CIS 36B – 4th Class / Lab Assignment – **10 Points-**
Student Name kachilau Student ID 10819338 Point Total

Section 1: Define / Answer

“programmer created” class- A class is the blueprint from which individual objects are created.

“programmer created” object- A typical Java program creates many objects, which as you know, interact by invoking methods. Through these object interactions, a program can carry out various tasks, such as implementing a GUI, running an animation, or sending and receiving information over a network. Once an object has completed the work for which it was created, its resources are recycled for use by other objects.

“programmer created” method- collection of statements that are grouped together to perform an operation; use to access the static field

Describe the piece of code below is doing:

Vehicle minivan = **new** Vehicle();

creating a vehicle object

void method-

“programmer created” class- How do “programmer created methods” using **void**, differ from using created using **return**? Briefly - Define how return works.

https://docs.oracle.com/javase/tutorial/essential/io/formatting.html

CIS 36B – 4th Class / Lab Assignment – **10 Points-**
Student Name kachilau Student ID 10819338 Point Total

Format Specifier- the sequence passed as the formatting string argument; "Characters matched" gives the format of the sequence sought or printed, with a hyperlink to the section on literals which applies to that format;

" %.2f **"-** Describe what the statement is saying- Print 2 decimal

"%n"- Describe what the statement is saying- new line

System.out.format("%.2f%n", b + a); // Example of code

System.out.format(…); Explain how this method differs from System.out.println

Println couldn't do the %.2f but format can

Pg. 577, Java Programming *A comprehensive Introduction*

http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format(Detailed explanation of Java documentation)

http://www.tutorialspoint.com/java/java_documentation.htm

http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments

Javadoc tags (Examples)

Tag	**Description**	**Syntax**
@author	Adds the author of a class.	@author name-text

CIS 36B – 4th Class / Lab Assignment – **10 Points-**
Student Name kachilau **Student ID 10819338** **Point Total**

{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecated-text
@exception	Adds a **Throws** subheading to the generated documentation, with the class-name and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with visible text label that points to the documentation for the specified package, class or member name of a referenced class. T	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description

CIS 36B – 4th Class / Lab Assignment – **10 Points-**
Student Name kachilau Student ID 10819338 Point Total

{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant:	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

Pg. 123, Java Programming *A comprehensive Introduction*

Programming Assignment

Task 1- Create a computer program that will calculate the range for 3 different vehicles.

The program should create a "programmer created" class, where 3 **int objects** are created passengers, fuel capacity, mpg.

Create a **void()** method inside the "programmer created " class to calculate vehicle range.

Create a second void() that averages the number of passengers, the fuel capacity, and mpg among the 3 vehicle objects.

range = fuel capacity * miles per gallon.

Each Vehicle type should have unique values for number of passengers, fuel capacity, and miles per gallon.

