 1 – Syntax Error

The Error which was detected first was syntax error where the instance variable should end with a semicolon after calling it.

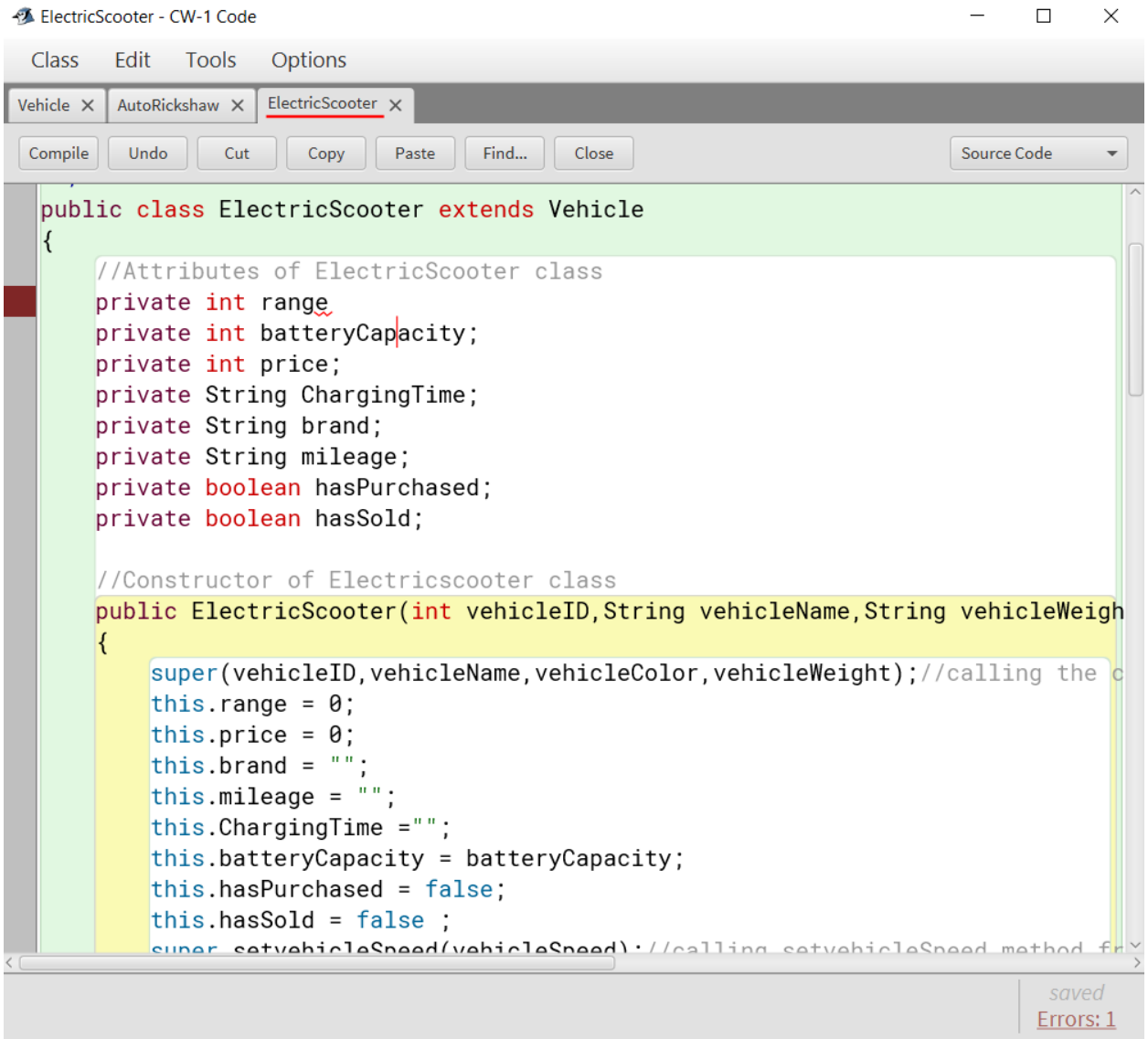

Figure 24: Error 1

6.2 Correction of Error 1

As we know that after declaring a variable it needs to end with a semicolon which was missing. So, the error was corrected by adding semicolon after the calling the variable.

