IT602: Object-Oriented Programming

Lab Assignment-1

Sagar Variya

202112114

Question #1:

You are supposed to create a class for vehicles. In the class, you should include at least two private members named no_of_seats and no_of_wheels. You should have methods for getters and setters for the required private members. In the driver code, you have to create two objects of the same class. (i.e., one for Motorcycle and one for Class). Your output should show the descriptions for Car and Motorcycle. Also, output the size occupied by the objects in the memory.

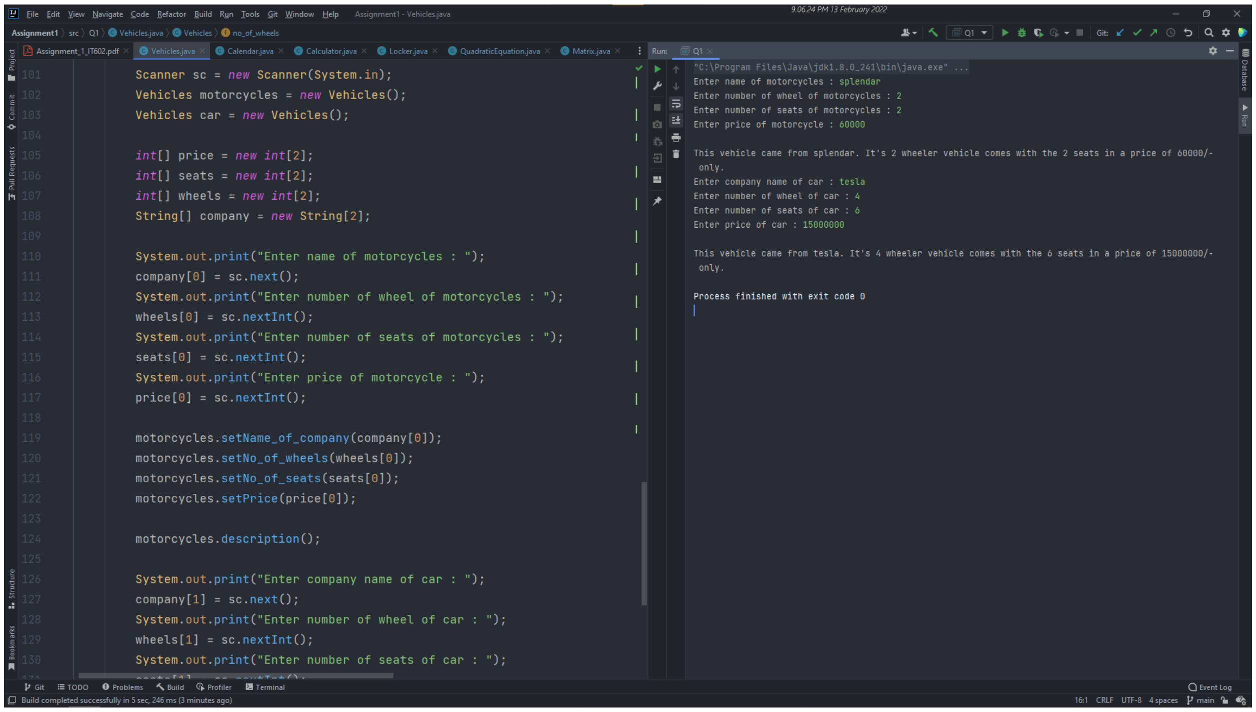
Code #1:

```
/**
 * @author Sagar Variya | 202112114
 */

package Q1;
import java.util.Scanner;

/**
 *  Vehicles class contains 4 data members (with getters & setters) that describes
the price, seats, wheels and the name of the company.
 *  it has member function that can describes the class value in formal way.
 */
public class Vehicles {

    /**
     *  creating 4 private data members
     */
    private int price;
    private int no_of_seats;
    private int no_of_wheels;
    private String name_of_company;

    /**
     * this function gives the price of Vehicle class
     * @return price of vehicle
     */
    public int getPrice() {
       return this.price;
    }

    /**
     * this function sets the price of Vehicle class
     * @param price price
     */
    public void setPrice(int price) {
       this.price = price;
    }

    /**
     * this function gives the number of seats of Vehicle
     * @return number of seats of vehicle
     */
    public int getNo_of_seats() {
       return this.no_of_seats;
    }

    /**
     * this function sets the number of seats of Vehicle
     * @param no_of_seats number of seats
     */
    public void setNo_of_seats(int no_of_seats) {
       this.no_of_seats = no_of_seats;
    }
```

```
/**
 * this function gives the number of wheels of Vehicle
 * @return number of wheels of vehicle
 */
public int getNo_of_wheels() {
   return this.no_of_wheels;
}

/**
 * this function sets the number of wheels of Vehicle
 * @param no_of_wheels number of wheels
 */
public void setNo_of_wheels(int no_of_wheels) {
   this.no_of_wheels = no_of_wheels;
}

/**
 * this function gives the name of the company of Vehicle
 * @return name of the company of vehicle
 */
public String getName_of_company() {
   return this.name_of_company;
}

/**
 * this function sets the name of the company of Vehicle
 * @param name_of_company name of the company
 */
public void setName_of_company(String name_of_company) {
   this.name_of_company = name_of_company;
}
```

```
    /**
     * this function gives the description about the vehicle in formal way
     */
    public void description() {
        System.out.println("\nThis vehicle came from " + getName_of_company() + ". "
                + "It's " + getNo_of_wheels() + " wheeler vehicle comes with the "
                + getNo_of_seats() + " seats in a price of " + getPrice() + "/- only.");
    }
}

class Q1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        Vehicles motorcycles = new Vehicles();
        Vehicles car = new Vehicles();

        int[] price = new int[2];
        int[] seats = new int[2];
        int[] wheels = new int[2];
        String[] company = new String[2];

        System.out.print("Enter name of motorcycles : ");
        company[0] = sc.next();
        System.out.print("Enter number of wheel of motorcycles : ");
        wheels[0] = sc.nextInt();
        System.out.print("Enter number of seats of motorcycles : ");
        seats[0] = sc.nextInt();
        System.out.print("Enter price of motorcycle : ");
        price[0] = sc.nextInt();

        motorcycles.setName_of_company(company[0]);
        motorcycles.setNo_of_wheels(wheels[0]);
        motorcycles.setNo_of_seats(seats[0]);
        motorcycles.setPrice(price[0]);

        motorcycles.description();

        System.out.print("Enter company name of car : ");
        company[1] = sc.next();
        System.out.print("Enter number of wheel of car : ");
        wheels[1] = sc.nextInt();
        System.out.print("Enter number of seats of car : ");
        seats[1] = sc.nextInt();
        System.out.print("Enter price of car : ");
        price[1] = sc.nextInt();

        car.setName_of_company(company[1]);
        car.setNo_of_wheels(wheels[1]);
        car.setNo_of_seats(seats[1]);
        car.setPrice(price[1]);

        car.description();
    }
}
```

Output #1:



```java
        Scanner sc = new Scanner(System.in);
        Vehicles motorcycles = new Vehicles();
        Vehicles car = new Vehicles();

        int[] price = new int[2];
        int[] seats = new int[2];
        int[] wheels = new int[2];
        String[] company = new String[2];

        System.out.print("Enter name of motorcycles : ");
        company[0] = sc.next();
        System.out.print("Enter number of wheel of motorcycles : ");
        wheels[0] = sc.nextInt();
        System.out.print("Enter number of seats of motorcycles : ");
        seats[0] = sc.nextInt();
        System.out.print("Enter price of motorcycle : ");
        price[0] = sc.nextInt();

        motorcycles.setName_of_company(company[0]);
        motorcycles.setNo_of_wheels(wheels[0]);
        motorcycles.setNo_of_seats(seats[0]);
        motorcycles.setPrice(price[0]);

        motorcycles.description();

        System.out.print("Enter company name of car : ");
        company[1] = sc.next();
        System.out.print("Enter number of wheel of car : ");
        wheels[1] = sc.nextInt();
        System.out.print("Enter number of seats of car : ");
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter name of motorcycles : splendar
Enter number of wheel of motorcycles : 2
Enter number of seats of motorcycles : 2
Enter price of motorcycle : 60000

This vehicle came from splendar. It's 2 wheeler vehicle comes with the 2 seats in a price of 60000/-
 only.
Enter company name of car : tesla
Enter number of wheel of car : 4
Enter number of seats of car : 6
Enter price of car : 15000000

This vehicle came from tesla. It's 4 wheeler vehicle comes with the 6 seats in a price of 15000000/-
 only.

Process finished with exit code 0
```

Observations/Remarks #1:

- The Vehicle Class needed 2 data members and I have added 2 more.
- There are 8 getters and setters for each data member.
- Description function describes the object value (class value) in a formal way.