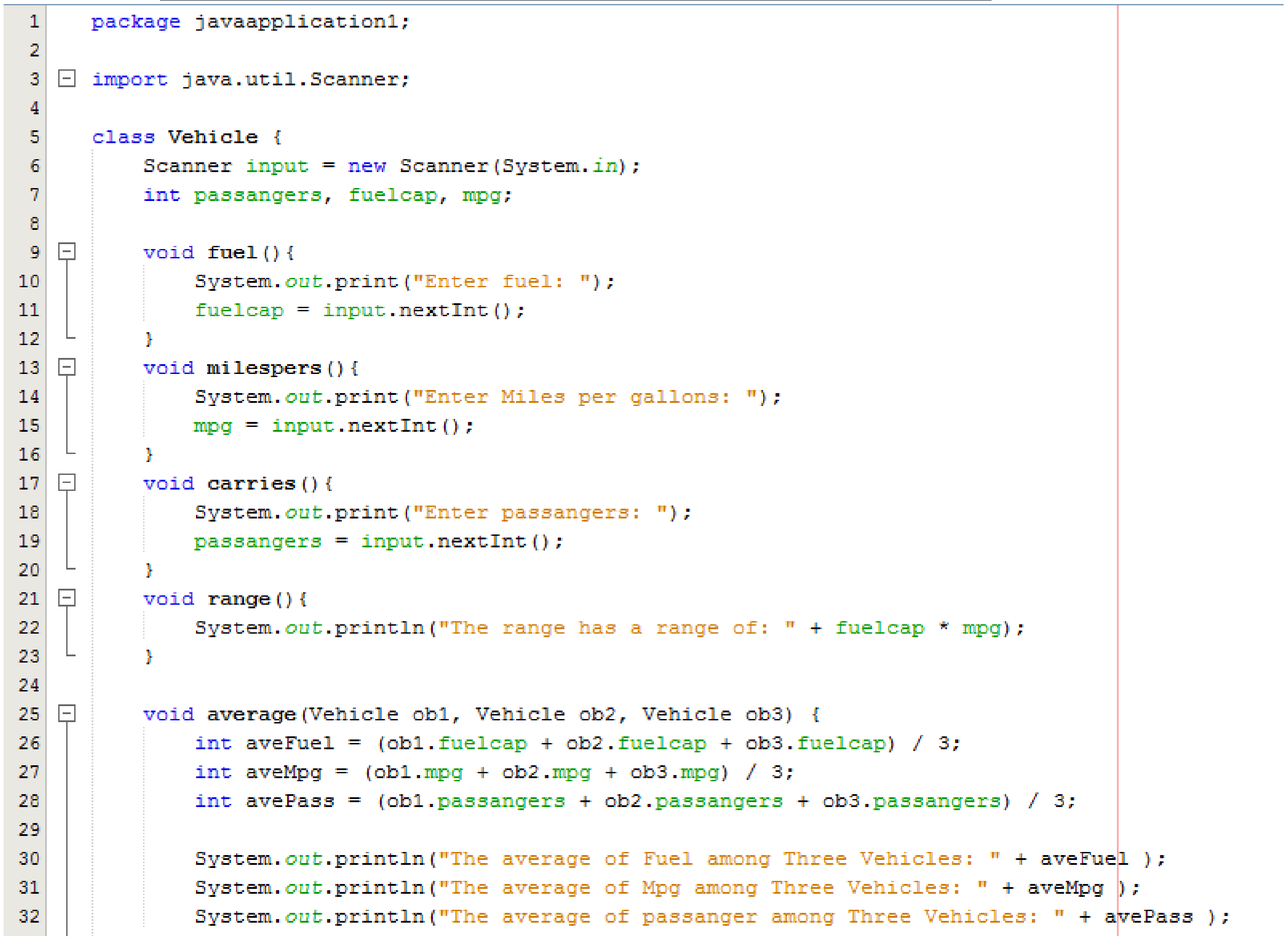
Follow the sample below and return information on 3 vehicle types.

<u>Sample Output: // Create similar output for 3 Vehicle Types</u>

```
The minivan carries= 7

The minivan has a fuel capacity of = 16

The minivan mpg = 21

The minivan has a range of: 336 miles
```