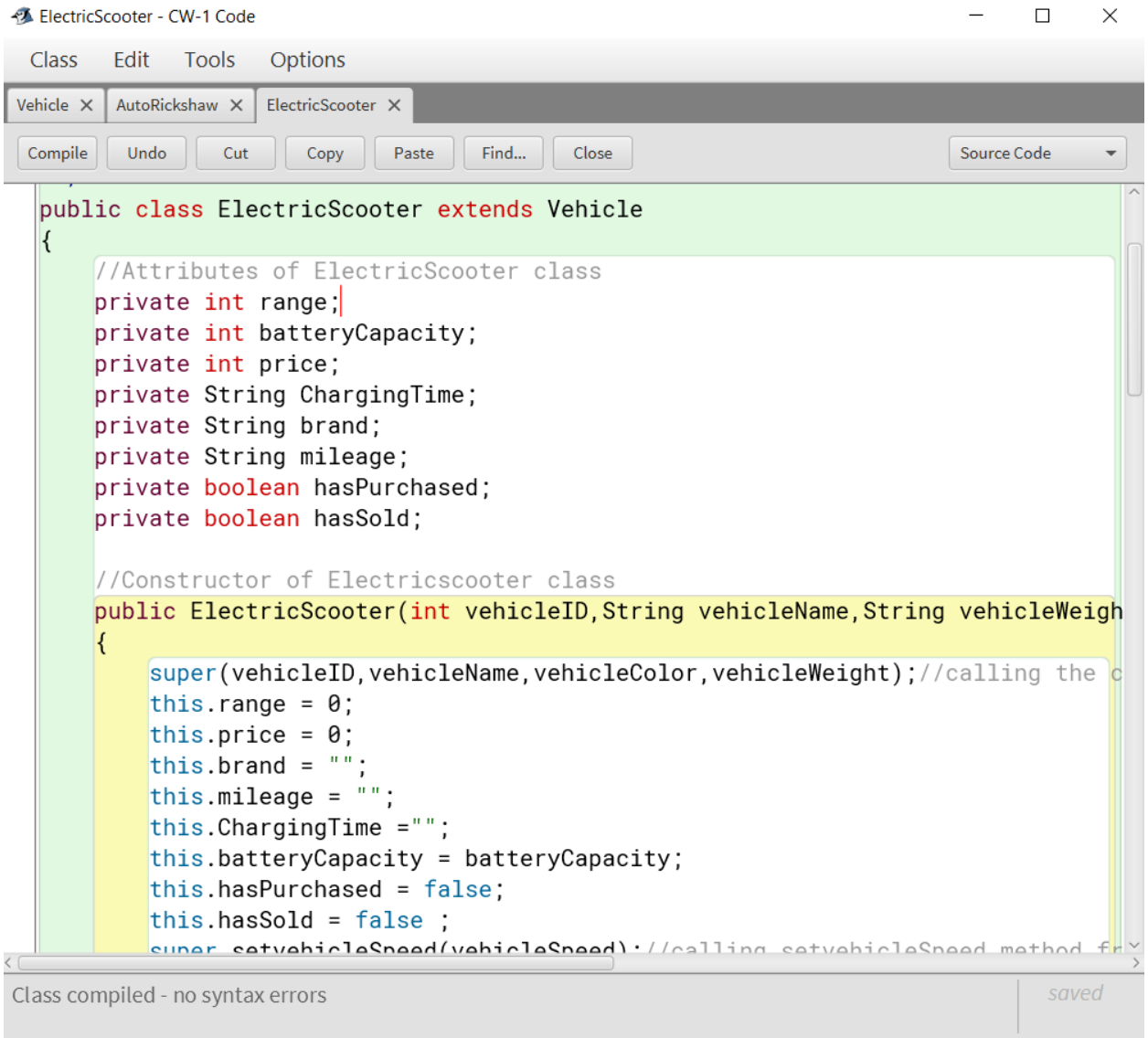

Figure 25: Correction of Error 1

6.3 Error 2 – Semantics Error

Semantics Error was the error which was detected next where I assigned a integer value to a String while initializing ChargingTime.