Question #2:

You have to design a class for the calendar. Your class should include basic information related to the calendar. In this, you are allowed to create only one public method, which will decide whether a given year is a leap year or not. Design a constructor through which you can assign the values to your private members. For the output, create three different objects of the calendar for three different years. Output whether the given years are leap years or not.

Note: A year is a leap year if,

- It has an extra day i.e. 366 instead of 365
- It occurs every 4 years e.g. 2008, 2012 are leap years
- For every 100 years, a special rule applies-1900 is not a leap year but 2000 is a leap year.

In those cases, we need to check whether it is divisible by 400 or not.

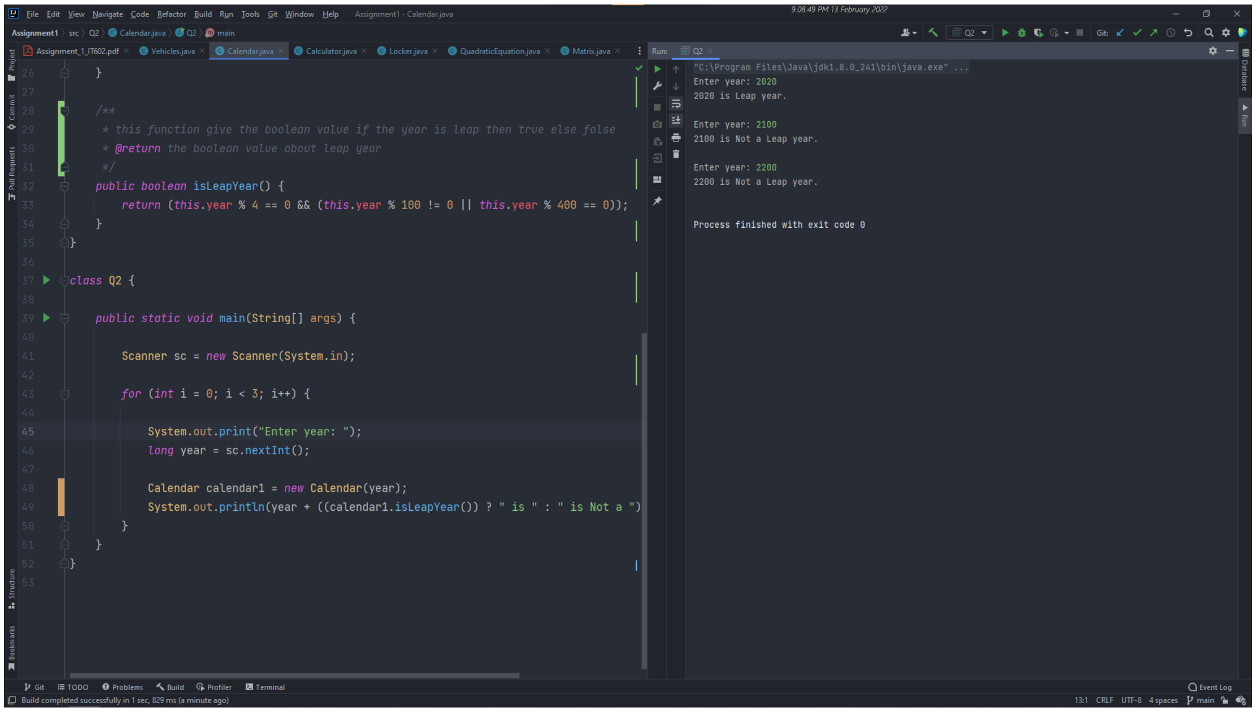
Code #2:

```
/**
 * @author Sagar Variya | 202112114
 */

package Q2;

import java.util.Scanner;

/**
 *  Calendar class contains 1 data members (without getters & setters) that
describes the year.
 *  it has member function that will give the boolean value about year is leap or
not.
 */
```

```
public class Calendar {

    /**
     * private data member
     */
    private final long year;

    /**
     * constructor of the calendar class to set the year
     * @param year year
     */
    public Calendar(long year) {
        this.year = year;
    }

    /**
     * this function give the boolean value if the year is leap then true else false
     * @return the boolean value about leap year
     */
    public boolean isLeapYear() {
        return (this.year % 4 == 0 && (this.year % 100 != 0 || this.year % 400 == 0));
    }
}
```

```
class Q2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

            System.out.print("Enter year: ");
            long year = sc.nextInt();

            Calendar calendar1 = new Calendar(year);
            System.out.println(year + ((calender1.isLeapYear()) ? " is " : " is Not a ") +
"Leap year.\n");        }
    }
}
```

Output #2:



```
    }

    /**
     * this function give the boolean value if the year is leap then true else false
     * @return the boolean value about leap year
     */
    public boolean isLeapYear() {
        return (this.year % 4 == 0 && (this.year % 100 != 0 || this.year % 400 == 0));
    }
}

class Q2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

            System.out.print("Enter year: ");
            long year = sc.nextInt();

            Calendar calendar1 = new Calendar(year);
            System.out.println(year + ((calendar1.isLeapYear()) ? " is " : " is Not a ")
        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter year: 2020
2020 is Leap year.

Enter year: 2100
2100 is Not a Leap year.

Enter year: 2200
2200 is Not a Leap year.


Process finished with exit code 0
```

Build completed successfully in 1 sec, 829 ms (a minute ago)

Observations/Remarks #2:

- Constructo takes year as an argument.
- The isLeapYear function returns the boolean value true for leap year and false for not a leap year.

Question #3:

You are supposed to create a calculator class. In this class, you can create methods for addition, subtraction, and multiplication. Your methods should have support for both integer values and float values. Method names must be the same for the respective tasks. Create overloading methods for doing this. You should have at least two overloaded methods. You can assign values to your private variables via different constructors. (i.e. One for integer values and one for the float values)

Code #3:

```
/**
 * @author Sagar Variya | 202112114
 */

package Q3;

import java.util.Scanner;
import java.util.Objects;
import java.math.BigDecimal;

/**
 * this class has several data members and member function to perform tasks.
 * this calculator class can calculate addition, subtraction and multiplication of
integer and float numbers.
 * it has 3 constructor with no parameter, 1 parameter and 2 parameter.
 * it will run in a loop and ask you about input value like which operations and
numbers.
 * it can decide automatically about the number is integer or float.
 * to exit give 0 when it asks about operations.
 */
```

```
public class Calculator {

    /**
     * private 5 data members
     */
    private long longNum = 0;
    private double doubleA = 0;
    private double doubleB = 0;
    private boolean flag, calculated;
    Scanner sc = new Scanner(System.in);

    /**
     * default constructor.
     */
    public Calculator() {

        takeNumber(1);
        calculate(true);
    }

    /**
     * 1 parameter constructor
     * it will set the 1st number and ask you 2nd number
     * @param a 1st number
     */
    public Calculator(double a) {

        this.doubleA = a;

        showNumber(1);
        calculate(true);
    }

    /**
     * 2 parameter constructor
```

```
 * it will set both the numbers
 * @param a 1st number
 * @param b 2nd number
 */
public Calculator(double a, double b) {

   this.doubleA = a;
   this.doubleB = b;
   showNumber(1);
   showNumber(2);
   calculate(false);
}

/**
 * default addition function
 */
void summation() {
   this.doubleA += this.doubleB;
   this.doubleB = 0;
}

/**
 * it will add 1st number and 2nd number come in the parameter
 * overloaded addition function with the integer value as parameter
 * @param x 2nd number integer type
 */
void summation(long x) {
   this.doubleA += x;
}

/**
 * it will add 1st number and 2nd number come in the parameter
 * overloaded addition function with the float/decimal value as parameter
 * @param x 2nd number float/decimal type
 */
```

