Parent Class Animal

This is the parent class Animal where String name and String category is declared with private access modifier. This will limit the access of these strings within the class.

Then constructor is made which takes values for two strings declared.

Getter methods(that return the values of the variables) are made which are getName() and getCategory().

Setter methods (that change the values of the variables) are made which are setName() and setCategory().

```
package workshop3;


public class Animal {
    private String name;
    private String category;

    // constructor
    public Animal(String name, String category) {
        this.name = name;
        this.category = category;
    }

    // getter methods
    public String getName() {
        return name;
    }

    public String getCategory() {
        return category;
    }

    // setter methods
    public void setName(String name) {
        this.name = name;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}
```

Child Class Dog

This dog class is the child class of the Animal parent class. This Dog class inherits all the attributes and methods of the Animal parent class. This class extends the attributes with an

addition of String color in its attributes.
Then the constructor takes all the variables of the parent class's constructor along with the String color.

Then there is a getter and setter method that returns and sets the value of the String color.

```
package workshop3;


public class Dog extends Animal {
    private String color;

    // constructor
    public Dog(String color, String name, String category) {
        super(name, category);
        this.color = color;
    }

    // getter methods
    public String getColor() {
        return color;
    }

    // setter methods
    public void setColor(String color) {
        this.color = color;
    }
}
```

Parent Class PalindromeLogic

This is the parent class that has one variable String text with private access modifier that limits the access of this variable within this PalindromeLogic class.

Then constructor takes one String text as argument.

Getter and setter methods of the String text is made which are getText() and setText() which returns and sets the value of String text respectively.

Then a method checkPalindrome() is made which returns a Boolean value. There a variable int last is initialized with which stores the length of the string – 1 to match the end of the index.
Another empty String revWord is initialized to store the reversed string.
Then a for loop is made where the revWord is stored with reverse of the string.
With each loop one character from the end of the String is taken and added to the revWord String.
Then a if condition is made which checks if the original String is equal to the reversed string. If they are equal then the method returns true else false.

```
package workshop3;

public class PalindromeLogic {
```

```
    private String text;

    // constructor
    public PalindromeLogic(String text) {
        this.text = text;
    }

    // getter method
    public String getText() {
        return text;
    }

    // setter Method
    public void setText(String text) {
        this.text = text;
    }

    public boolean checkPalindrome() {
        int last = text.length() - 1;
        String revWord = "";
        for (int i = last; i >= 0; i--) {
            revWord = revWord + text.charAt(i);
        }
        if (revWord.equals(text)) {
            return true;
        } else {
            return false;
        }
    }
}
```

Child Class Palindrome

This is a child class which extends the PalindromeLogic class. It takes a variable String text with access modifier private which limits the access of the variable within the class Palindrome.

Then the constructor of Palindrome class inherits the constructor of the parent class Palindrome with super keyword.

Then there is a method of void return type there this keyword is used to give value to the instance variable text from the String text variable from the parent class using the super keyword and getter method (getText()) from the parent class Palindrome logic.
Then a if else condition is made where the condition takes true or false from the checkPalindrome() method of the parent class using super keyword.
If the return Boolean is true then it prints the provided text is a palindrome else it prints the provided text is not a palindrome.

```
package workshop3;

public class Palindrome extends PalindromeLogic {
    private String text;
```

```
    // constructor
    public Palindrome(String text) {
        super(text);
    }

    public void printResult() {
        this.text = super.getText();
        if (super.checkPalindrome()) {
            System.out.println(text + " is a palindrome");
        } else {
            System.out.println(text + " is not a palindrome");
        }

    }
}
```

Parent Class PrimeLogic

This is a parent class which has variable int number declared with private access modifier. This limits the access of the variable inside of this class.

Then the constructor is made which takes one int number argument and assigns the value to the instance variable number.

Then getter and setter method getNumber() and setNumber() is made which returns and changes the value of the variable int number.

Then a method checkPrime() which returns boolean is made. There is a int range variable which stores the half of the input number to run the loop to the half of the number. Then a for loop is made which checks if the number is perfectly divisible by any of the numbers. If it is not perfectly divisible by the numbers then it returns true else it returns false.

```
package workshop3;

public class PrimeLogic {
    private int number;

    //constructor
    public PrimeLogic(int number){
        this.number = number;
    }

    //getter method
    public int getNumber(){
        return number;
    }

    //setter methods
```

```
    public void setNumber(int number){
        this.number = number;
    }

    public boolean checkPrime(){
        int range = number/2;
        for(int i = 2; i < range; i++){
            if(number % i == 0){
                return false;
            }
        }
        return true;
    }
}
```

Child Class Prime

This is the child class of the parent class PrimeLogic which is extended. Here is no instance variables.
Here is a constructor made which takes all the arguments as the constructor of the parent class. Then there is a method printPrimeOrNot() which doesn't return any thing as it is of void type. There is a int number variable which stores the number from the parent class using the super keyword and getNumber() getter method. Then a if else condition is made which checks if the number is prime or not by the help of parent class's method checkPime() through the use of super keyword. If the number is prime then it prints the number is prime else it prints it is not prime.

```
package workshop3;


public class Prime extends PrimeLogic {
    //constructor
    public Prime(int number) {
        super(number);
    }

    public void printPrimeOrNot(){
        int number = super.getNumber();
        if(super.checkPrime()){
            System.out.println("The Given number " + number + " is Prime."
);
        }else{
            System.out.println("The Given number " + number + " is Not
Prime.");
        }
    }
}
```

Main Class

This is the main class which contains the main method which runs and uses all the classes.

Here dog object of the Dog class is created which passes “Black”, “Rock” and “Mammal” as arguments. Then dog.getColor() is used to get the color of the dog object.

Then scanner class is used to take input from the user and stored in String word. This word string is passed in the object palindrome of Palindrome class. Then it is checked if it is palindrome or not by the use of palindrome.printResult() method.

Then similarly the scanner class is used to take input from user and stored in int number. The value is passed to the object prime of Prime class as argument. Then it is checked if it is a prime or not by using prime.printPrimeOrNot() method.

Then at last scanner.close() is used to close the scanner class.

```
package workshop3;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

        // object of the Dog class which extends Animal class
        Dog dog = new Dog("Black", "Rocky", "Mammal");
        dog.getColor();//gets the color of the dog

        // scanner class to take user input
        System.out.println("\nPlease enter a word/number to check
Palindrome: ");
        Scanner scanner = new Scanner(System.in);
        String word = scanner.next();

        // Palindrome words
        Palindrome palindrome = new Palindrome(word);
        palindrome.printResult();

        //check number
        System.out.println("\nPlease enter any number");
        int number = scanner.nextInt();

        Prime prime = new Prime(number);
        prime.printPrimeOrNot();
        scanner.close();
    }
}
```

Class Calculator

This is a Calculator class where there is a instance variable String alphabets which has a private access modifier and value “abcdefghijklmnopqrstuvwxyz”.