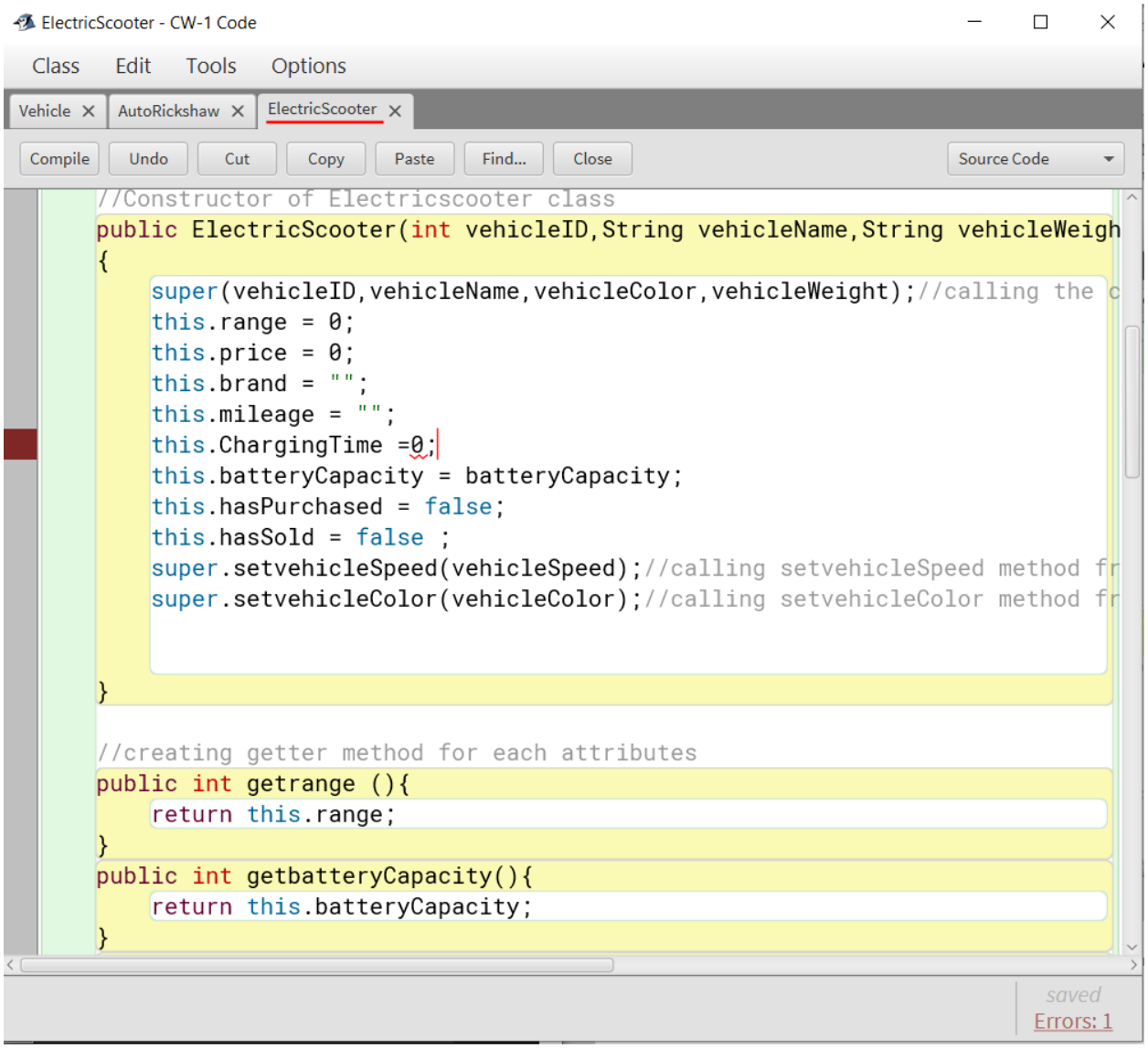

Figure 26: Error 2

6.4 Correction of Error 2

This semantic error was corrected when the integer value was changed to String data type.