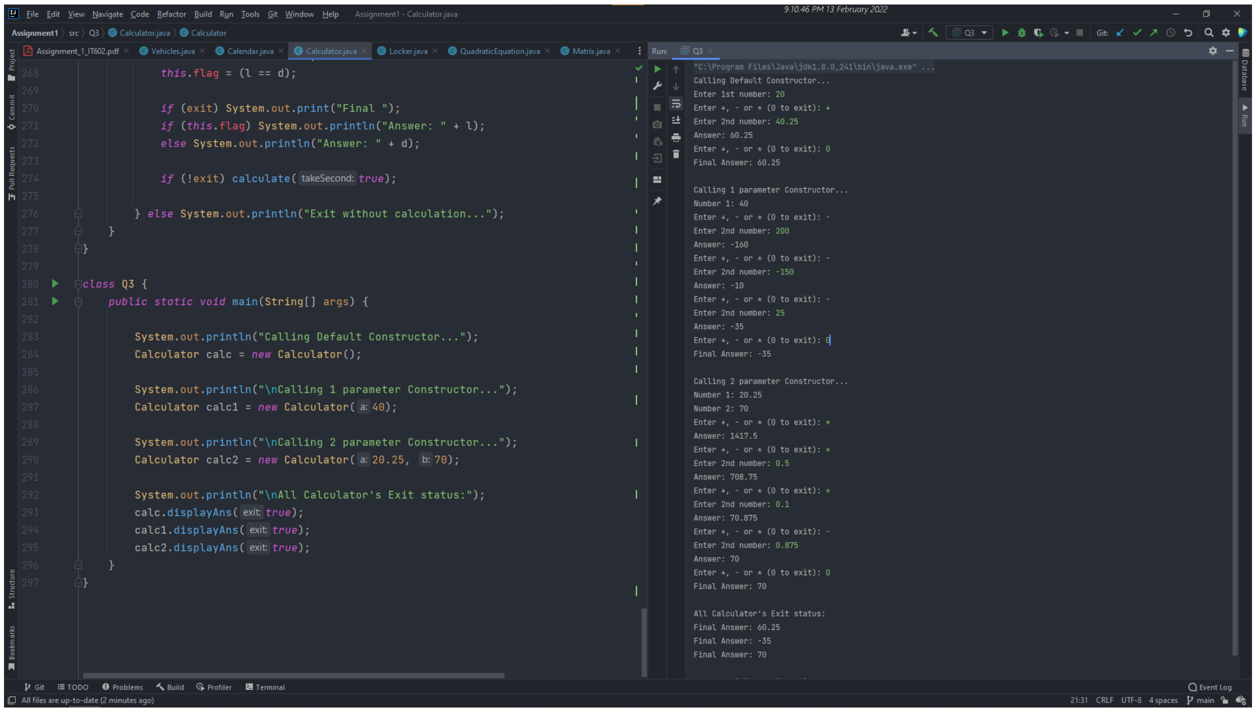
```
void summation(double x) {
   this.doubleA += x;
}

/**
 * default subtraction function
 */
void subtraction() {
   this.doubleA -= this.doubleB;
   this.doubleB = 0;
}

/**
 * it will subtract 1st number and 2nd number come in the parameter
 * overloaded addition function with the integer value as parameter
 * @param x 2nd number integer type
 */
void subtraction(long x) {
   this.doubleA -= x;
}

/**
 * it will subtract 1st number and 2nd number come in the parameter
 * overloaded addition function with the float/decimal value as parameter
 * @param x 2nd number float/decimal type
 */
void subtraction(double x) {
   this.doubleA -= x;
}

/**
 * default multiplication function
 */
```

```
void multiplication() {
   this.doubleA *= this.doubleB;
   this.doubleB = 0;
}

/**
 * it will multiply 1st number and 2nd number come in the parameter
 * overloaded addition function with the integer value as parameter
 * @param x 2nd number integer type
 */
void multiplication(long x) {
   this.doubleA *= x;
}

/**
 * it will multiply 1st number and 2nd number come in the parameter
 * overloaded addition function with the float/decimal value as parameter
 * @param x 2nd number float/decimal type
 */
void multiplication(double x) {
   this.doubleA *= x;
}

/**
 * this function take 1st or 2nd number parameter (which to show e.g. 1 or 2)
 * it will show the number according to its type (integer/float)
 * @param n integer type index number 1 or 2
 */

void showNumber(int n) {

   long l = (n == 1) ? (long) this.doubleA : (long) this.doubleB;
   double d = (n == 1) ? this.doubleA : this.doubleB;
   this.flag = (l == d);
```

```
        if (this.flag) System.out.println("Number " + n + ": " + l);
        else System.out.println("Number " + n + ": " + d);
    }

    /**
     * this function ask use to insert the operation type +, -, *
     * it will continue ask about operation if user give input is other than +, -. * or
zero
     * if you want to quite/exit then enter 0(zero)
     * @return the operation you want to perform
     */
    String askOperation() {

        String s;

        do {
            System.out.print("Enter +, - or * (0 to exit): ");
            s = this.sc.next();
            s = s.trim();

            if (Objects.equals(s, "0")) displayAns(true);

        } while (!s.equals("+") && !s.equals("-") && !s.equals("*") && !s.equals("0"));

        return s;
    }

    /**
     * this function will take the number value from the user to do operation
     * it gives the flag about which type of data(integer/float) user has inserted
     * @param index integer type to take 1st or 2nd number
     */
    void takeNumber(int index) {

        long l;
        double d;
        BigDecimal num;

        System.out.print("Enter " + index + ((index == 1) ? "st" : "nd") + " number: ");
        num = this.sc.nextBigDecimal();

        l = num.longValue();
        d = num.doubleValue();

        if (index == 1) {
            this.doubleA = d;
        } else {
            this.flag = (l == d);
            if (this.flag) this.longNum = l;
            else this.doubleB = d;
        }
    }

    /**
     * this function do the main work of calculation.
     * it will decide with operation(+, -, * or exit) to do according to flag value(integer
or float)
     * @param takeSecond boolean type if it has to take 2nd number than true else
false
     */
    void calculate(boolean takeSecond) {

        String op = askOperation();

        if (!Objects.equals(op, "0")) {

            this.calculated = true;
        if (takeSecond) takeNumber(2);

        if (Objects.equals(op, "+")) {

            if (takeSecond) {
                if (this.flag) summation(this.longNum);
                else summation(this.doubleB);
            } else {
                summation();
            }
            displayAns(false);

        } else if (Objects.equals(op, "-")) {

            if (takeSecond) {
                if (this.flag) subtraction(this.longNum);
                else subtraction(this.doubleB);
            } else {
                subtraction();
            }
            displayAns(false);

        } else if (Objects.equals(op, "*")) {

            if (takeSecond) {
                if (this.flag) multiplication(this.longNum);
                else multiplication(this.doubleB);
            } else {
                multiplication();
            }
            displayAns(false);

        } else System.out.println("Invalid Operation...");
    }
}
```

```
    /**
     * this function displays the answer of the calculation
     * if user hasn't done any calculation then it will show "Exit without
calculation..."
     * this is the only public function in the class
     * @param exit boolean value to check if true then exit else continue
     */
    public void displayAns(boolean exit) {

        if (this.calculated) {
            long l = (long) this.doubleA;
            double d = this.doubleA;
            this.flag = (l == d);

            if (exit) System.out.print("Final ");
            if (this.flag) System.out.println("Answer: " + l);
            else System.out.println("Answer: " + d);

            if (!exit) calculate(true);

        } else System.out.println("Exit without calculation...");
    }
}


class Q3 {
    public static void main(String[] args) {

        System.out.println("Calling Default Constructor...");
        Calculator calc = new Calculator();

        System.out.println("\nCalling 1 parameter Constructor...");
        Calculator calc1 = new Calculator(40);
        System.out.println("\nCalling 2 parameter Constructor...");
        Calculator calc2 = new Calculator(20.25, 70);

        System.out.println("\nAll Calculator's Exit status:");
        calc.displayAns(true);
        calc1.displayAns(true);
        calc2.displayAns(true);
    }
}
```