The constructor is taken which just prints a starting message.

Then add(), subtract(), multiply(), divide(), alphabetize() are of void return type and take two arguments. The add, and subtract methods take int one and int two arguments. The multiply and divide methods takes double one and double two. The alphabetize method takes String one and String two arguments.

Here the add, subtract, multiply and divide methods perform the operations as their names define.
The alphabetize method prints which words comes were alphabetically.
There in the main method, takes the user input using scanner class and operation too. Then the operation is performed according to the user input using switch case.

```
package Calculator;

import java.util.Scanner;

class Calculator {
    private String alphabets = "abcdefghijklmnopqrstuvwxyz";

    // constructor
    public Calculator() {
        System.out.println("List of operators: add subtract multiply
divide alphabetize");
    }

    // addition method
    void add(int one, int two) {
        int result = one + two;
        System.out.println("Answer: " + result);
    }

    // subtraction method
    void subtract(int one, int two) {
        int result = one - two;
        System.out.println("Answer: " + result);
    }

    // multiplication method
    void multiply(double one, double two) {
        double result = one * two;
        System.out.printf("Answer: %.2f", result);
    }

    // division method
    void divide(double one, double two) {
        double result = one / two;
        System.out.printf("Answer : %.2f", result);
    }

    // check the lexicographical order
    void alphabetize(String one, String two) {

        // strings to lower case
```

```
        one = one.toLowerCase();
        two = two.toLowerCase();

        // position of the first char of the two strings one and two
        int oneIndex1 = alphabets.indexOf(one.charAt(0));
        int twoIndex1 = alphabets.indexOf(two.charAt(0));
        int result = oneIndex1 - twoIndex1;

        // if both the first chars are same
        if (result == 0) {
            int oneIndex2 = alphabets.indexOf(one.charAt(1));
            int twoIndex2 = alphabets.indexOf(two.charAt(1));
            int result2 = oneIndex2 - twoIndex2;
            // if both the second chars are same
            if (result2 == 0) {
                System.out.println("Answer: Chicken or Egg");
            } else if (result2 > 0) {
                System.out.println("Answer: " + two + " comes before " +
one + " alphabetically.");
            } else {
                System.out.println("Answer: " + one + " comes before " +
two + " alphabetically.");
            }
        } else if (result > 0) {
            System.out.println("Answer: " + two + " comes before " + one +
" alphabetically.");
        } else {
            System.out.println("Answer: " + one + " comes before " + two +
" alphabetically.");
        }
    }

    public static void main(String[] args) {
        // taking user input
        try (Scanner scanner = new Scanner(System.in)) {

            // object of calculator class
            Calculator calculator = new Calculator();

            // operation to perform
            System.out.println("Enter an operation: ");
            String operator = scanner.nextLine();

            switch (operator) {
                case "add":
                    System.out.println("Enter two integers: ");
                    try {
                        int oneAdd = scanner.nextInt();
                        int twoAdd = scanner.nextInt();
                        calculator.add(oneAdd, twoAdd);
                    } catch (Exception e) {
                        System.out.println("Invalid Input Entered.
Terminating...");
                    }
```

```
                    break;
                case "subtract":
                    System.out.println("Enter two integers: ");
                    try {
                        int oneSub = scanner.nextInt();
                        int twoSub = scanner.nextInt();
                        calculator.subtract(oneSub, twoSub);
                    } catch (Exception e) {
                        System.out.println("Invalid Input Entered.
Terminating...");
                    }
                    break;
                case "multiply":
                    System.out.println("Enter two doubles: ");
                    try {
                        double oneMul = scanner.nextDouble();
                        double twoMul = scanner.nextDouble();
                        calculator.multiply(oneMul, twoMul);
                    } catch (Exception e) {
                        System.out.println("Invalid Input Entered.
Terminating...");
                    }
                    break;
                case "divide":
                    System.out.println("Enter two doubles: ");
                    try {
                        double oneDiv = scanner.nextDouble();
                        double twoDiv = scanner.nextDouble();
                        calculator.divide(oneDiv, twoDiv);
                    } catch (Exception e) {
                        System.out.println("Invalid Input Entered.
Terminating...");
                    }
                    break;
                case "alphabetize":
                    System.out.println("Enter two words: ");
                    try {
                        String oneStr = scanner.next();
                        String twoStr = scanner.next();
                        calculator.alphabetize(oneStr, twoStr);
                    } catch (Exception e) {
                        System.out.println("Invalid Input Entered.
Terminating...");
                    }
                    break;
                default:
                    System.out.println("Invalid input entered.
Terminating...");
            }
        }

    }
}
```

Class SimpleInterest

This SimpleInterest has four instance variables int principle, int time, double rate, double simpleInterest which has private access modifiers.

Then the constructor is used to print welcome message and assign values to the instance variables principle, time and rate.

Then there is getSimpleInterest() method of double return type which returns the interest.

The amountToGet() function returns the total amount (sum of principle and interest).

The main method has one interest object of SimpleInterest class which takes all the required arguments. Then getSimpleInterest() method and amountToGet() method to get interest and total amount.

```
package Interest;

public class SimpleInterest {
    private int principle, time;
    private double rate, simpleInterest;

    public SimpleInterest(int principle, int time, double rate) {
        System.out.println("////// Welcome to the Simple Interest
Calculator ///////");
        this.principle = principle;
        this.time = time;
        this.rate = rate;
    }

    public double getSimpleInterest() {
        simpleInterest = (this.principle * this.time * this.rate) / 100;
        return simpleInterest;
    }

    public double amountToGet(){
        return simpleInterest + this.principle;
    }

    public static void main(String[] args) {
        SimpleInterest interest = new SimpleInterest(100000, 5, 2.937034);
        System.out.println("The simple interest is : \n" +
interest.getSimpleInterest());
        System.out.println("The Total amount to recieve is : \n" +
interest.amountToGet());
    }
}
```