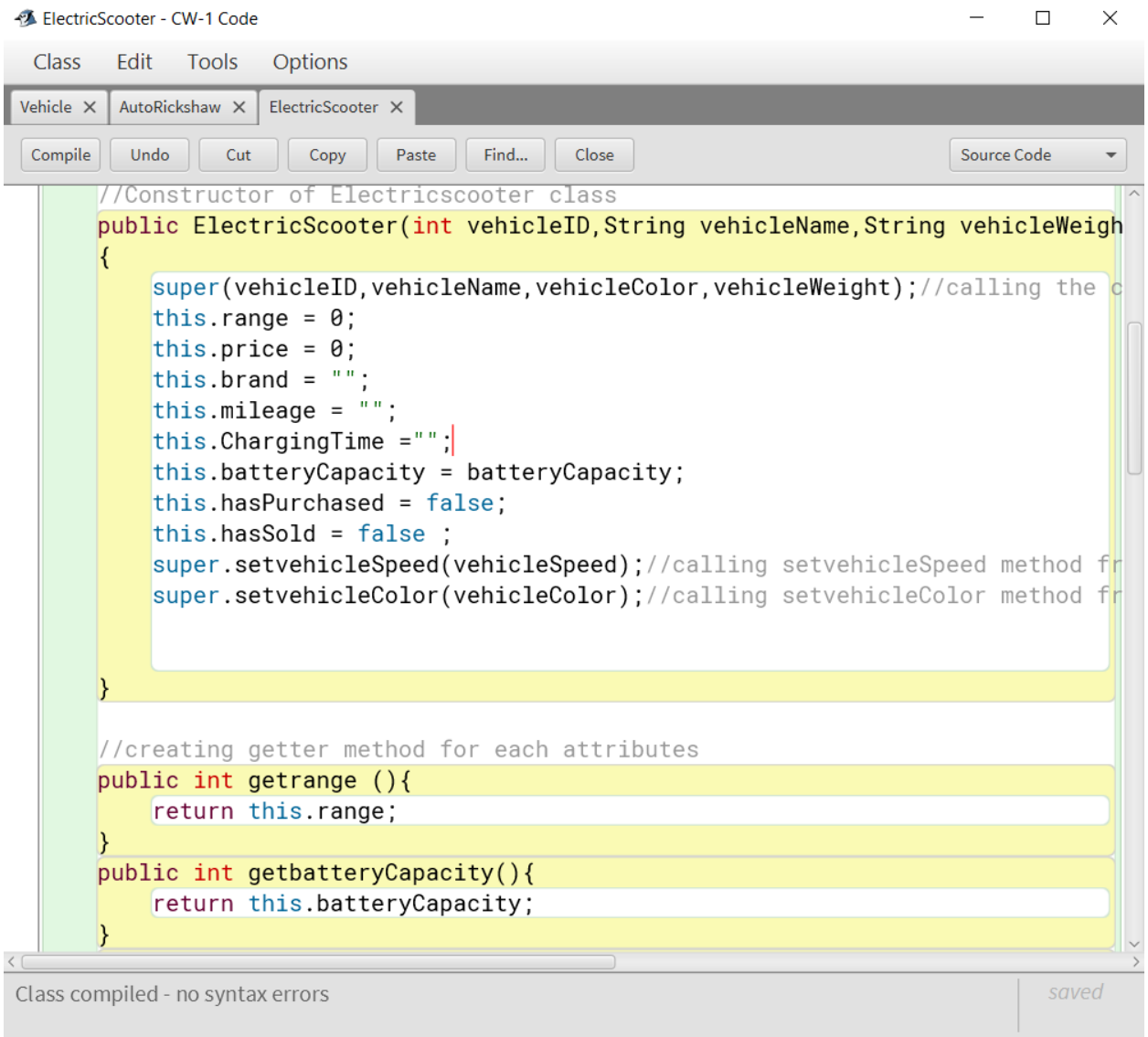

Figure 27: Correction of Error 2

6.5 Error 3 – Logical Error

Logical Error was the third error which was detected where hasPurchased was assigned as false. But as per our requirements, hasPurchased should be assigned as “true”.