Output #3:



```java
            this.flag = (l == d);

            if (exit) System.out.print("Final ");
            if (this.flag) System.out.println("Answer: " + l);
            else System.out.println("Answer: " + d);

            if (!exit) calculate( takeSecond: true);

        } else System.out.println("Exit without calculation...");
    }
}

class Q3 {
    public static void main(String[] args) {

        System.out.println("Calling Default Constructor...");
        Calculator calc = new Calculator();

        System.out.println("\nCalling 1 parameter Constructor...");
        Calculator calc1 = new Calculator( a: 40);

        System.out.println("\nCalling 2 parameter Constructor...");
        Calculator calc2 = new Calculator( a: 20.25,  b: 70);

        System.out.println("\nAll Calculator's Exit status:");
        calc.displayAns( exit: true);
        calc1.displayAns( exit: true);
        calc2.displayAns( exit: true);
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Calling Default Constructor...
Enter 1st number: 20
Enter +, - or * (0 to exit): +
Enter 2nd number: 40.25
Answer: 60.25
Enter +, - or * (0 to exit): 0
Final Answer: 60.25

Calling 1 parameter Constructor...
Number 1: 40
Enter +, - or * (0 to exit): -
Enter 2nd number: 200
Answer: -160
Enter +, - or * (0 to exit): -
Enter 2nd number: -150
Answer: -10
Enter +, - or * (0 to exit): -
Enter 2nd number: 25
Answer: -35
Enter +, - or * (0 to exit): 0
Final Answer: -35

Calling 2 parameter Constructor...
Number 1: 20.25
Number 2: 70
Enter +, - or * (0 to exit): *
Answer: 1417.5
Enter +, - or * (0 to exit): *
Enter 2nd number: 0.5
Answer: 708.75
Enter +, - or * (0 to exit): *
Enter 2nd number: 0.1
Answer: 70.875
Enter +, - or * (0 to exit): -
Enter 2nd number: 0.875
Answer: 70
Enter +, - or * (0 to exit): 0
Final Answer: 70

All Calculator's Exit status:
Final Answer: 60.25
Final Answer: -35
Final Answer: 70
```

Observations/Remarks #3:

- This calculator runs in an infinite loop once the constructor calls.
- It will show value in integer or float accordingly.
- To get out of the loop just enter zero when it asks about operation.
- Minimum constructor parameter requirement is 0 and maximum is 2.
- Program will run according to different constructor calls.

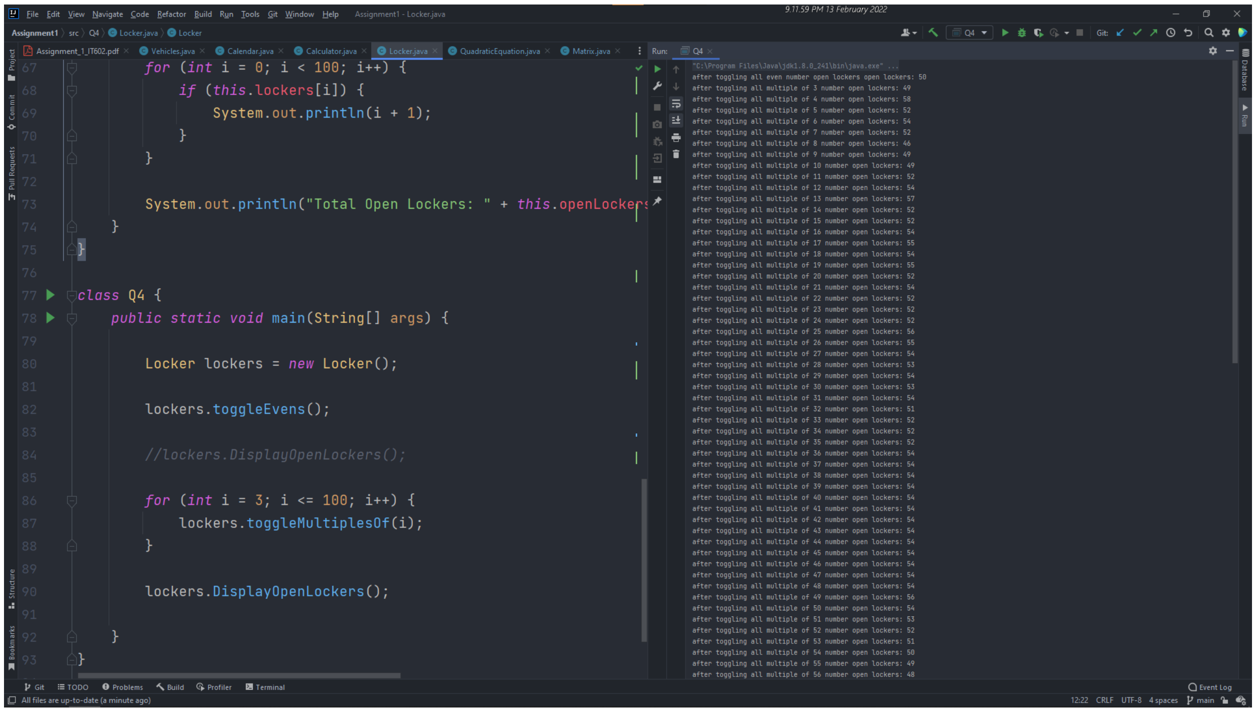
Question #4:

You are in a locker room with 100 open lockers, numbered 1 to 100. Toggle all of the lockers that are even. By toggle, we mean close if it is open, and open if it is closed. Now toggle all of the lockers that are multiples of three. Repeat with multiples of 4, 5, up to 100. Output the number of lockers that will be open after the simulation.

Code #4:

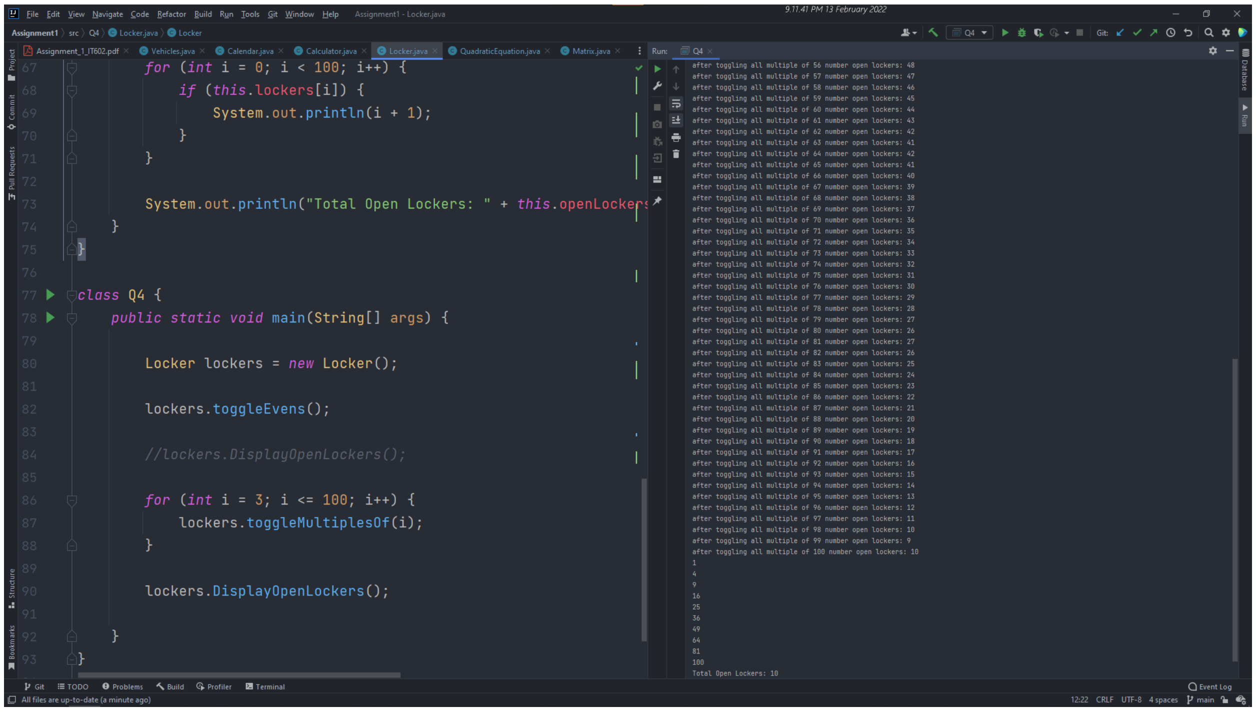
```
/**
 * @author Sagar Variya | 202112114
 */
package Q4;

/**
 * this locker class is about the open and closed ones
 * it has default 100 lockers all are open
 * there are 2 functions to toggle the locker opening and closing
 */
public class Locker {

    /**
     * initializing and declaring private data member
     * lockers array to hold 100 locker value in boolean
     * openLockers to count how many lockers are open
     */
    private final boolean[] lockers = new boolean[100];
    private int openLockers;

    /**
     * default constructor
     * it will set all the lockers open and total openLockers to 100
     */
    public Locker() {
        for (int i = 0; i < 100; i++) {
            this.lockers[i] = true;
        }
        this.openLockers = 100;
    }

    /**
     * this function will toggle all the even number of lockers
     */
    // same as toggleMultiplesOf(2) but its requirement
    public void toggleEvens() {
        for (int i = 1; i < 100; i += 2) {

            if (!this.lockers[i]) this.openLockers++;
            this.lockers[i] = !this.lockers[i];
            if (!this.lockers[i]) this.openLockers--;
        }

        System.out.println("after toggling all even number open lockers open lockers:
" + this.openLockers);
    }

    /**
     * this function take integer number and toggle it's all multiply number of
lockers
     * @param x integer type | multiply of x
     */
    public void toggleMultiplesOf(int x) {
        for (int i = x - 1; i < 100; i += x) {

            if (!this.lockers[i]) this.openLockers++;
            this.lockers[i] = !this.lockers[i];
            if (!this.lockers[i]) this.openLockers--;
        }
        System.out.println("after toggling all multiple of " + x + " number open lockers: 
" + this.openLockers);
    }

    /**
     * this function displays all the open lockers and total count of open lockers
     */
    public void DisplayOpenLockers() {
        for (int i = 0; i < 100; i++) {
            if (this.lockers[i]) {
                System.out.println(i + 1);
            }
        }

        System.out.println("Total Open Lockers: " + this.openLockers);
    }
}

class Q4 {
    public static void main(String[] args) {

        Locker lockers = new Locker();

        lockers.toggleEvens();

        //lockers.DisplayOpenLockers();

        for (int i = 3; i <= 100; i++) {
            lockers.toggleMultiplesOf(i);
        }

        lockers.DisplayOpenLockers();
    }
}
```

Output #4:



```
        for (int i = 0; i < 100; i++) {
            if (this.lockers[i]) {
                System.out.println(i + 1);
            }
        }

        System.out.println("Total Open Lockers: " + this.openLocker
    }
}

class Q4 {
    public static void main(String[] args) {

        Locker lockers = new Locker();

        lockers.toggleEvens();

        //lockers.DisplayOpenLockers();

        for (int i = 3; i <= 100; i++) {
            lockers.toggleMultiplesOf(i);
        }

        lockers.DisplayOpenLockers();

    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
after toggling all even number open lockers open lockers: 50
after toggling all multiple of 3 number open lockers: 49
after toggling all multiple of 4 number open lockers: 58
after toggling all multiple of 5 number open lockers: 52
after toggling all multiple of 6 number open lockers: 54
after toggling all multiple of 7 number open lockers: 52
after toggling all multiple of 8 number open lockers: 46
after toggling all multiple of 9 number open lockers: 49
after toggling all multiple of 10 number open lockers: 49
after toggling all multiple of 11 number open lockers: 52
after toggling all multiple of 12 number open lockers: 54
after toggling all multiple of 13 number open lockers: 57
after toggling all multiple of 14 number open lockers: 52
after toggling all multiple of 15 number open lockers: 52
after toggling all multiple of 16 number open lockers: 54
after toggling all multiple of 17 number open lockers: 55
after toggling all multiple of 18 number open lockers: 54
after toggling all multiple of 19 number open lockers: 55
after toggling all multiple of 20 number open lockers: 52
after toggling all multiple of 21 number open lockers: 54
after toggling all multiple of 22 number open lockers: 52
after toggling all multiple of 23 number open lockers: 52
after toggling all multiple of 24 number open lockers: 52
after toggling all multiple of 25 number open lockers: 56
after toggling all multiple of 26 number open lockers: 55
after toggling all multiple of 27 number open lockers: 54
after toggling all multiple of 28 number open lockers: 53
after toggling all multiple of 29 number open lockers: 54
after toggling all multiple of 30 number open lockers: 53
after toggling all multiple of 31 number open lockers: 54
after toggling all multiple of 32 number open lockers: 51
after toggling all multiple of 33 number open lockers: 52
after toggling all multiple of 34 number open lockers: 52
after toggling all multiple of 35 number open lockers: 52
after toggling all multiple of 36 number open lockers: 54
after toggling all multiple of 37 number open lockers: 54
after toggling all multiple of 38 number open lockers: 54
after toggling all multiple of 39 number open lockers: 54
after toggling all multiple of 40 number open lockers: 54
after toggling all multiple of 41 number open lockers: 54
after toggling all multiple of 42 number open lockers: 54
after toggling all multiple of 43 number open lockers: 54
after toggling all multiple of 44 number open lockers: 54
after toggling all multiple of 45 number open lockers: 54
after toggling all multiple of 46 number open lockers: 54
after toggling all multiple of 47 number open lockers: 54
after toggling all multiple of 48 number open lockers: 54
after toggling all multiple of 49 number open lockers: 56
after toggling all multiple of 50 number open lockers: 54
after toggling all multiple of 51 number open lockers: 53
after toggling all multiple of 52 number open lockers: 52
after toggling all multiple of 53 number open lockers: 51
after toggling all multiple of 54 number open lockers: 50
after toggling all multiple of 55 number open lockers: 49
after toggling all multiple of 56 number open lockers: 48
```



```
        for (int i = 0; i < 100; i++) {
            if (this.lockers[i]) {
                System.out.println(i + 1);
            }
        }

        System.out.println("Total Open Lockers: " + this.openLocker
    }
}

class Q4 {
    public static void main(String[] args) {

        Locker lockers = new Locker();

        lockers.toggleEvens();

        //lockers.DisplayOpenLockers();

        for (int i = 3; i <= 100; i++) {
            lockers.toggleMultiplesOf(i);
        }

        lockers.DisplayOpenLockers();

    }
}
```

```
after toggling all multiple of 56 number open lockers: 48
after toggling all multiple of 57 number open lockers: 47
after toggling all multiple of 58 number open lockers: 46
after toggling all multiple of 59 number open lockers: 45
after toggling all multiple of 60 number open lockers: 44
after toggling all multiple of 61 number open lockers: 43
after toggling all multiple of 62 number open lockers: 42
after toggling all multiple of 63 number open lockers: 41
after toggling all multiple of 64 number open lockers: 42
after toggling all multiple of 65 number open lockers: 41
after toggling all multiple of 66 number open lockers: 40
after toggling all multiple of 67 number open lockers: 39
after toggling all multiple of 68 number open lockers: 38
after toggling all multiple of 69 number open lockers: 37
after toggling all multiple of 70 number open lockers: 36
after toggling all multiple of 71 number open lockers: 35
after toggling all multiple of 72 number open lockers: 34
after toggling all multiple of 73 number open lockers: 33
after toggling all multiple of 74 number open lockers: 32
after toggling all multiple of 75 number open lockers: 31
after toggling all multiple of 76 number open lockers: 30
after toggling all multiple of 77 number open lockers: 29
after toggling all multiple of 78 number open lockers: 28
after toggling all multiple of 79 number open lockers: 27
after toggling all multiple of 80 number open lockers: 26
after toggling all multiple of 81 number open lockers: 27
after toggling all multiple of 82 number open lockers: 26
after toggling all multiple of 83 number open lockers: 25
after toggling all multiple of 84 number open lockers: 24
after toggling all multiple of 85 number open lockers: 23
after toggling all multiple of 86 number open lockers: 22
after toggling all multiple of 87 number open lockers: 21
after toggling all multiple of 88 number open lockers: 20
after toggling all multiple of 89 number open lockers: 19
after toggling all multiple of 90 number open lockers: 18
after toggling all multiple of 91 number open lockers: 17
after toggling all multiple of 92 number open lockers: 16
after toggling all multiple of 93 number open lockers: 15
after toggling all multiple of 94 number open lockers: 14
after toggling all multiple of 95 number open lockers: 13
after toggling all multiple of 96 number open lockers: 12
after toggling all multiple of 97 number open lockers: 11
after toggling all multiple of 98 number open lockers: 10
after toggling all multiple of 99 number open lockers: 9
after toggling all multiple of 100 number open lockers: 10
1
4
9
16
25
36
49
64
81
100
Total Open Lockers: 10
```