CIS 36B – 4th Class / Lab Assignment – **10 Points-**

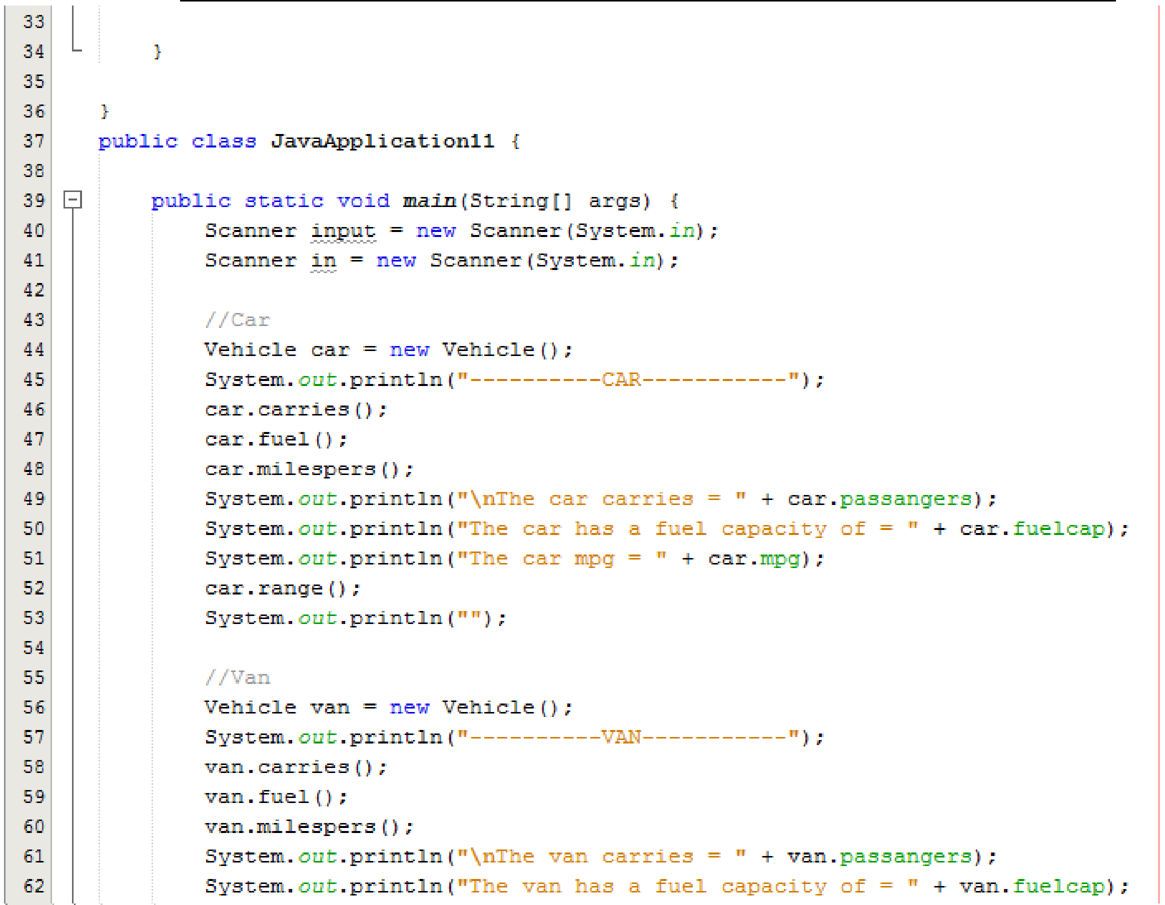
Student Name kachilau **Student ID 10819338** **Point Total**



