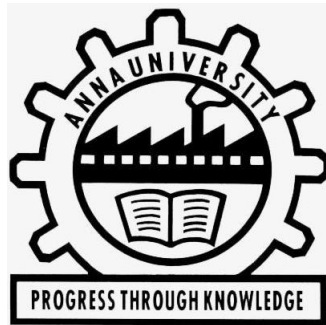


DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044



BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by

Mr/Ms. _____ of Register Number _____

for the _____ laboratory during the period

August 2023 to November 2023.

Date:

Lab In-charge

INDEX

EX NO	DATE	EXPERIMENT NAME	PAGE NO	REMARKS
1		JAVA BASICS		
2		CONTROL STRUCTURES IN JAVA		
3		CONSTRUCTORS AND METHODS IN JAVA		
4		ARRAYS IN JAVA		
5		STRING HANDLING IN JAVA		
6		INHERITANCE AND INTERFACE IN JAVA		
7		EXCEPTION HANDLING MECHANISM IN JAVA		
8		FILE HANDLING IN JAVA		
9		GUI		
10		MULTITHREADING AND DEADLOCKS		
11		SOCKET PROGRAMMING USING TCP AND UDP		
12		JAVA DATABASE CONNECTION		
13		COLLECTIONS		
14		SERVLETS AND JSP		

BASICS OF JAVA PROGRAMMING

Aim: To explore about basics of java programming

Algorithm:

Step 1: To print Hello world in Java

- `public static void main(String args[])`: The main method is the entry point of the program, where execution begins.
- `System.out.print()` → Used to print the data in output console

Step 2: To display user profile

- `in.close()`: Closes the Scanner object to release system resources.
- `in.nextLine()`: Reads the entire line of text as input, which is used for capturing the user's name and address.
- `in.nextInt()`: Reads an integer input, which is used for capturing the user's register number.
- `in.next()`: Reads a single word or token of input, which is used for capturing the user's phone number

Step 3: To add two numbers

- `Scanner in = new Scanner(System.in)`: Creates a Scanner object for reading user input from the console.
- `System.out.println(String s)`: Prints the specified string to the standard output.
- `in.nextInt()`: Reads an integer input from the user, capturing the first number.

Step 4: To display welcome note

- `input.nextLine()`: Reads the entire line of text as input, capturing the user's name.
- `input.nextLong()`: Reads a long integer input from the user, capturing the user's register number.
- String concatenation: The code uses concatenation (+) to create a formatted welcome message that includes the user's name and register number.

Step 5: To build chat application

- `Scanner in = new Scanner(System.in);` Creates a Scanner object for reading user input from the console.
- `ArrayList<String> chats = new ArrayList<>();` Creates an ArrayList to store chat messages.
- `in.nextLine();` Reads the entire line of text as input, capturing chat messages for both "Person 1" and "Person 2."
- `in.close();` Closes the Scanner object to release system resources

Step 6: To compile and run a java program

- `javac Filename.java` → To compile the java program
- `java Classname` → To run the java program

Step 7: To print current date and time

- `java.time.LocalDate.now();` → To get current System Date
- `java.time.LocalTime.now();` → To get current System Time

Step 8: Identify the most error-prone one-line output & provide a brief description.

- `System.out.print();` → Used to print the data in output console
- `System.out.print(args[0]);` → Used to print the cmdline input in output console

Step 9: Identify the most error-prone statement and rectify them.

- `new Scanner(System.in)` → Initialize the Scanner object
- `in.nextInt();` → Get the integer input from the user in command line during execution

Step 10: To know about Data types in Java

- `int`: -2,147,483,648 to 2,147,483,647
- `float`: Approximately 1.4e-45 to 3.4e38
- `byte`: -128 to 127
- `long`: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
- `Boolean`: Accepts only true and false

Step 11: To convert Fahrenheit to Celsius and vice versa

- `new DecimalFormat("0.00");` Creates a DecimalFormat object for formatting floating-point numbers with two decimal places.

- `decimalFormat.format(double d)`: Formats a double value as a string with the specified formatting.
- `decimalFormat.setRoundingMode(RoundingMode.DOWN)`: Sets the rounding mode for the `DecimalFormat` to round down (toward zero)
- Convert temperatures between Fahrenheit and Celsius $(f - 32) * 5 / 9$ and $(9 * c / 5) + 32$.