Class MultiplicationTable

This MultiplicationTable class has instance variables int multi and int upto which have private

access modifiers.
Then the constructor takes and puts values in the instance variables.
Then there are getter and setter methods for both instance variables.
There is a getMutiplicationTable() method which prints the multiplication table. All the inputs are taken from the main method using scanner class and passed to the object of the MultiplicationTable class.

```
package MultiplicationTable;

import java.util.Scanner;

public class MultiplicationTable {

    // variables declaration
    private int multi, upto;

    // assigning the values to the instance variables
    public MultiplicationTable(int multi, int upto) {
        this.multi = multi;
        this.upto = upto;
    }

    //getter methods
    public int getMulti() {
        return this.multi;
    }

    public int getUpto() {
        return this.upto;
    }


    //setter methods
    public void setMulti(int multi) {
        this.multi = multi;
    }
    public void setUpto(int upto) {
        this.upto = upto;
    }

    public void getMultiplicationTable() {
        // looping the multiplication statement for the number of range
provided
        for (int i = 1; i <= this.upto; i++) {
            System.out.println(multi + " x " + i + " = " + multi * i);
        }
        System.out.println("#Thank You  :)");
    }

    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            // taking the first number to get the multiplication of
            System.out.println("Please enter the number: ");
            int multi = scanner.nextInt();
```

```
            // the range of the multiplication table
            System.out.println("Please enter the range: ");
            int upto = scanner.nextInt();

            // Creating the multiplication table object of multiplication
table class
            MultiplicationTable multiplicationTable = new
MultiplicationTable(multi, upto);

            // getting the multiplication table
            multiplicationTable.getMultiplicationTable();
        }
    }

}
```

Class Car

This is a Car class where there are instance variables String brand, String color, int mileage, int year, and Boolean hasFourWheels which has private access modifier.

The constructor takes the all the values and assigns it to the instance variables.

There is a checkYear() which prints if the vehicle should run or not according to the year it is made. If it is older than 2010 then it should not run else it can run.

There is a printBrand() method which prints the brand of the car.

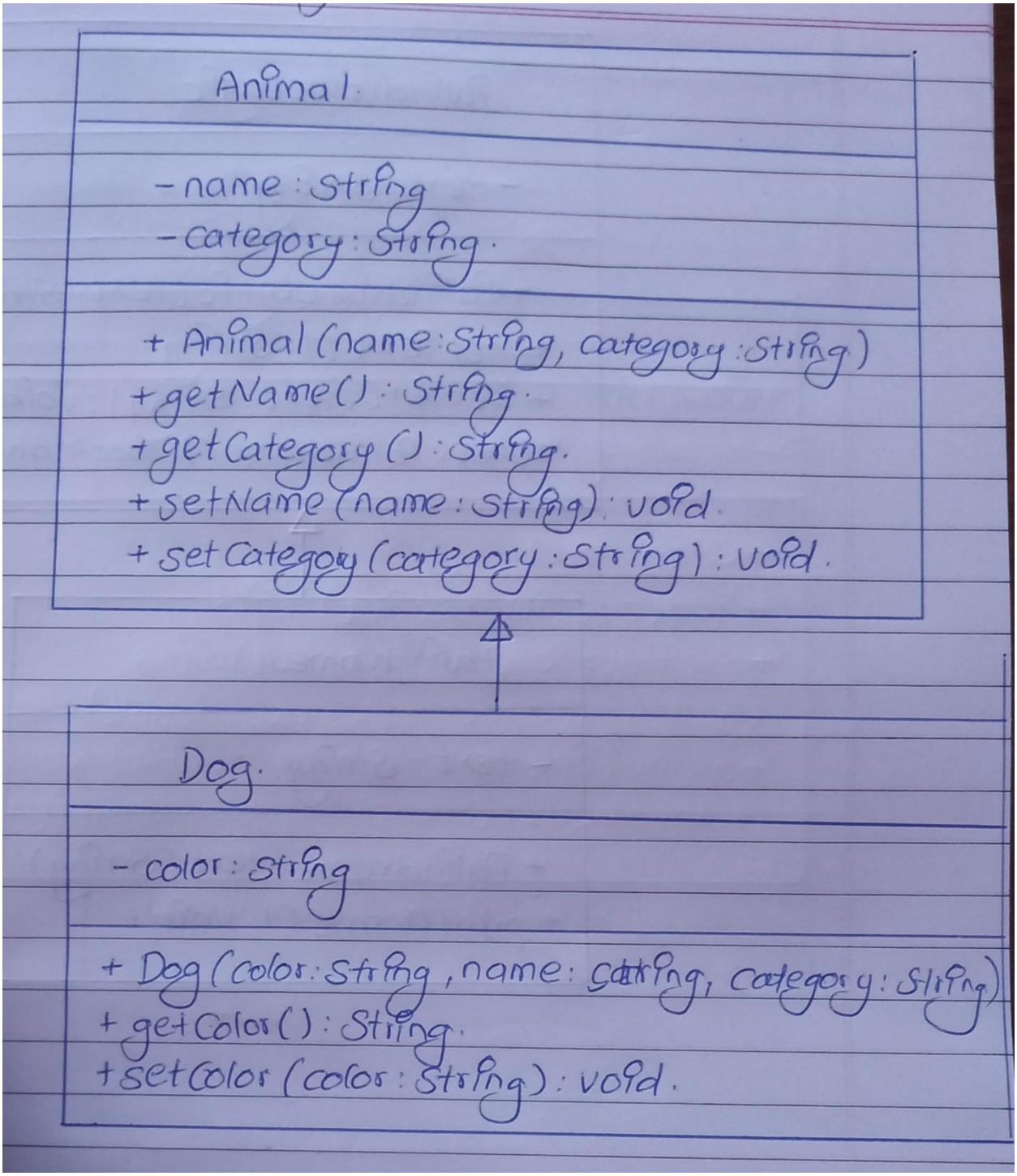
There are two methods increaseMileage() and decreaseMileage() which increases and decreases the mileage of the vehicle by one.