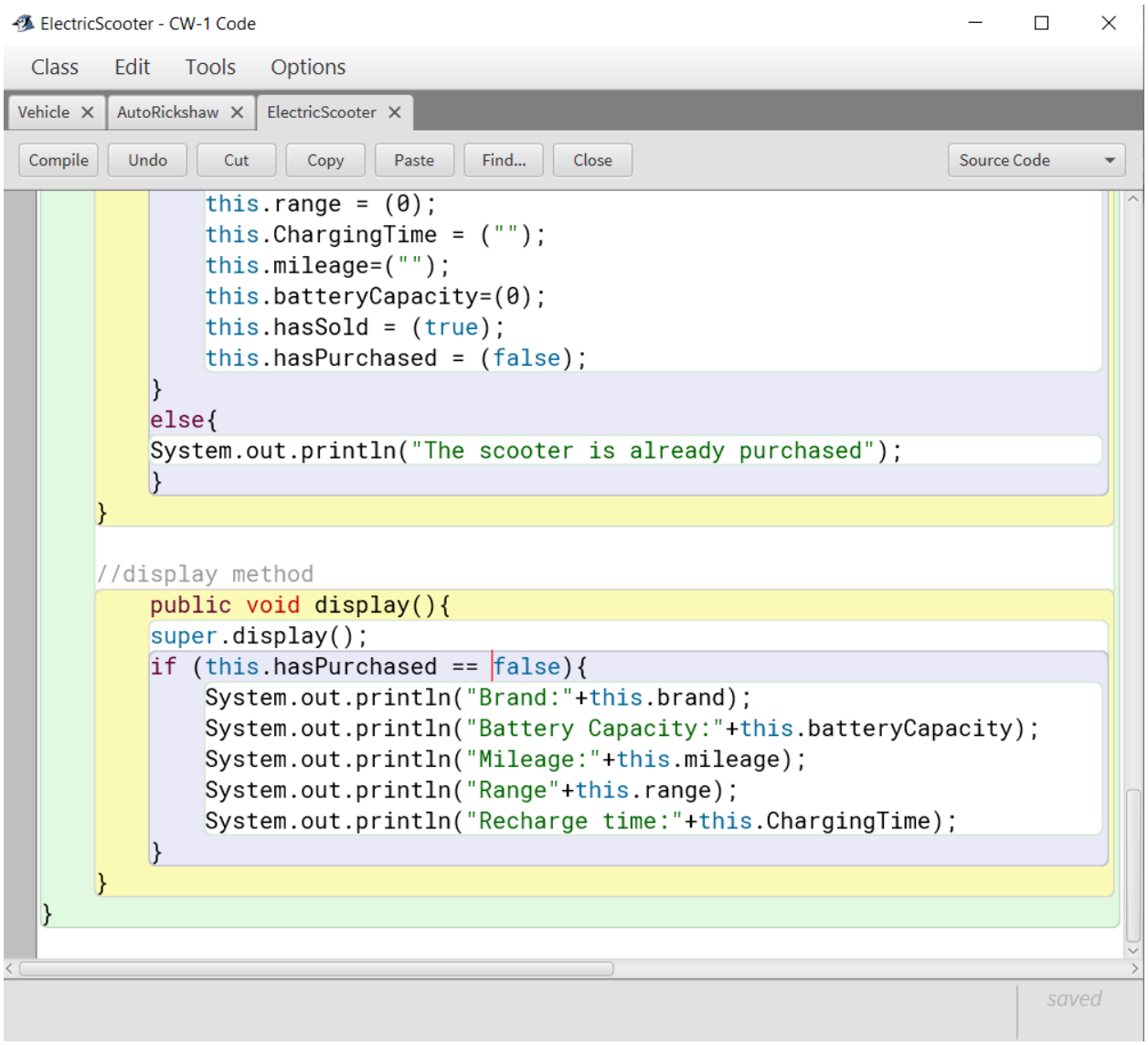

Figure 28: Error 3

6.6 Correction of Error 3

This error can be corrected by assigning “true” instead of “false” in hasPushased.