Step 12: To find the velocity

- `in.nextDouble()`: Reads a double value input from the user.
- $v = u + a*t$: Finding velocity of the program
- `in.close()` : Closes the Scanner object to release system resources

1.1) Write a java to print Hello world

CODE:

```
class HelloWorld3568{
    public static void main(String args[]){
        System.out.println("Hello World!!");
    }
}
```

OUTPUT:

```
PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208>
Hello World!!
```

1.2) Write a Java program to get the user profile details and display them.

CODE:

```
import java.util.Scanner;
class Profile3568{
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);

        System.out.println("Enter your name: ");
        String name = in.nextLine();
```

```

System.out.println("Enter your register number: ");
int reg = in.nextInt();
System.out.println("Enter your phone number: ");
String phone = in.next();
System.out.println("Enter your Address: ");
String address = in.next();
address += in.nextLine();

System.out.println("Hello " + name );
System.out.println("Your register number is " + reg );
System.out.println("Your Phone number is " + phone);
System.out.println("Your are residing in the following address: "+address);
in.close();
}
}

```

OUTPUT:

```

PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208> cd "c:\Users\2021503568\
Enter your name:
Vijai Suria
Enter your register number:
2021503568
Enter your phone number:
6381544020
Enter your Address:
1/106A, South Street, Salem
Hello Vijai Suria
Your register number is 2021503568
Your Phone number is 6381544020
Your are residing in the following address: 1/106A, South Street, Salem

```

1.3) Write a Java program to add two given numbers from the user

CODE:

```

import java.util.Scanner;
class Add3568 {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Number 1: ");

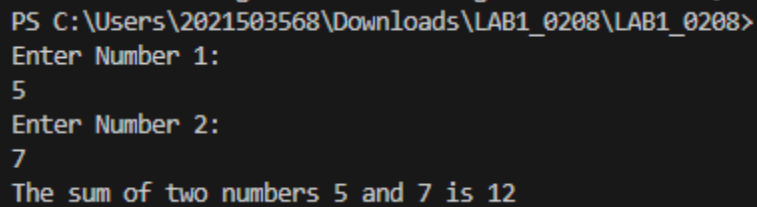
```

```

int a=in.nextInt();
System.out.println("Enter Number 2: ");
int b=in.nextInt();
System.out.print("The sum of two numbers " + a + " and " + b + " is " +
(a+b));
    in.close();
}
}

```

OUTPUT:



```

PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208>
Enter Number 1:
5
Enter Number 2:
7
The sum of two numbers 5 and 7 is 12

```

1.4) Write a Java program to display the welcome note to the user

CODE:

```

import java.util.Scanner;
class Welcome3568{
    public static void main(String args[]){
        String name;
        long reg;
        Scanner input = new Scanner(System.in);
        System.out.println("Enter your Name: ");
        name = input.nextLine();
        System.out.println("Enter your Register Number: ");
        reg = input.nextLong();
        System.out.println("Welcome to MIT, \nMr. " + name + "\nYour register is " +
reg);
        input.close();
    }
}

```

OUTPUT:

```
PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208>
Enter your Name:
Vijai
Enter your Register Number:
2021503568
Welcome to MIT,
Mr. Vijai
Your register is 2021503568
```

1.5) Write a Java Program to build the chat application where two user can chat.

CODE:

```
import java.util.*;
import java.time.LocalDateTime;
import java.time.LocalDate;
class Chat3568{
    public static void main(String args[]){
        System.out.println("Name: Vijai Suria M");
        System.out.println("Register Number: 2021503568");
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("\nChat Application: ");
        String p1="", p2="";
        Scanner in = new Scanner(System.in);
        ArrayList<String> chats = new ArrayList<>();
        while(!p1.equals("bye") && !p2.equals("bye")){
            System.out.println("Person 1");
            p1=in.nextLine();
            chats.add(p1);
            System.out.println("Person 2");
            p2=in.next();
            p2+=in.nextLine();
            chats.add(p2);
        }
        System.out.println("The conversation is ");
        int i=1;
```



```
    for (String chat : chats) {  
        System.out.println("Person" + i + ": " + chat);  
        i=i+1; // i = i + 1  
    }  
    in.close();  
}  
}
```

OUTPUT:

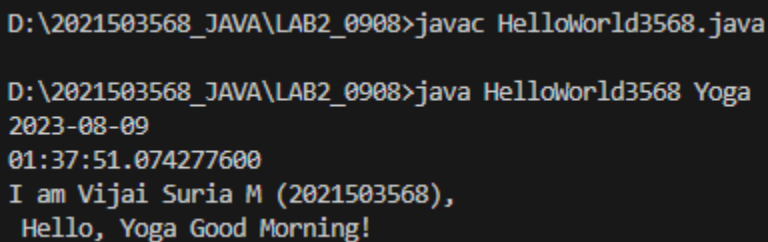
```
PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208>  
Name: Vijai Suria M  
Register Number: 2021503568  
Current Date: 2023-11-09  
Current Time: 2023-11-09  
Chat Application:  
Person 1  
Exception in thread "main"  
PS C:\Users\2021503568\Downloads\LAB1_0208\LAB1_0208>  
Name: Vijai Suria M  
Register Number: 2021503568  
Current Date: 2023-11-09  
Current Time: 2023-11-09  
  
Chat Application:  
Person 1  
Hi  
Person 2  
How are you  
Person 1  
Fine  
Person 2  
Bye  
Person 1  
Text you later  
Person 2  
Bye
```

1.6) Write a java program and compile the code @ command line to execute

CODE:

```
import java.time.LocalDateTime;
class HelloWorld3568 {
    public static void main(String args[]) {
        System.out.println(java.time.LocalDate.now());
        System.out.println(LocalTime.now());
        System.out.print("I am Vijai Suria M (2021503568), \n Hello, ");
        System.out.print(args[0]);
        System.out.println(" Good Morning!");
    }
}
```

Output:



```
D:\2021503568_JAVA\LAB2_0908>javac HelloWorld3568.java
D:\2021503568_JAVA\LAB2_0908>java HelloWorld3568 Yoga
2023-08-09
01:37:51.074277600
I am Vijai Suria M (2021503568),
Hello, Yoga Good Morning!
```

1.7) Write the program to print current date and time

CODE:

```
import java.time.LocalDateTime;
public class Date3568 {
    public static void main(String args[]) {
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568)");
    }
}
```

OUTPUT:

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Date3568.java && java Date3568
Current Date: 2023-08-09
Current Time: 01:48:03.570836
Name: Vijai Suria M
Register Number: (2021503568)
[Done] exited with code=0 in 0.403 seconds
```

1.8) Find the maximum Compile time and Runtime error messages of simple one line output message

CODE:

```
import java.time.LocalDateTime;
class HelloWorld3568 {
    public static void main(String args[]) {
        System.out.println(java.time.LocalDate.now());
        System.out.println(LocalTime.now());
        System.out.print("I am Vijai Suria M (2021503568), \n Hello, ");
        System.out.print(args[0]);
        System.out.println(" Good Morning!");
    }
}
```

OUTPUT:

a) Delete any of the semicolons.

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error3568.java:7: error: ';' expected
    System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568)")
    ^
1 error
[Done] exited with code=1 in 0.269 seconds
```

b) Misspell the word public, static, void, main

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error3568.java:3: error: class, interface, enum, or record expected
Public class Error3568 {
^
1 error

[Done] exited with code=1 in 0.265 seconds
```

c) omit the word public, static, void, main, arg

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Current Date: 2023-08-09
Current Time: 01:55:10.785300600
Name: Vijai Suria M
Register Number: (2021503568)
[Done] exited with code=0 in 0.429 seconds
```

d) Remove the quotation marks around string

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error3568.java:7: error: unclosed string literal
    System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568));
                    ^
1 error
```

e) change the main method argument data type

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error: Main method not found in class Error3568, please define the main method as:
  public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

[Done] exited with code=1 in 0.421 seconds
```

f) omit the argument

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error: Main method not found in class Error3568, please define the main method as:
  public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