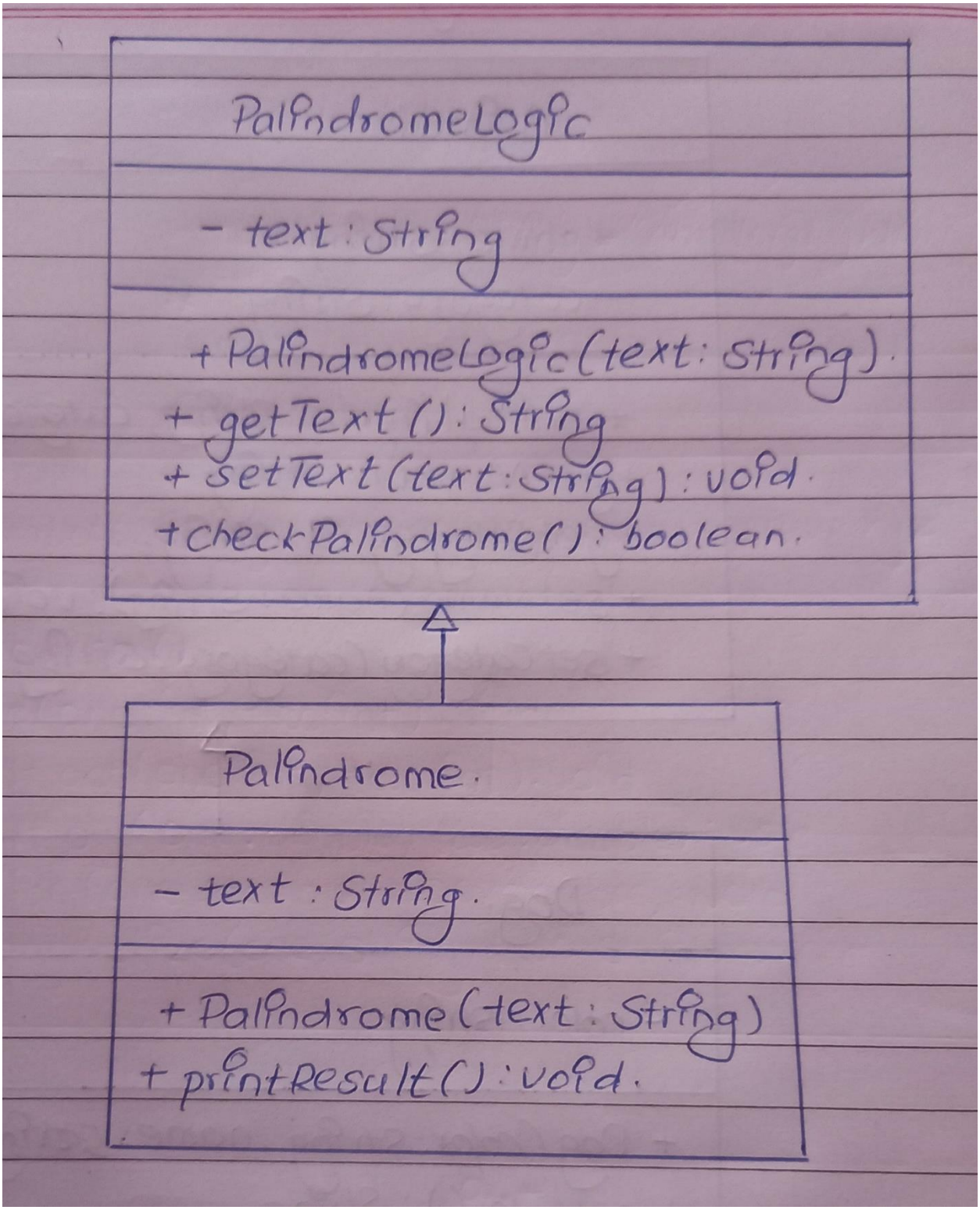
There is another method printAll() which prints all the values of the instance variables.
In he main method car object is created of Car class with arguments passed in it.

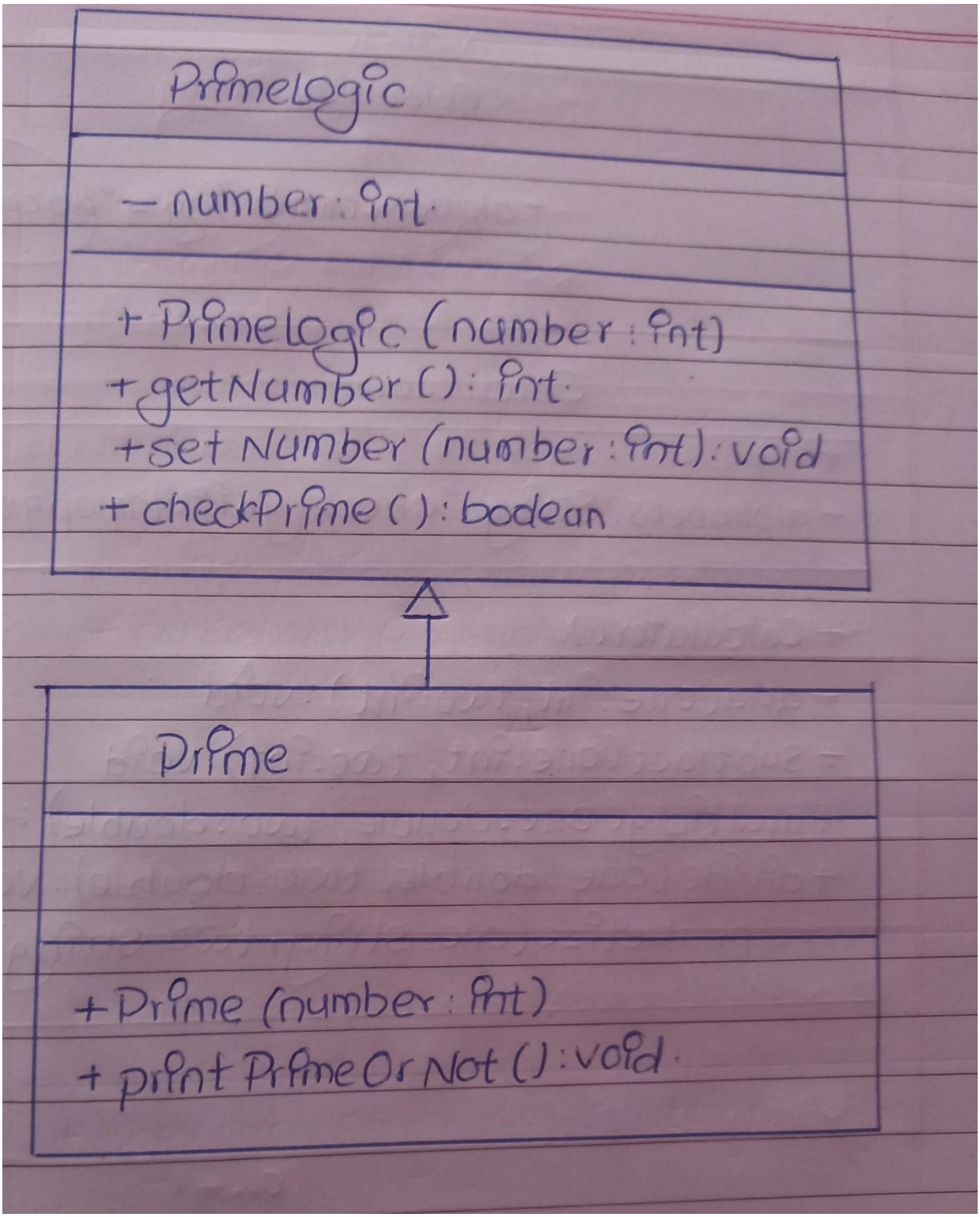
```
package Vehicle;

public class Car {
    private String brand;
    private String color;
    private int mileage;
    private int year;
    private boolean hasFourWheels;

    // set the values
    public Car(String brand, String color, int mileage, boolean
hasFourWheels,int year) {

        //separator
        for (int i = 0; i < 3; i++) {
```