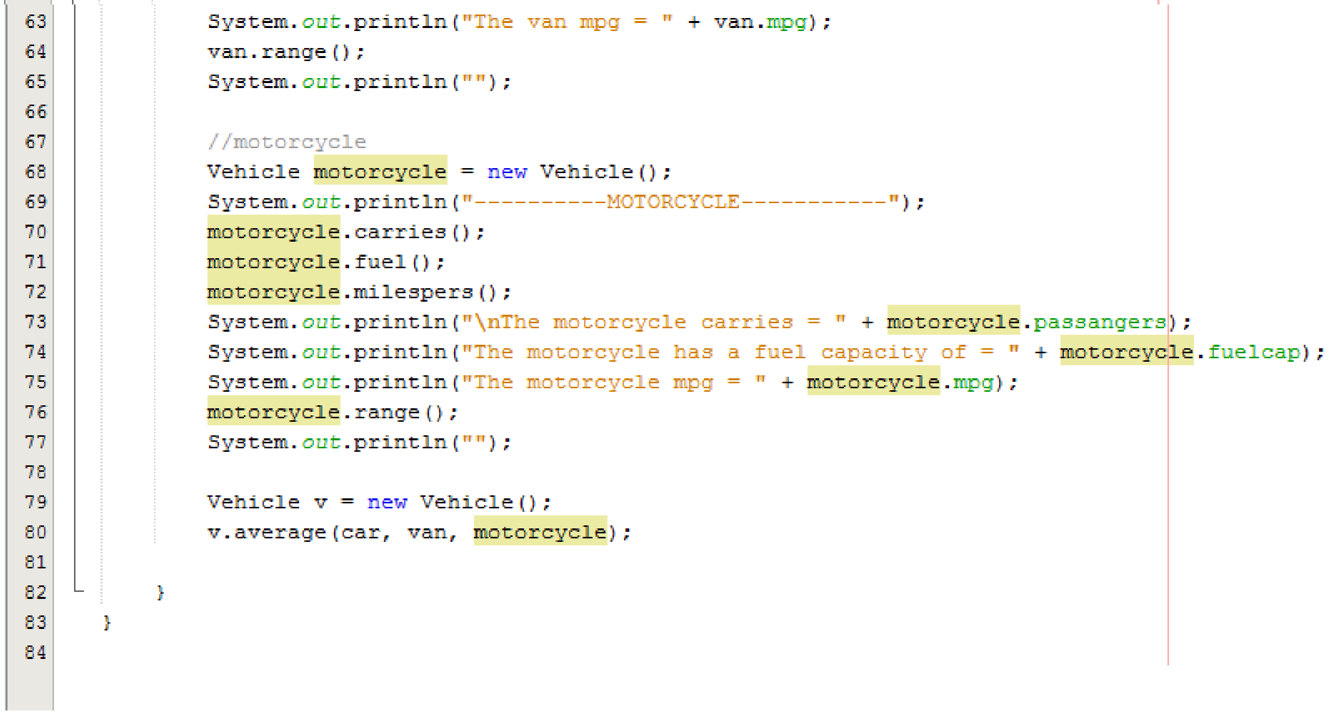
```java
package javaapplication1;

import java.util.Scanner;

class Vehicle {
    Scanner input = new Scanner(System.in);
    int passangers, fuelcap, mpg;

    void fuel(){
        System.out.print("Enter fuel: ");
        fuelcap = input.nextInt();
    }
    void milespers(){
        System.out.print("Enter Miles per gallons: ");
        mpg = input.nextInt();
    }
    void carries(){
        System.out.print("Enter passangers: ");
        passangers = input.nextInt();
    }
    void range(){
        System.out.println("The range has a range of: " + fuelcap * mpg);
    }

    void average(Vehicle ob1, Vehicle ob2, Vehicle ob3) {
        int aveFuel = (ob1.fuelcap + ob2.fuelcap + ob3.fuelcap) / 3;
        int aveMpg = (ob1.mpg + ob2.mpg + ob3.mpg) / 3;
        int avePass = (ob1.passangers + ob2.passangers + ob3.passangers) / 3;

        System.out.println("The average of Fuel among Three Vehicles: " + aveFuel );
        System.out.println("The average of Mpg among Three Vehicles: " + aveMpg );
        System.out.println("The average of passanger among Three Vehicles: " + avePass );
```