[Done] exited with code=1 in 0.434 seconds
```

g) change the argument variable name

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Current Date: 2023-08-09
Current Time: 02:00:00.077779
Name: Vijai Suria M
Register Number: (2021503568)
[Done] exited with code=0 in 0.428 seconds
```

h) Remove the curly braces

```
[Running] cd "d:\2021503568_JAVA\LAB2_0908\" && javac Error3568.java && java Error3568
Error3568.java:3: error: '{' expected
public class Error3568
      ^
Error3568.java:4: error: class, interface, enum, or record expected
    public static void main(String var[]) {
            ^
Error3568.java:6: error: class, interface, enum, or record expected
        System.out.println("Current Time: " + LocalDateTime.now());
        ^
Error3568.java:7: error: class, interface, enum, or record expected
        System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568)");
        ^
Error3568.java:8: error: class, interface, enum, or record expected
    }
    ^
5 errors

[Done] exited with code=1 in 0.269 seconds
```

1.9) Copy the program and compile it. Find the error messages that the compiler finds out. Correct it out and repeat the process until the code runs.

CODE:

```
import java.time.*;
import java.util.Scanner;

public class Bug3568 {
    public static void main (String args[]) {
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
    }
}
```

```

        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        String name;
        Scanner in=new Scanner(System.in);
        System.out.println("Hello. Please type your name: ");
        name = in.next();
        System.out.println("Hello "+name);
        System.out.println ("Have a, nice day!");
        in.close();
    }
}

```

OUTPUT:

```

PS D:\2021503568_JAVA> cd "d:\2021503568_JAVA\LAB2_0908\" ; if ($?) { javac Bug3568.java } ;
Current Date: 2023-08-09
Current Time: 02:18:02.360450900
Name: Vijai Suria M
Register Number: (2021503568)
Hello. Please type your name:
Vijai
Hello Vijai
Have a, nice day!
PS D:\2021503568_JAVA\LAB2_0908> 

```

1.10) Write a program to learn the different Java data types and their correct / incorrect values

CODE:

```

import java.time.*;
public class DataType3568 {
    public static void main(String args[]) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568)");

        byte byteValue1 = 127; // Correct
        byte byteValue2 = 128; // Incorrect: Value exceeds the valid range (-128 to

```

127)

```
    long longValue1 = 9223372036854775807; // Incorrect, require L at the end
    long longValue2 = 9223372036854775807L; // Correct
    long longValue3 = 9223372036854775807; // Incorrect, require L at the end
    long longValue4 = 9223372036854775808L; // Incorrect: Value exceeds the
valid range

    // Floating-point data types
    float floatValue1 = 3.14f; // Correct
    float floatValue2 = 3.14; // Incorrect: Floating-point literals need 'f' or 'F' suffix
    // Boolean data type

    boolean booleanValue1 = true; // Correct
    boolean booleanValue2= 0; // Incorrect: Use 'true' or 'false' for boolean values
}
}
```

OUTPUT:

```
PS D:\2021503568_JAVA\LAB2_0908> javac DataType3568.java
DataType3568.java:11: error: integer number too large
    long longValue1 = 9223372036854775807; // Incorrect, require L at the end
    ^
DataType3568.java:13: error: integer number too large
    long longValue3 = 9223372036854775807; // Incorrect, require L at the end
    ^
DataType3568.java:14: error: integer number too large
    long longValue4 = 9223372036854775808L; // Incorrect: Value exceeds the valid range
    ^
3 errors
```

```
PS D:\2021503568_JAVA\LAB2_0908> javac DataType3568.java
DataType3568.java:10: error: incompatible types: possible lossy conversion from int to byte
    byte byteValue2 = 128; // Incorrect: Value exceeds the valid range (-128 to 127)
    ^
DataType3568.java:18: error: incompatible types: possible lossy conversion from double to float
    float floatValue2 = 3.14; // Incorrect: Floating-point literals need 'f' or 'F' suffix
    ^
DataType3568.java:22: error: incompatible types: int cannot be converted to boolean
    boolean booleanValue2= 0; // Incorrect: Use 'true' or 'false' for boolean values
    ^
3 errors
```

1.11) Write a program that takes as input Fahrenheit temperature. It converts the input temperature to Celsius and prints out the converted temperature as shown in the example. The formula for conversion between the two is: $C = 5/9(F - 32)$, Where C is the temperature in Celsius and F is the temperature in Fahrenheit