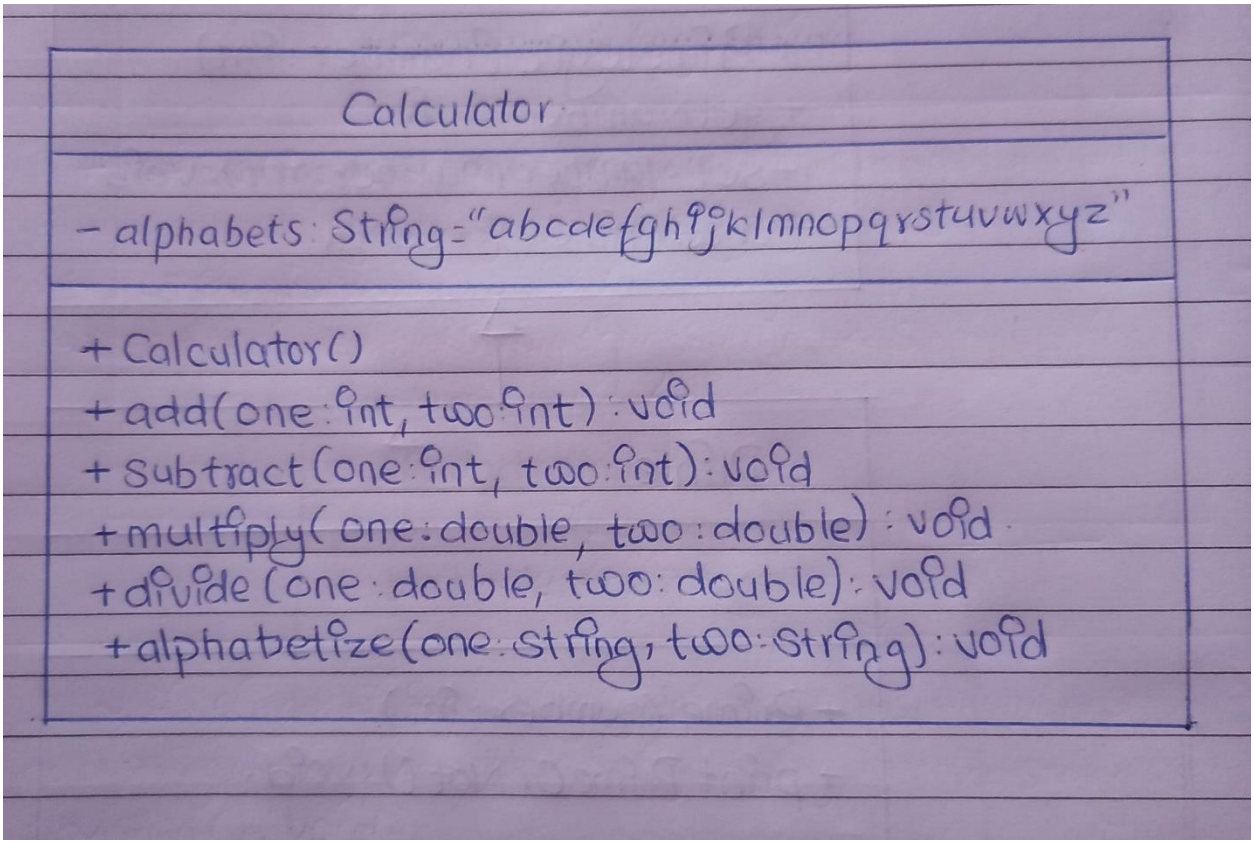
```
System.out.println("////////////////////////////////////////");
        }
        this.brand = brand;
        this.color = color;
        this.mileage = mileage;
        this.hasFourWheels = hasFourWheels;
        this.year = year;
    }
    public void checkYear(){
        if(this.year <= 2010){
            System.out.println("the vehicle should not run");
        }else{
            System.out.println("The vehicle can run");
        }
    }

    // print the brand of the car
    public void printBrand() {
        System.out.println(this.brand);
    }

    public void increaseMileage() {
        this.mileage++;
    }

    public void decreaseMileage() {
        this.mileage--;
    }

    public void printAll() {
        System.out.println(this.brand);
        System.out.println(this.color);
        System.out.println(this.mileage);
        System.out.println(this.hasFourWheels);
    }

    public static void main(String[] args) {
        Car car = new Car("toyota", "red", 40, false,1998);
        car.checkYear();
    }
}
```