Observations/Remarks #4:

- Array starts from 0 so needed to subtract 1 from the given arguments in the toggle function.
- Once the locker will close or open the total open loacker value increases or decreases accordingly.

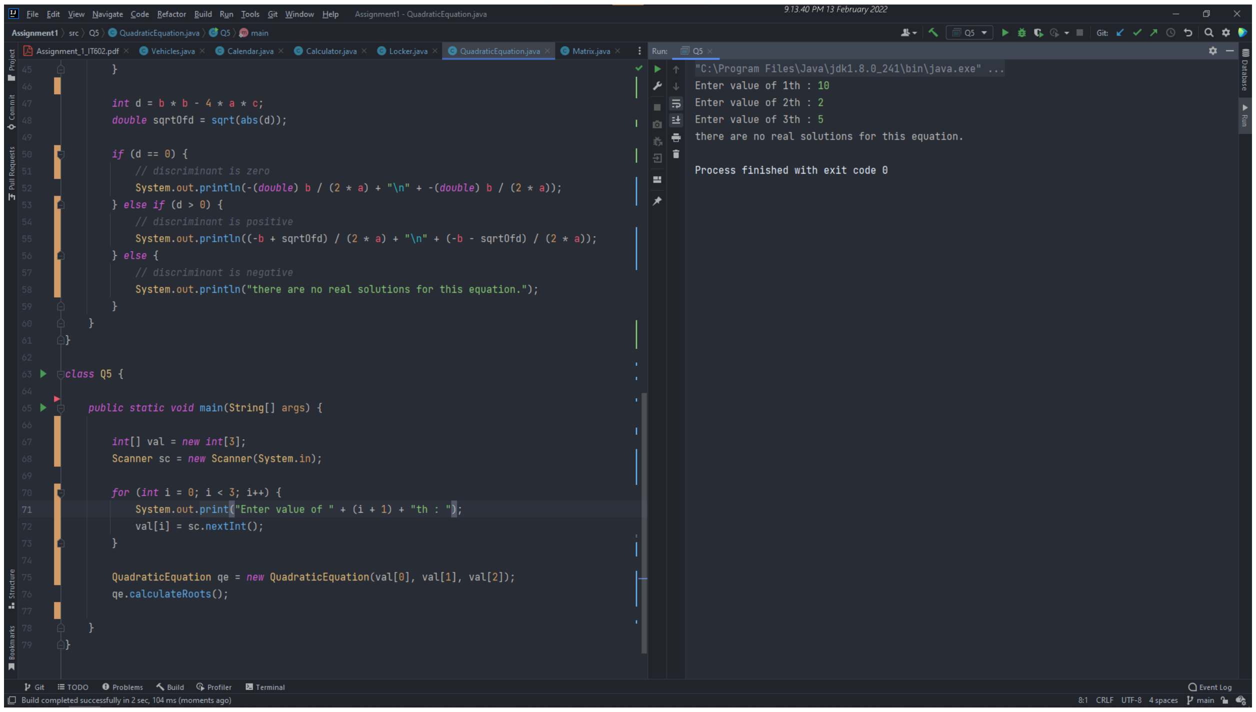
Question #5:

Write a Java program that prints all real solutions to the quadratic equation ax2+ bx + c = 0. Read in a, b, c and use the quadratic formula. Before finding the solution, you should make a check for the real solutions. Find the value of the discriminant (i.e. b 2 - 4ac). If the discriminant value is negative, display a message stating that there are no real solutions. To implement this solution, make an appropriate class and appropriate methods.

Code #5:

```
/**
 * @author Sagar Variya | 202112114
 */

package Q5;

import java.util.Scanner;

import static java.lang.Math.*;

/**
 * this class can calculate the quadratic equation : ax^2 + bx + c = 0
 */
public class QuadraticEquation {

    /**
     * ax^2+bx+c=0
     * declaring 3 private data members
     */
    private final int a, b, c;
    /**
     * public constructor that sets the value of equation
     *
     * @param a integer type A's value
     * @param b integer type B's value
     * @param c integer type C's value
     */
    public QuadraticEquation(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    /**
     * this function will find the roots of the quadratic equation
     * first it will compare the value of a with 0, if A is 0 then the equation is not
quadratic
     * then it will calculate discriminant (d)
     */
    public void calculateRoots() {

        if (a == 0) {
            System.out.println("The value of A cannot be 0.");
            return;
        }

        int d = b * b - 4 * a * c;
        double sqrtOfd = sqrt(abs(d));

        if (d == 0) {
            // discriminant is zero
            System.out.println(-(double) b / (2 * a) + "\n" + -(double) b / (2 * a));
        } else if (d > 0) {
            // discriminant is positive
            System.out.println((-b + sqrtOfd) / (2 * a) + "\n" + (-b - sqrtOfd) / (2 * a));
        } else {
            // discriminant is negative
            System.out.println("there are no real solutions for this equation.");
        }
    }
}

class Q5 {

    public static void main(String[] args) {

        int[] val = new int[3];
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 3; i++) {
            System.out.print("Enter value of " + (i + 1) + "th : ");
            val[i] = sc.nextInt();
        }

        QuadraticEquation qe = new QuadraticEquation(val[0], val[1], val[2]);
        qe.calculateRoots();

    }
}
```