CODE:

```
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.time.*;
import java.util.Scanner;

public class F2C3568 {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        float f, c;
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.print("Name: Vijai Suria M \nRegister Number: (2021503568)");

        System.out.println("\n Enter your choice \n 1) Fahrenheit to Celsius \n 2)
Celsius to Fahrenheit ");
        int choice = in.nextInt();
        DecimalFormat decimalFormat = new DecimalFormat("0.00");
        switch (choice) {
            case 1:
                System.out.print("Temperature in Fahrenheit: ");
                f = in.nextFloat();
                c = (f - 32) * 5 / 9;
                System.out.println("Equivalent Temperature in Celsius: " +
decimalFormat.format(c));
                decimalFormat.setRoundingMode(RoundingMode.DOWN);
                break;
            case 2:
                System.out.print("Temperature in Celsius: ");
                c = in.nextFloat();
                f = (9 * c / 5) + 32;
                System.out.println("Equivalent Temperature in Fahrenheit: "+
decimalFormat.format(f));
```



```

        decimalFormat.setRoundingMode(RoundingMode.DOWN);
        break;
    default:
        System.out.println("\n Please, Enter the valid choice.....");
    }
    in.close();
}
}

```

OUTPUT:

```

PS D:\2021503568_JAVA\LAB2_0908> cd "d:\2021503568_JAVA\LAB2_0908\" ;
Current Date: 2023-08-09
Current Time: 15:18:14.530804900
Name: Vijai Suria M
Register Number: (2021503568)
Enter your choice
1) Fahrenheit to Celsius
2) Celsius to Fahrenheit
1
Temperature in Fahrenheit: 0.555
Equivalent Temperature in Celsius: -17.47

```

```

PS D:\2021503568_JAVA\LAB2_0908> cd "d:\2021503568_JAVA\LAB2_0908\" ;
Current Date: 2023-08-09
Name: Vijai Suria M
Register Number: (2021503568)
Enter your choice
1) Fahrenheit to Celsius
2) Celsius to Fahrenheit
1
Temperature in Fahrenheit: 212
Equivalent Temperature in Celsius: 100.00

```

```

Temperature in Fahrenheit: 212
Equivalent Temperature in Celsius: Exception in thread "main" java.util.IllegalFormatConversionException: f != java.lang.String
    at java.base/java.util.Formatter$FormatSpecifier.failConversion(Formatter.java:4522)
    at java.base/java.util.Formatter$FormatSpecifier.printFloat(Formatter.java:3056)
    at java.base/java.util.Formatter$FormatSpecifier.print(Formatter.java:3004)
    at java.base/java.util.Formatter.format(Formatter.java:2769)
    at java.base/java.io.PrintStream.format(PrintStream.java:1222)
    at java.base/java.io.PrintStream.printf(PrintStream.java:1118)
    at F2C3568.main(F2C3568.java:21)
PS D:\2021503568_JAVA\LAB2_0908> 

```

1.12) Write a program that accepts the three numbers u, a, and t as input. Here, u denotes the starting speed, the acceleration, and t the amount of time. The program outputs the displacement covered (d) in time (t). The program prints the final velocity (v). $v=u+at$. Since velocity and acceleration are continuous vectors (in physics), u and a can have any real value. Only non-negative real values, or 0 t, can be assigned to time t, i.e., $0 \leq t$. Note: round your answer to up to two decimal places.

CODE:

```
import java.time.LocalDateTime;
import java.util.Scanner;

public class FinalVelocity3568 {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        System.out.print("Enter the Starting Speed u=");
        double u = in.nextDouble();
        System.out.print("Enter the Acceleration a=");
        double a = in.nextDouble();
        System.out.print("Enter the Time t=");
        double t = in.nextDouble();
        while(t<0){
            System.out.print("(Invalid Time Value, Enter the Time t=");
            t = in.nextDouble();
        }
        double v = u + a*t;
        System.out.println("Value of Velocity v=" + v);
        in.close();
    }
}
```

OUTPUT:

```
PS D:\2021503568_JAVA\LAB2_0908> cd "d:\2021503568_JAVA\LAB2_0908\" ;
Current Date: 2023-08-09
Current Time: 15:34:39.078593900
Name: Vijai Suria M
Register Number: (2021503568)
Enter the Starting Speed u=20.0
Enter the Acceleration a=15
Enter the Time t=2
Value of Velocity v=50.0
```

```
PS D:\2021503568_JAVA\LAB2_0908> cd "d:\2021503568_JAVA\LAB2_0908\" ;
Current Date: 2023-08-09
Current Time: 15:35:12.795577300
Name: Vijai Suria M
Register Number: (2021503568)
Enter the Starting Speed u=20
Enter the Acceleration a=12
Enter the Time t=-2
(Invalid Time Value, Enter the Time t=-9
(Invalid Time Value, Enter the Time t=0
Value of Velocity v=20.0
```

Result: Thus, basic concepts in Java programming were explored successfully. And thus, outputs were verified.