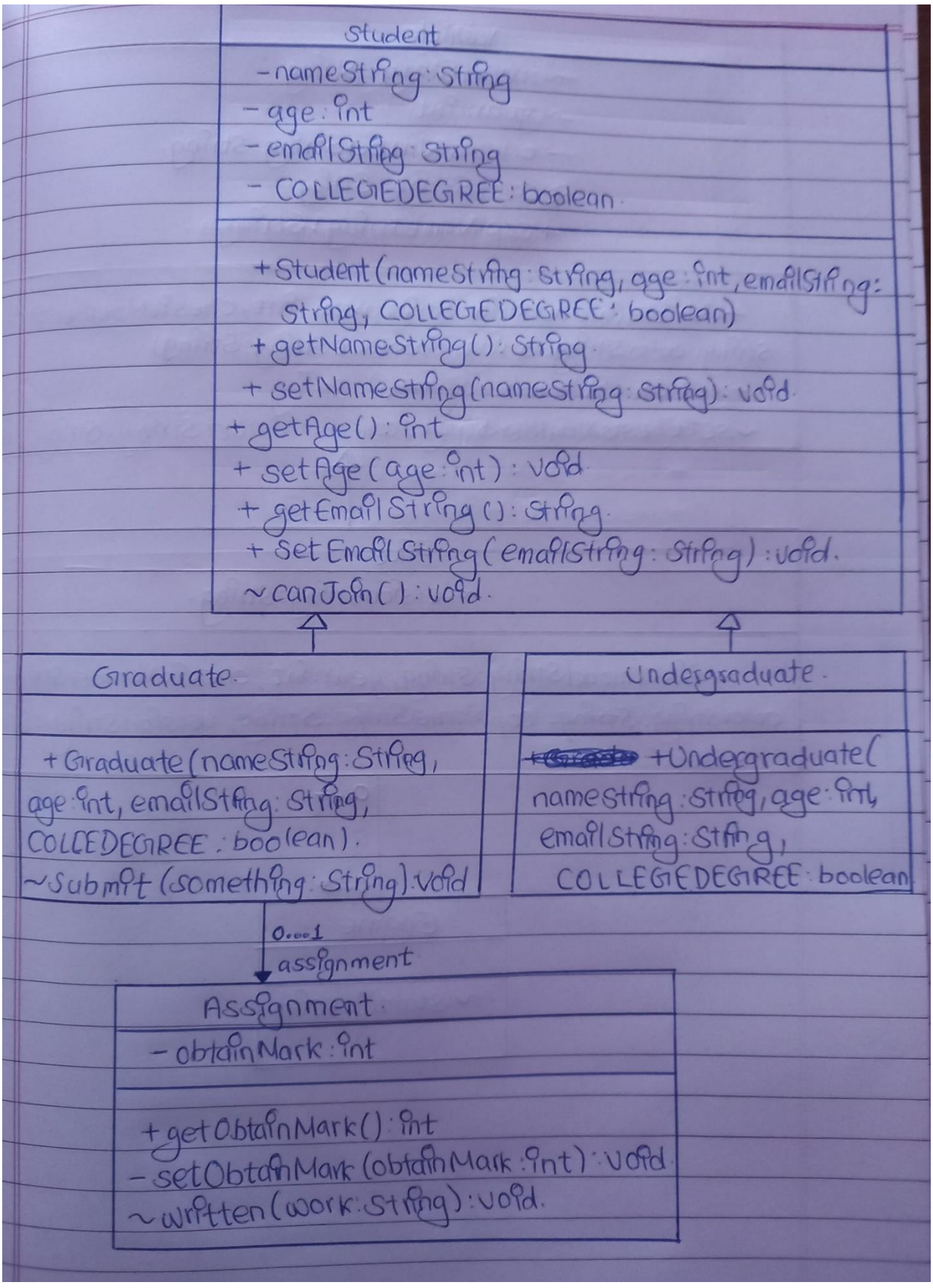
Class Diagrams

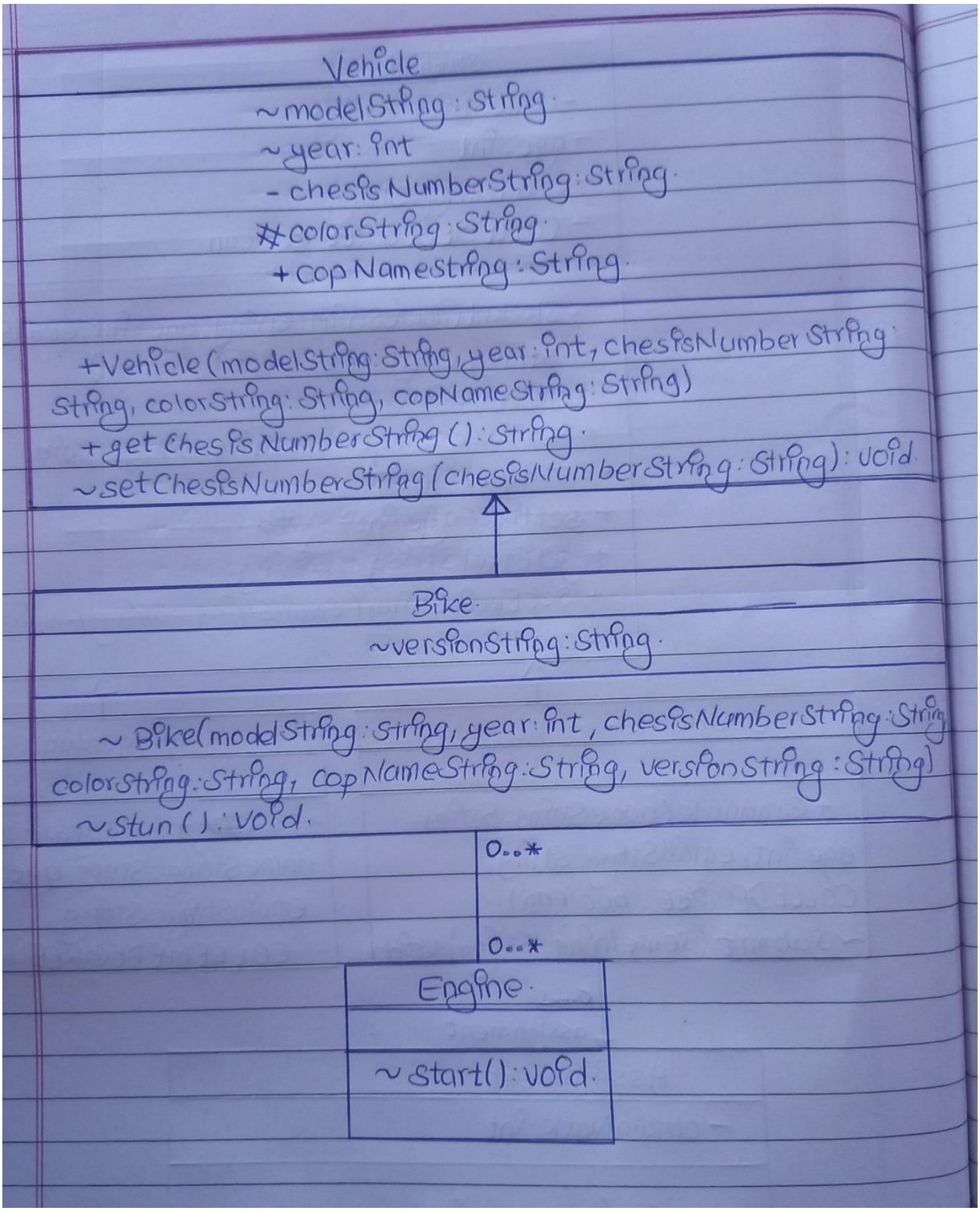

Animal
-name : String
-category : String
+ Animal (name : String, category : String)
+getName() : String
+getCategory() : String
+setName(name : String) : void
+setCategory(category : String) : void
Dog
- color : String
+ Dog(color : String, name : String, category : String)
+getColor() : String
+setColor(color : String) : void