CIS 36B – 4th Class / Lab Assignment – **10 Points-**

Student Name **kachilau** **Student ID 10819338** **Point Total**

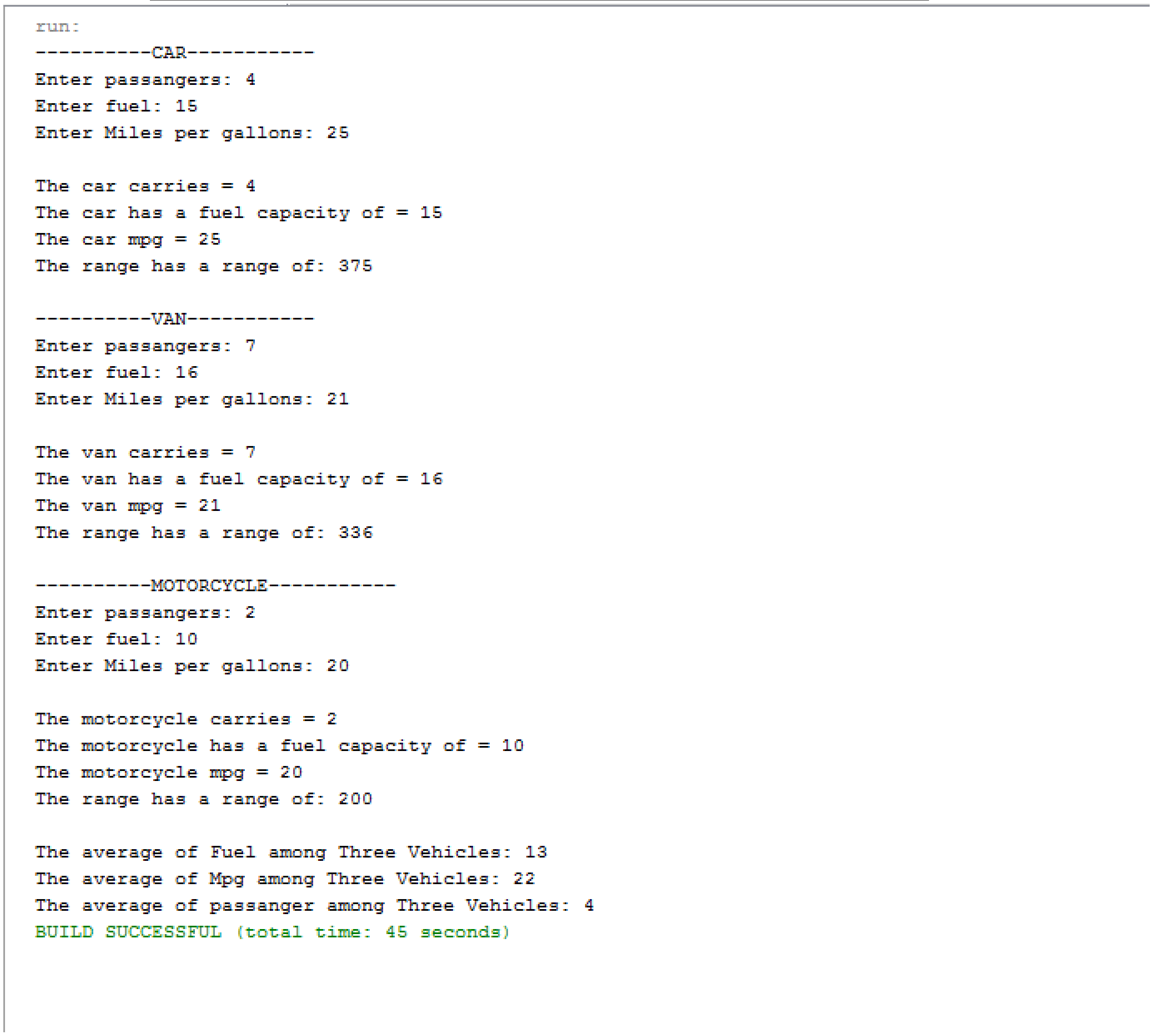


```
    }

}
public class JavaApplication11 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Scanner in = new Scanner(System.in);

        //Car
        Vehicle car = new Vehicle();
        System.out.println("----------CAR-----------");
        car.carries();
        car.fuel();
        car.milespers();
        System.out.println("\nThe car carries = " + car.passangers);
        System.out.println("The car has a fuel capacity of = " + car.fuelcap);
        System.out.println("The car mpg = " + car.mpg);
        car.range();
        System.out.println("");

        //Van
        Vehicle van = new Vehicle();
        System.out.println("----------VAN-----------");
        van.carries();
        van.fuel();
        van.milespers();
        System.out.println("\nThe van carries = " + van.passangers);
        System.out.println("The van has a fuel capacity of = " + van.fuelcap);
```



```
        System.out.println("The van mpg = " + van.mpg);
        van.range();
        System.out.println("");

        //motorcycle
        Vehicle motorcycle = new Vehicle();
        System.out.println("----------MOTORCYCLE-----------");
        motorcycle.carries();
        motorcycle.fuel();
        motorcycle.milespers();
        System.out.println("\nThe motorcycle carries = " + motorcycle.passangers);
        System.out.println("The motorcycle has a fuel capacity of = " + motorcycle.fuelcap);
        System.out.println("The motorcycle mpg = " + motorcycle.mpg);
        motorcycle.range();
        System.out.println("");

        Vehicle v = new Vehicle();
        v.average(car, van, motorcycle);

    }
}
```