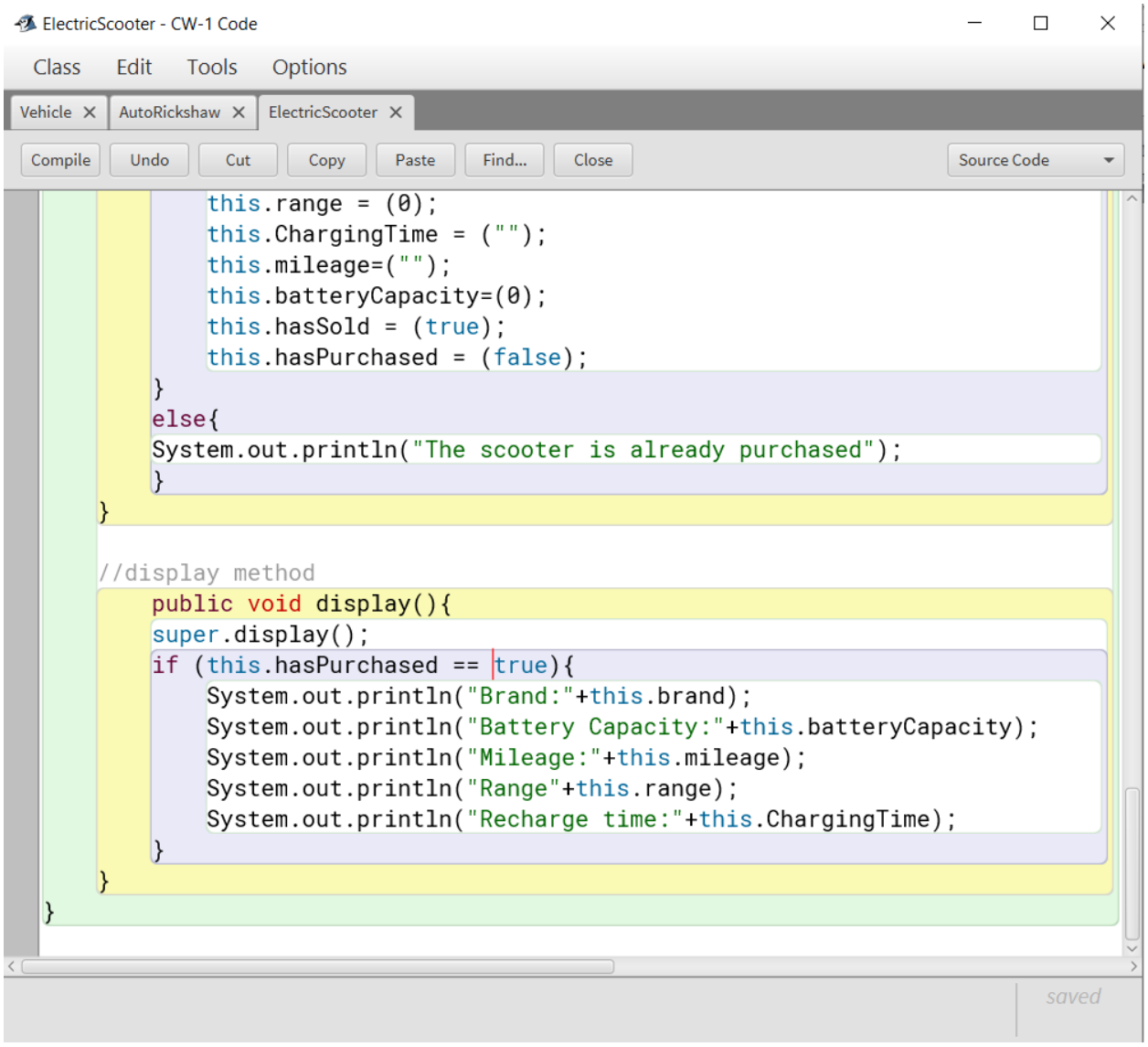

Figure 29: Correction of Error 3

7. Conclusion

This coursework had a huge impact on my understanding of Java. I started off by coding through all the details that were given and made the program. It was going smooth and sorrow but there were some difficulties along the way. This module taught me how to do the coursework with utmost sincerity and dedication. The teachers were clear and taught everything that was needed. Teachers also gave all the answers to queries that I had.

This coursework taught a lot of things in the grand scheme. This assignment also made me clear about parent class and its child class as the parent class is the tree and the child classes are the branches. The child class extends the parent class which allows the child class to inherit the properties from parent class. The concept of getters and setters were also understood through this project.

There were quiet a lot of difficulties that I encountered while completing my coursework. The concept of getters and setters were a bit difficult to understand at first. I got so many errors in while coding and the output sometimes showed the same value to different variables. Sometimes it showed random errors because of incorrect naming of methods and variables which was frustrating, but at the same time these errors taught me about my mistakes and not repeat those mistakes again.

I overcame these difficulties by asking the teachers about it, consulting with friends, learning from the course content given on Google Classroom and searching about it online. Teachers of this particular module helped a lot through and throughout who provided with all the answers to my difficulties.

8. Bibliography

GeeksforGeeks. (2020, July 1). *Introduction of BlueJ*. Retrieved May 17, 2022, from geeksforgeeks: https://www.geeksforgeeks.org/introduction-of-bluej/

GeeksforGeeks. (2022, March 7). *Methods in Java*. Retrieved May 17, 2022, from Geeksfor Geeks: https://www.geeksforgeeks.org/methods-in-java/

IBM Cloud Education. (2019, May 8). *What is Java?* Retrieved May 17, 2022, from ibm: https://www.ibm.com/cloud/learn/java-explained#:~:text=Java%20is%20a%20widely%20used,the%20C%20and%20C%2B%2B%20languages.

Metwalli, S. A. (2021, Januaru 2). *Pseudocode 101: An Introduction to Writing Good Pseudocode*. Retrieved May 17, 2022, from towardsdatascience: https://towardsdatascience.com/pseudocode-101-an-introduction-to-writing-good-pseudocode-1331cb855be7

Schildt, H. (2017). *Java: The Complete Reference* (10th ed.). New York: McGraw-Hill Education.

Wilkins, J. (2021, October 8). *What is Computer Programming?* Retrieved May 17, 2022, from freecodecamp: https://www.khanacademy.org/computing/computer-programming/programming/intro-to-programming/v/programming-intro#:~:text=Programming%20is%20the%20process%20of,%2C%20Python%2C%20and%20C%2B%2B.