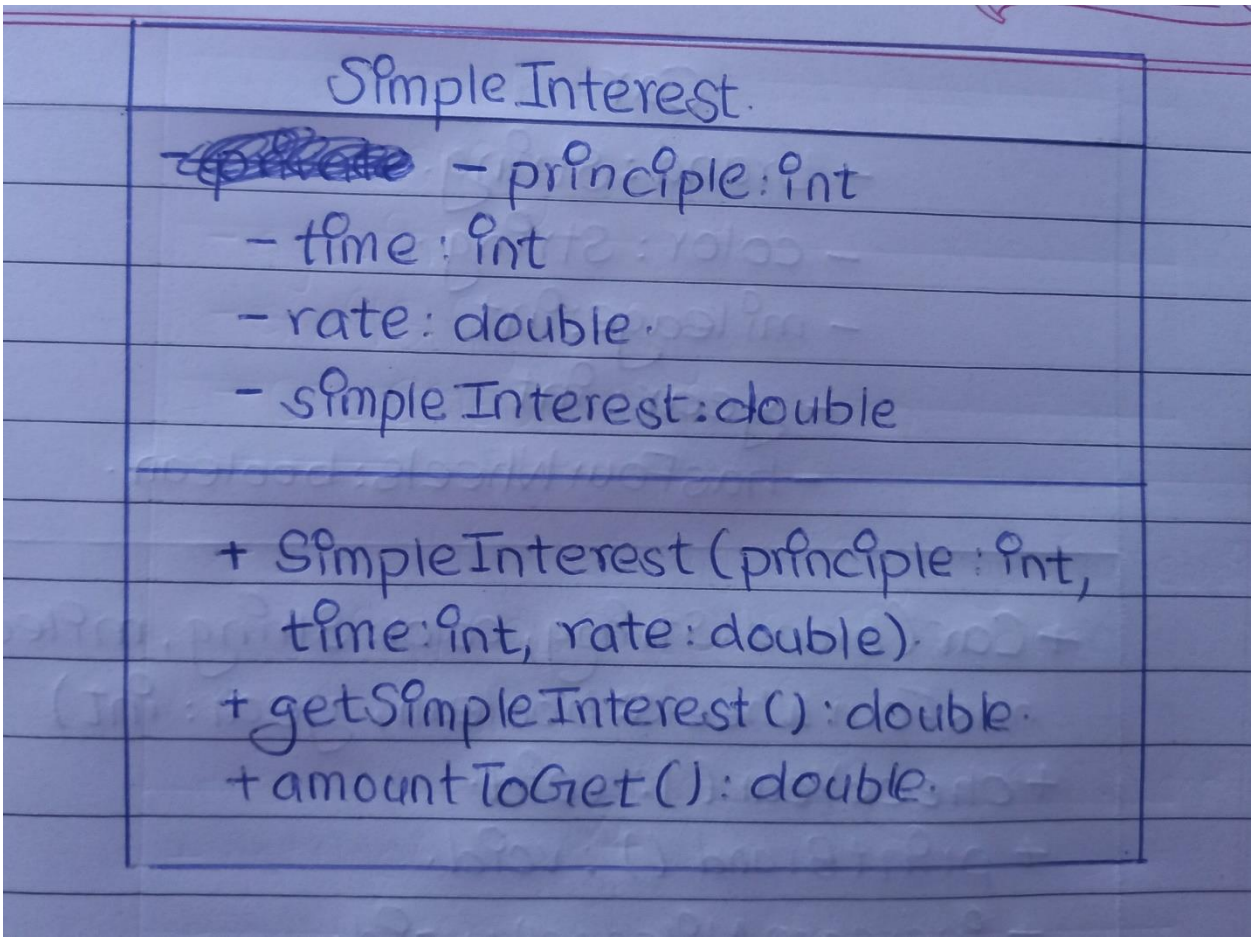

PalindromeLogic
- text : String
+ PalindromeLogic(text: String).
+ getText(): String
+ setText(text: String): void.
+ checkPalindrome(): boolean.
Palindrome.
- text : String.
+ Palindrome(text : String)
+ printResult(): void.

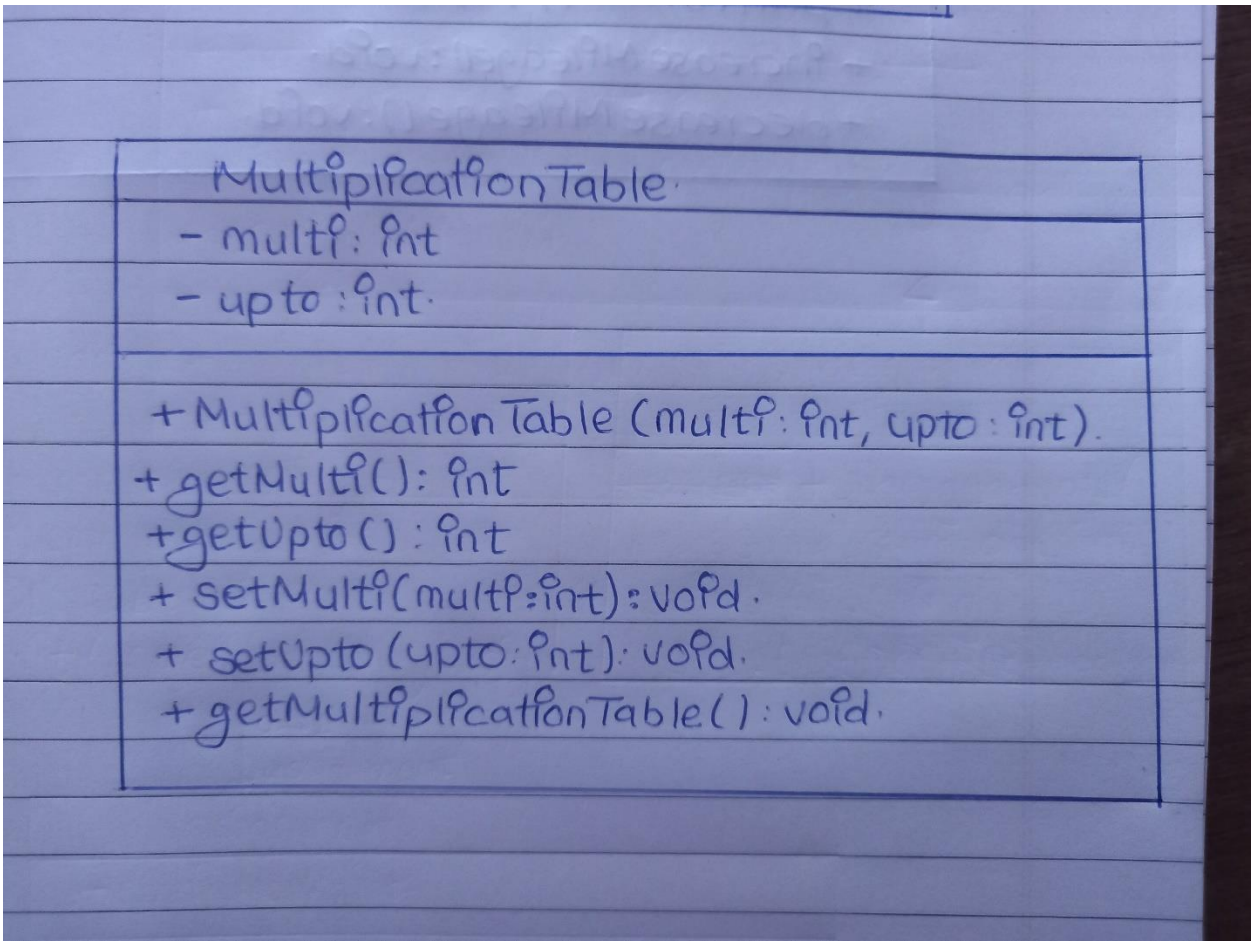

PrimeLogic
- number : int
+ PrimeLogic (number : int)
+ getNumber () : int
+ set Number (number : int) : void
+ checkPrime () : bodean
Prime
+ Prime (number : int)
+ print Prime Or Not () : void