CIS 36B – 4th Class / Lab Assignment – **10 Points-**
Student Name __**kachilau**__ **Student ID 10819338** **Point Total**



```
run:
----------CAR-----------
Enter passangers: 4
Enter fuel: 15
Enter Miles per gallons: 25

The car carries = 4
The car has a fuel capacity of = 15
The car mpg = 25
The range has a range of: 375

----------VAN-----------
Enter passangers: 7
Enter fuel: 16
Enter Miles per gallons: 21

The van carries = 7
The van has a fuel capacity of = 16
The van mpg = 21
The range has a range of: 336

----------MOTORCYCLE-----------
Enter passangers: 2
Enter fuel: 10
Enter Miles per gallons: 20

The motorcycle carries = 2
The motorcycle has a fuel capacity of = 10
The motorcycle mpg = 20
The range has a range of: 200

The average of Fuel among Three Vehicles: 13
The average of Mpg among Three Vehicles: 22
The average of passanger among Three Vehicles: 4
BUILD SUCCESSFUL (total time: 45 seconds)
```

<u>Task 2:</u> Write a program that computes a single filer's income tax burden.

TAX RATE	**Single Filers Income**
10%	Up to $6000
15%	$6,001 - $27,950

27%	$27,951 - $67,700
30%	$67,701 - $141,250
35%	$141,251 - $307,050
38.6%	$307, 051 or more

The user should be able input her income using **new** Scanner (System.in), input method and then be returned the amount of tax owed.

All source code for solving the problem and handling user input should be created in a "programmer created class."

Use **return** for retrieving all values from calculations or **if/else** statements etc.

main will be used to operate the program.

Output should have proper formatting for dollars, 2 decimal places.