9. Appendix

9.1 Vehicle Class

```
/**
 * @author (NP01CP4S220137 Pramit Badgami)
 * @version (1.0.0)
 */
public class Vehicle
{
    //Attributes of Vehicle class
    private int vehicleID;
    private String vehicleName;
    private String vehicleColor;
    private String vehicleSpeed;
    private String vehicleWeight;

    //Constructor
    public Vehicle (int vehicleID, String vehicleName, String vehicleColor, String
vehicleWeight )
    {

        //Initializing variables
        this.vehicleID = vehicleID; //values are signed in variable
        this.vehicleName= vehicleName;
        this.vehicleColor= vehicleColor;
```

```
        this.vehicleWeight= vehicleWeight;
    }

    //creating getter method for each attributes
    public int getvehicleID()//getter method
    {
        return this.vehicleID;
    }

    public String getvehicleName()
    {
        return this.vehicleName;
    }

    public String getvehicleColor()
    {
        return this.vehicleColor;
    }

    public String getvehicleSpeed()
    {
        return this.vehicleSpeed;
    }

    public String getvehicleWeight()
```

```
    {
        return this.vehicleWeight;
    }

    //creating setter method to set the vehicle speed
    public void setvehicleSpeed(String newSpeed)//setter method
    {
        this.vehicleSpeed = newSpeed;
    }

    ////creating setter method to set the vehicle color
    public void setvehicleColor(String newColor)
    {
        this.vehicleColor = newColor;
    }

    //creating display method
    public void display()
    {

        System.out.println("Vehicle ID: " + this.vehicleID);
        System.out.println("Vehicle Name: " + this.vehicleName);
        System.out.println("Vehicle Color: " + this.vehicleColor);
        System.out.println("Vehicle Speed: " + this.vehicleSpeed);
```

```
        //Whether vehile weight is empty or not
        if(vehicleWeight == null || vehicleWeight == ""|| vehicleWeight == "0") {
            System.out.println("Vehicle weight is empty");
        }
        else {
            System.out.println("Vehicle Weight: " + this.vehicleWeight);
        }
    }
}
```

9.2 AutoRickshaw Class

```
/**
 * @author (NP01CP4S220137 Pramit Badgami)
 * @version (1.0.0)
 */
public class AutoRickshaw extends Vehicle
{
   //Attributes of Vehicle class
   private int engineDisplacement;
   private String torque;
   private int numberOfseats;
   private int fuelTankcapacity;
   private String groundClearance;
   private int chargeAmount;
   private String bookDate;
   private boolean isBooked;

   //Constructor
   public AutoRickshaw (int vehicleID,String vehicleName,String vehicleColor,String
vehicleWeight ,String vehicleSpeed,int engineDispalcement,String torque
                        ,int fuelTankcapacity,String groundClearance)
   {

         //calling the constructor of Vehicle class
         super(vehicleID, vehicleName, vehicleColor, vehicleWeight);
```

```
        //Initializing variables
        this.engineDisplacement = engineDispalcement;
        this.torque = torque;
        this.fuelTankcapacity = fuelTankcapacity;
        this.groundClearance= groundClearance;
        super.setvehicleSpeed(vehicleSpeed);
        super.setvehicleColor(vehicleColor);
        this.isBooked= false;
       }

  //creating getter method for each attributes
  public int getengineDispalcement()
  {
      return this.engineDisplacement;
  }

  public String gettorque()
  {
      return this.torque;
  }

  public int getnumberOfseats()
  {
      return this.numberOfseats;
  }
```

```
public int getfuelTankcapacity()
{
    return this.fuelTankcapacity;
}

public String getgroundClearance()
{
    return this.groundClearance;
}

public int getchargeAmount()
{
    return this.chargeAmount;
}

public String getbookDate()
{
    return this.bookDate;
}

public boolean isBooked()
{
    return this.isBooked;
}
```

```
//creating setter method to set the charge amount
public void setchargeAmount(int newChargeamount)
{
    this.chargeAmount= newChargeamount;
}
////creating setter method to set the number of seats
public void setnumberOfseats(int newNumberOfseats)
{
    this.numberOfseats= newNumberOfseats;
}

//method used to book
public void book (String bookedDate,int Chargeamount, int numberofseats)
{

    if (this.isBooked== false ){
        this.bookDate= bookedDate;
        System.out.println("Your autorickshaw"+ super.getvehicleID()+"is now booked");
        setchargeAmount(Chargeamount);
        setnumberOfseats(numberofseats);
        isBooked= true;
        System.out.println("your autorickshaw "+ super.getvehicleID()+"is already
booked");
    }
```

```
        else{
            System.out.println("the status of isBooked is: " +isBooked);
            System.out.println("your autorickshaw "+ super.getvehicleID()+"is already
booked");
        }
    }

   public void display()
   {

        //calling display method from Vehilce class
        super.display();
        if (this.isBooked == true ){
            System.out.println("Engine Displacement:"+ this.engineDisplacement);
            System.out.println("Torque:"+this.torque);
            System.out.println("Fuel tank capacity: "+ this.fuelTankcapacity);
            System.out.println("Ground clearance:"+ this.groundClearance);
            System.out.println("Booked Date:"+ this.bookDate);
        }
        if (this.chargeAmount==0){
            System.out.println("Charge amount is not recorded");
        }
        else{
            System.out.println("Charged amount:" + this.chargeAmount);
            }
```

```
        if (this.numberOfseats == 0){
            System.out.println("Number of seats are not recorded");
        }
        else{
            System.out.println("Number of seats:"+ this.numberOfseats);
        }
    }


}
```