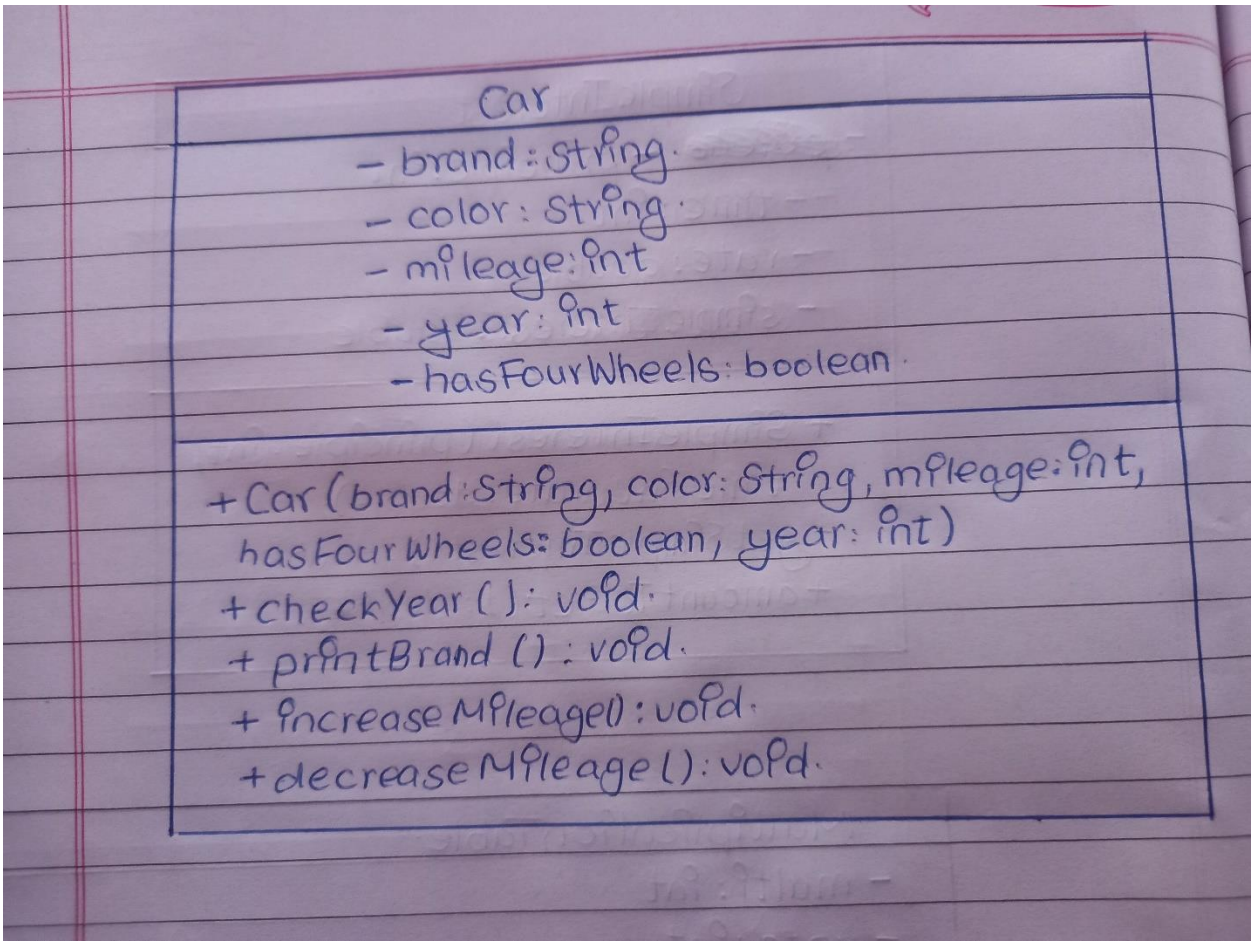

Calculator
- alphabets: String="abcdefghijklmnopqrstuvwxyz"
+ Calculator()
+ add(one: int, two: int): void
+ subtract(one: int, two: int): void
+ multiply(one: double, two: double): void
+ divide(one: double, two: double): void
+ alphabetize(one: String, two: String): void


Student
-nameString: String
-age: int
-emailString: String
-COLLEGEDEGREE: boolean
+Student(nameString: String, age: int, emailString: String, COLLEGEDEGREE: boolean)
+getNameString(): String
+setNamestring(namestring: String): void
+getAge(): int
+setAge(age: int): void
+getEmailString(): String
+SetEmailString(emailString: String): void
~canJoin(): void
Graduate
+Graduate(nameString: String, age: int, emailString: String, COLLEGEDEGREE: boolean)
~Submit(something: String): void
Undergraduate
+Undergraduate(namestring: String, age: int, emailString: String, COLLEGEDEGREE: boolean)
0...1
assignment
Assignment
-obtainMark: int
+getObtainMark(): int
-setObtainMark(obtainMark: int): void
~written(work: String): void


Vehicle
~modelString : String
~year: int
-chesisNumberString: String
#colorString: String
+copNamestring: String
+Vehicle(modelString: String, year: int, chesisNumber String: String, colorString: String, copNameString: String)
+getChesisNumberString(): String
~setChesisNumberString(chesisNumberString: String): void
Bike
~versionString: String
~Bike(modelString: String, year: int, chesisNumberString: String, colorString: String, copNameString: String, versionString: String)
~Stun(): void
0..*
0..*
Engine
~start(): void


SimpleInterest
- principle: int
- time: int
- rate: double
- simpleInterest: double
+ SimpleInterest(principle: int, time: int, rate: double)
+ getSimpleInterest(): double
+ amountToGet(): double


MultiplicationTable
- multi: int
- upto : int.
+MultiplicationTable (multi: int, upto : int).
+getMulti(): int
+getUpto() : int
+ setMulti(multi:int): void.
+ setUpto (upto: int): void.
+getMultiplicationTable() : void.


Car
- brand : String
- color : String
- mileage : int
- year : int
- hasFourWheels : boolean
+ Car (brand : String, color : String, mileage : int, hasFourWheels : boolean, year : int)
+ checkYear ( ) : void
+ printBrand ( ) : void
+ increaseMileage() : void
+ decreaseMileage ( ) : void