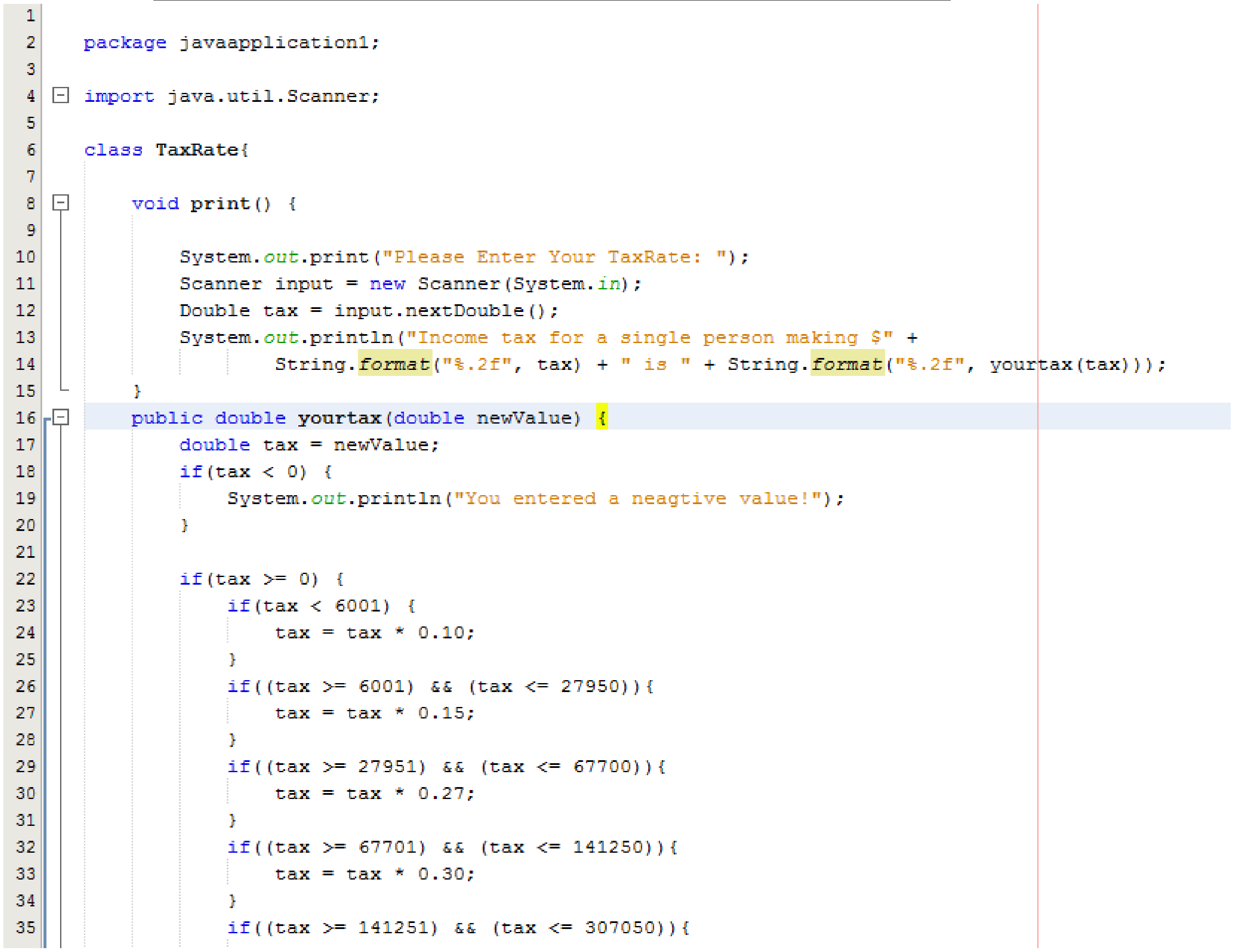
Sample Output- //Output should have proper formatting for dollars, 2 decimal places

```
Income tax for a single person making $85000.00 is $25500.00
```

```
Income tax for a single person making $9800.00 is $1470.00
```