9.3 ElectricScooter Class

```
/**
 * @author (NP01CP4S220137 Pramit Badgami)
 * @version (1.0.0)
 */
public class ElectricScooter extends Vehicle
{
    //Attributes of ElectricScooter class
    private int range;
    private int batteryCapacity;
    private int price;
    private String ChargingTime;
    private String brand;
    private String mileage;
    private boolean hasPurchased;
    private boolean hasSold;

    //Constructor of Electricscooter class
    public ElectricScooter(int vehicleID,String vehicleName,String vehicleWeight, String
vehicleSpeed, String vehicleColor,int batteryCapacity )
    {

        //calling the constructor of Vehicle class
        super(vehicleID,vehicleName,vehicleColor,vehicleWeight);
        this.range = 0;
```

```
        this.price = 0;
        this.brand = "";
        this.mileage = "";
        this.ChargingTime ="";
        this.batteryCapacity = batteryCapacity;
        this.hasPurchased = false;
        this.hasSold = false ;
        super.setvehicleSpeed(vehicleSpeed);//calling setvehicleSpeed method from
Vehicle class
        super.setvehicleColor(vehicleColor);//calling setvehicleColor method from Vehicle
class
    }

    //creating getter method for each attributes
    public int getrange ()
    {
        return this.range;
    }

    public int getbatteryCapacity()
    {
        return this.batteryCapacity;
    }

    public int getprice ()
    {
```

```
        return this.price;
    }

    public String getChargingTime()
    {
        return this.ChargingTime;
    }

    public String getbrand()
    {
        return this.brand;
    }

    public String getmileage()
    {
        return this.mileage;
    }

    public boolean gethasPurchased()
    {
        return this.hasPurchased;
    }

    public boolean gethasSold()
    {
```

```
        return this.hasSold;
    }

    //creating setter method to set the brand
    public void setBrand(String newBrand)
    {

        if(this.hasPurchased == false){
            this.brand= newBrand;
        }
        else{
            System.out.println("Brand cannot be changed");
        }
    }

    //method used to purchase
    public void purchase(String newBrand,int newPrice,String newChargingTime, String
newMileage, int newRange)
    {

        if (this.hasPurchased == false){
            setBrand(newBrand);
            this.price=newPrice;
            this.ChargingTime= newChargingTime;
            this.mileage=newMileage;
```

```
            this.range=newRange;
            this.hasPurchased = true;
        }
        else{
            System.out.println("has already been purchased");
        }
    }

    //method used to sell
    public void sell(int newPrice)
    {

        if (this.hasSold == false){
            this.price = newPrice;
            this.range = (0);
            this.ChargingTime = ("");
            this.mileage=("");
            this.batteryCapacity=(0);
            this.hasSold = (true);
            this.hasPurchased = (false);
        }
        else{
        System.out.println("The scooter is already purchased");
        }
    }
```

```
    //creating display method
    public void display()
    {

        //calling display method from Vehilce class
        super.display();
        if (this.hasPurchased == true){
            System.out.println("Brand:"+this.brand);
            System.out.println("Battery Capacity:"+this.batteryCapacity);
            System.out.println("Mileage:"+this.mileage);
            System.out.println("Range"+this.range);
            System.out.println("Recharge time:"+this.ChargingTime);
        }
    }
}
```