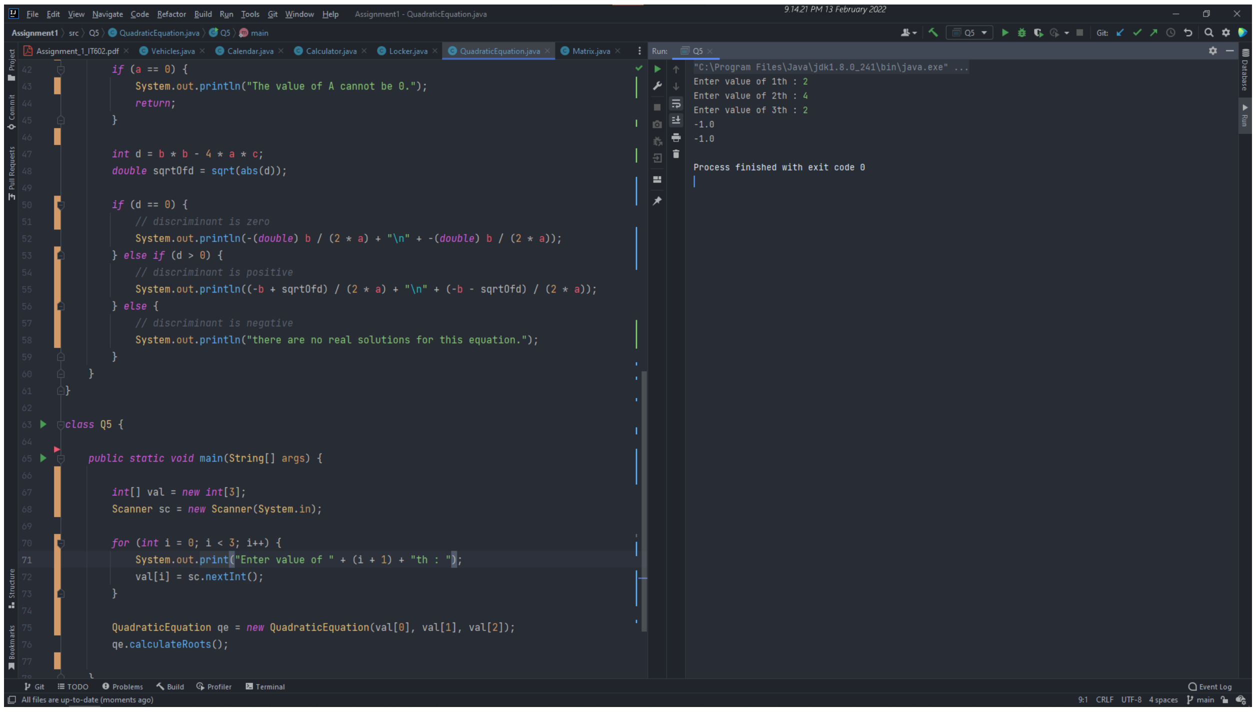
Output #5:



```java
        }

        int d = b * b - 4 * a * c;
        double sqrtOfd = sqrt(abs(d));

        if (d == 0) {
            // discriminant is zero
            System.out.println(-(double) b / (2 * a) + "\n" + -(double) b / (2 * a));
        } else if (d > 0) {
            // discriminant is positive
            System.out.println((-b + sqrtOfd) / (2 * a) + "\n" + (-b - sqrtOfd) / (2 * a));
        } else {
            // discriminant is negative
            System.out.println("there are no real solutions for this equation.");
        }
    }
}

class Q5 {

    public static void main(String[] args) {

        int[] val = new int[3];
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 3; i++) {
            System.out.print("Enter value of " + (i + 1) + "th : ");
            val[i] = sc.nextInt();
        }

        QuadraticEquation qe = new QuadraticEquation(val[0], val[1], val[2]);
        qe.calculateRoots();

    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter value of 1th : 10
Enter value of 2th : 2
Enter value of 3th : 5
there are no real solutions for this equation.

Process finished with exit code 0
```

Build completed successfully in 2 sec, 104 ms (moments ago)



```java
        if (a == 0) {
            System.out.println("The value of A cannot be 0.");
            return;
        }

        int d = b * b - 4 * a * c;
        double sqrtOfd = sqrt(abs(d));

        if (d == 0) {
            // discriminant is zero
            System.out.println(-(double) b / (2 * a) + "\n" + -(double) b / (2 * a));
        } else if (d > 0) {
            // discriminant is positive
            System.out.println((-b + sqrtOfd) / (2 * a) + "\n" + (-b - sqrtOfd) / (2 * a));
        } else {
            // discriminant is negative
            System.out.println("there are no real solutions for this equation.");
        }
    }
}

class Q5 {

    public static void main(String[] args) {

        int[] val = new int[3];
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 3; i++) {
            System.out.print("Enter value of " + (i + 1) + "th : ");
            val[i] = sc.nextInt();
        }

        QuadraticEquation qe = new QuadraticEquation(val[0], val[1], val[2]);
        qe.calculateRoots();
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter value of 1th : 2
Enter value of 2th : 4
Enter value of 3th : 2
-1.0
-1.0

Process finished with exit code 0
```

All files are up-to-date (moments ago)

Observations/Remarks #5:

- If the user enters A = 0 then the program will terminate showing msg "a cannot be 0".

Question #6:

The program needs to calculate the nth power of a matrix whose elements, as well as the value of n, are specified by the user. You should store your matrix in the 2D Array. You must create supporting methods for doing matrix multiplications.

Code #6:

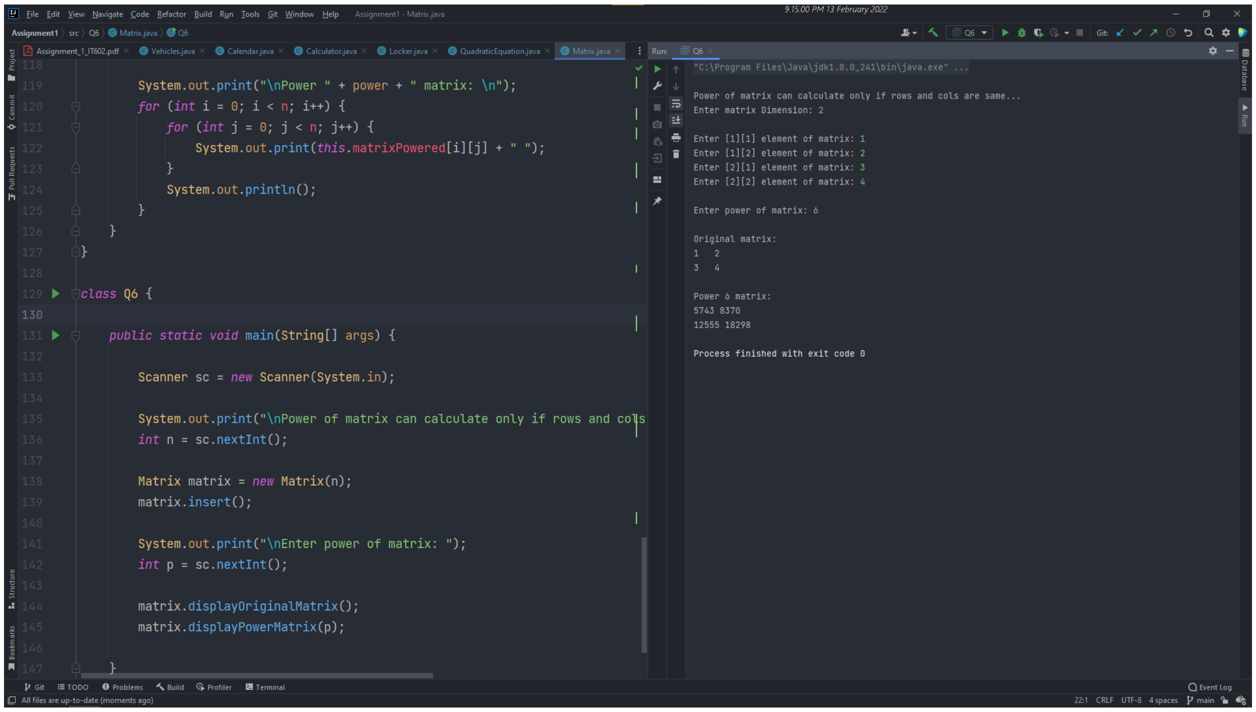
```
/**
 * @author Sagar Variya | 202112114
 */

package Q6;
import java.util.Scanner;

/**
 * the matrix class can take any dimension of matrix and can provide the power if
it.
 * to calculate the power of matrix it should be square like 2*2, 3*3 etc...
 */
public class Matrix {

   /**
    * declaring private data members
    */
   private final int n;
   private long[][] matrixPowered;
   private long[][] identityMatrix;
   private final long[][] matrixOriginal;

   /**
    * constructor with 1 parameter
    * it will create the size of the matrix
    * @param n integer type to create n*n matrix
    */
   public Matrix(int n) {
       this.matrixOriginal = new long[n][n];
       this.n = n;
   }

   /**
    * this function used to insert the matrix value.
    * just call it, and it will ask user all the values and store it.
    */
   public void insert() {
       Scanner sc = new Scanner(System.in);

       System.out.println();
       for (int i = 0; i < n; i++) {
           for (int j = 0; j < n; j++) {
               System.out.print("Enter [" + (i + 1) + "][" + (j + 1) + "] element of matrix: ");
               this.matrixOriginal[i][j] = sc.nextLong();
           }
       }
   }

   /**
    * this function will display original matrix user has inserted
    */
   public void displayOriginalMatrix() {
       System.out.print("\nOriginal matrix: \n");
       for (int i = 0; i < n; i++) {
           for (int j = 0; j < n; j++) {
               System.out.print(this.matrixOriginal[i][j] + "\t");
           }
           System.out.println();
       }
   }
```

```
    /**
     * this function will handle the multiplication of the matrix
     * it will multiply by itself or by identity according to parameter value came and
also store it to accordingly.
     * @param withIdentity boolean type if with identity then true else false
     */
    void matrixMultiplication(final boolean withIdentity) {
        long temp, mod = Long.MAX_VALUE;
        long[][] tempMatrix = new long[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                tempMatrix[i][j] = 0;
                for (int k = 0; k < n; k++) {
                    if (withIdentity) temp = (this.identityMatrix[i][k] *
this.matrixPowered[k][j]) % mod;
                    else temp = (this.matrixPowered[i][k] * this.matrixPowered[k][j]) % mod;

                    tempMatrix[i][j] = (tempMatrix[i][j] + temp) % mod;
                }
            }
        }

        if (withIdentity) this.identityMatrix = tempMatrix;
        else this.matrixPowered = tempMatrix;
    }

    /**
     * this function is used to calculate the matrix to multiply by itself or identity
     * it will reduce the multiplication time complexity from O(n) to O(log n)
     * to reduce time complexity this function uses identity matrix (all diagonal
elements are 1).
     * so, the multiplication with identity matrix will produce the multiply by 1 and
store it in
```

```
     * @param power integer type power of the matrix
     */
    void calculateMatrixPower(int power) {
        this.matrixPowered = this.matrixOriginal;
        this.identityMatrix = new long[n][n];
        for (int i = 0; i < n; i++) this.identityMatrix[i][i] = 1;

        while (power != 0) {
            if (power % 2 != 0) {
                matrixMultiplication(true);
                power--;
            } else {
                matrixMultiplication(false);
                power /= 2;
            }
        }
        this.matrixPowered = this.identityMatrix;
    }

    /**
     * this function take power number of matrix then calculate and show the result
     * @param power integer type power of the matrix
     */
    public void displayPowerMatrix(final int power) {
        calculateMatrixPower(power);

        System.out.print("\nPower " + power + " matrix: \n");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(this.matrixPowered[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

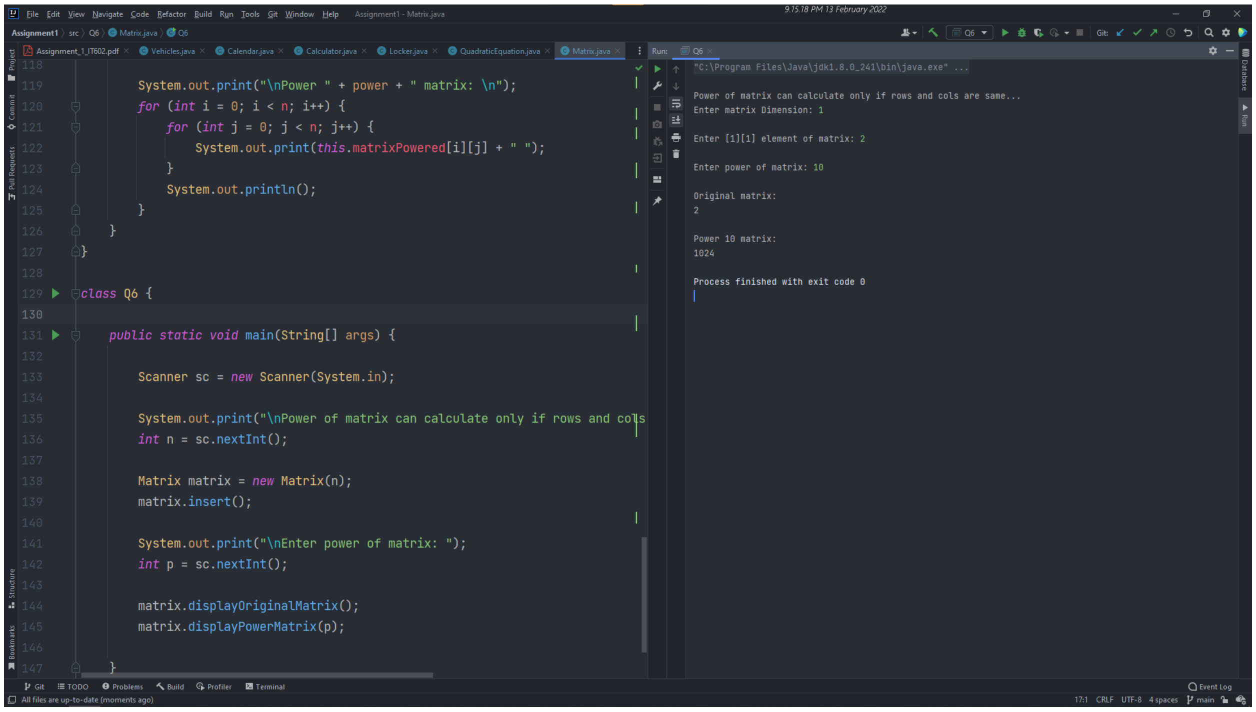
```
class Q6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nPower of matrix can calculate only if rows and cols are
same...\nEnter matrix Dimension: ");
        int n = sc.nextInt();

        Matrix matrix = new Matrix(n);
        matrix.insert();

        System.out.print("\nEnter power of matrix: ");
        int p = sc.nextInt();

        matrix.displayOriginalMatrix();
        matrix.displayPowerMatrix(p);

    }
}
```

Output #6:



```java
            System.out.print("\nPower " + power + " matrix: \n");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.print(this.matrixPowered[i][j] + " ");
                }
                System.out.println();
            }
        }
}

class Q6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nPower of matrix can calculate only if rows and cols
        int n = sc.nextInt();

        Matrix matrix = new Matrix(n);
        matrix.insert();

        System.out.print("\nEnter power of matrix: ");
        int p = sc.nextInt();

        matrix.displayOriginalMatrix();
        matrix.displayPowerMatrix(p);

    }
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Power of matrix can calculate only if rows and cols are same...
Enter matrix Dimension: 2

Enter [1][1] element of matrix: 1
Enter [1][2] element of matrix: 2
Enter [2][1] element of matrix: 3
Enter [2][2] element of matrix: 4

Enter power of matrix: 6

Original matrix:
1   2
3   4

Power 6 matrix:
5743 8370
12555 18298

Process finished with exit code 0
```



```java
            System.out.print("\nPower " + power + " matrix: \n");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.print(this.matrixPowered[i][j] + " ");
                }
                System.out.println();
            }
        }
}

class Q6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("\nPower of matrix can calculate only if rows and cols
        int n = sc.nextInt();

        Matrix matrix = new Matrix(n);
        matrix.insert();

        System.out.print("\nEnter power of matrix: ");
        int p = sc.nextInt();

        matrix.displayOriginalMatrix();
        matrix.displayPowerMatrix(p);

    }
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Power of matrix can calculate only if rows and cols are same...
Enter matrix Dimension: 1

Enter [1][1] element of matrix: 2

Enter power of matrix: 10

Original matrix:
2

Power 10 matrix:
1024

Process finished with exit code 0
```

Observations/Remarks #6:

- Power of matrix calculated only if the matrix is square. So, the user hase to enter only 1 number as dimensions.
- The power of any number means to multiply the number by itself nth - 1 times so the brute force approach is to multiply power matrix with the original matrix nth - 1 times.
- Example: 2^{10} : 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 1024
- In this approach the time complexity will be O(n).
- My approach will reduce the the time complexity from O(n) to O(log n).
- The idea is to multiply the multiplied number with itself rethen then multiply with the original number when the power is even.
- Example: 2^{10} : if power is even then (/2) else (-1)
- original = 2 & powered = 1 & power = 10
- Step 1 p(10) : (2 * 2) = 4 | original = 2 & powered = 4 & temp= 1
- Step 2 p(5)   : (4 * 1) = 8 | original = 2 & powered = 4 & temp= 4
- Step 1 p(4)   : (4 * 4) = 16 | original = 2 & powered = 16 & temp= 4
- Step 1 p(2):(16 * 16) = 256 | original = 2 & powered = 256 & temp= 4
- Step 1 p(1) : (256 * 4) = 1024| original = 2 & powered = 1024 & temp= 4
- Now power = 0 and powered = 1024.