CIS 36B – 4th Class / Lab Assignment – **10 Points-**

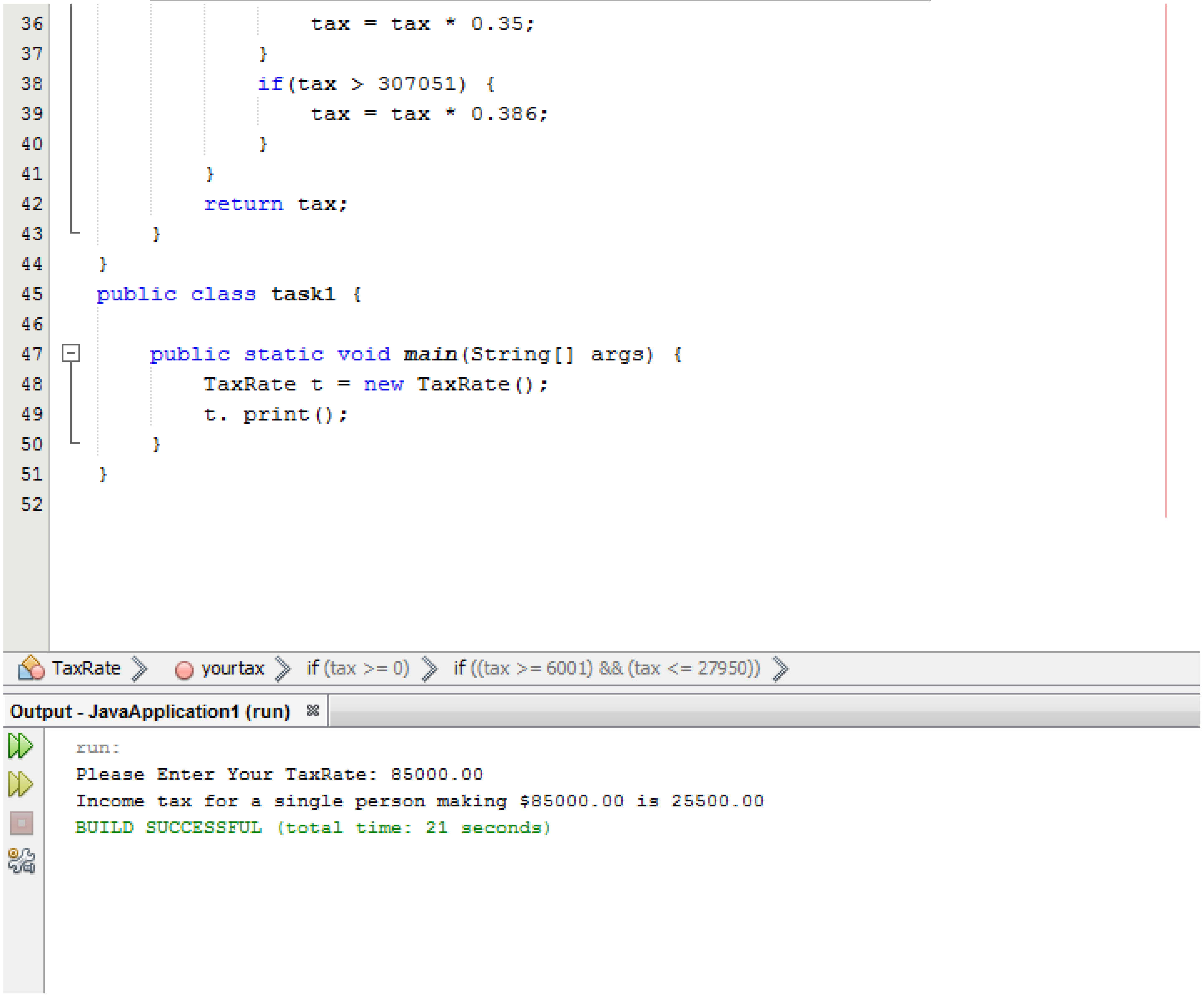
Student Name **kachilau** **Student ID 10819338** **Point Total**



```java

package javaapplication1;

import java.util.Scanner;

class TaxRate{

    void print() {

        System.out.print("Please Enter Your TaxRate: ");
        Scanner input = new Scanner(System.in);
        Double tax = input.nextDouble();
        System.out.println("Income tax for a single person making $" +
                String.format("%.2f", tax) + " is " + String.format("%.2f", yourtax(tax)));
    }
    public double yourtax(double newValue) {
        double tax = newValue;
        if(tax < 0) {
            System.out.println("You entered a neagtive value!");
        }

        if(tax >= 0) {
            if(tax < 6001) {
                tax = tax * 0.10;
            }
            if((tax >= 6001) && (tax <= 27950)){
                tax = tax * 0.15;
            }
            if((tax >= 27951) && (tax <= 67700)){
                tax = tax * 0.27;
            }
            if((tax >= 67701) && (tax <= 141250)){
                tax = tax * 0.30;
            }
            if((tax >= 141251) && (tax <= 307050)){
```

CIS 36B – 4th Class / Lab Assignment – **10 Points-**

Student Name kachilau Student ID 10819338 Point Total



```
                    tax = tax * 0.35;
                }
                if(tax > 307051) {
                    tax = tax * 0.386;
                }
            }
            return tax;
        }
    }
    public class task1 {

        public static void main(String[] args) {
            TaxRate t = new TaxRate();
            t. print();
        }
    }
```

TaxRate » yourtax » if (tax >= 0) » if ((tax >= 6001) && (tax <= 27950)) »

Output - JavaApplication1 (run)

```
run:
Please Enter Your TaxRate: 85000.00
Income tax for a single person making $85000.00 is 25500.00
BUILD SUCCESSFUL (total time: 21 seconds)
```