CONTROL STRUCTURES

Aim: To implement basic concepts of Control Structures in Java

Algorithm:

Step 1: Custom Grade

Methods used:

- a) **System.out.println(String)** - Print a string to the standard output.
- b) **LocalDate.now()** - To get the current date.
- c) **Scanner in = new Scanner(System.in)** - To create a Scanner object to read user input from the standard input.
- d) **in.nextInt()** - To read and return the next integer from the input.
- e) **int[] arr = new int[n]** - To initialize an integer array to store marks for each subject.

Step 2: Coin toss

Methods used:

- a) **Random rand = new Random()** - To create a random number generator.
- b) **rand.nextInt(int)** - To generate a random integer within the specified range.
- c) **in.next().charAt(int)** - To read the next token as a string and get the character at the specified index.

Step 3: No of days in a month

Methods used:

- a) **isLeapYear(int)** – To check if the given year is leap year or not
- b) **noOfDays(int)** – To get the no of days in a month

Step 4: Verbalize numbers

Methods used:

- a) **convertToWords(int)** – To convert a number to its verbal representation, and it uses recursion to handle hundreds and thousands.

Step 5: 24-hour digital watch

Methods used:

- a) **isValidDay(String day)** - To check if the given day is valid.
- b) **getNextDay(String day)** - To get the next day based on the input day.

2.1) Write a program to determine the custom grade of the marks given. If the marks, go beyond 100 or lower than 0 then state the input is invalid.

90 to 100: Grade O

80 TO 89: Grade A+

70 TO 79: Grade A

60 TO 69: Grade B+

50 to 59: Grade B

0 TO 49: Grade U Hint: use If statement.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
public class Grade3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the number of Subject: ");
        int n = in.nextInt();
        int[] marks = new int[n];
        System.out.println("Enter the Subject marks (space seperated): ");
        for(int i=0;i<n;i++){
            marks[i] = in.nextInt();
        }
        System.out.println("Grade Reports:");
        for(int i=0;i<n;i++){
            if(marks[i]>=90 && marks[i]<=100)
                System.out.println("Subject-"+ (i+1)+" Grade: O");
            else if(marks[i]>=80 && marks[i]<=89)
                System.out.println("Subject-"+ (i+1)+" Grade: A+");
            else if(marks[i]>=70 && marks[i]<=79)
                System.out.println("Subject-"+ (i+1)+" Grade: A");
```

```

        else if(marks[i]>=60 && marks[i]<=69)
            System.out.println("Subject-"+ (i+1)+" Grade: B+");
        else if(marks[i]>=50 && marks[i]<=59)
            System.out.println("Subject-"+ (i+1)+" Grade: B");
        else if(marks[i]>=0 && marks[i]<=49)
            System.out.println("Subject-"+ (i+1)+" Grade: U");
        else
            System.out.println("Subject-"+ (i+1)+" Invalid Marks");
    }
    in.close();
}
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\"
Current Date: 2023-08-16
Current Time: 01:01:16.482710700
Name: Vijai Suria M
Register Number: (2021503568)
Enter the number of Subject: 5
Enter the Subject marks (space seperated):
89 91 76 -5 52
Grade Reports:
Subject-1 Grade: A+
Subject-2 Grade: O
Subject-3 Grade: A
Subject-4 Invalid Marks
Subject-5 Grade: B

```

2.2) Write a program that simulates the coin toss as head or tail. Use Random number 0 Or 1 to determine the system input. The program should print the result as head if it one and tail if it is zero. Read the input from the user as a character 'h' or 't' or 'H' or 'T' and tell the user whether he or she has predicted the coin toss correctly. Declare the result as Won the toss! Or lose! (Hint: Use Random class to get the random input 0 to 1).

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Random;
import java.util.Scanner;
public class Toss3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);
        Random random = new Random();
        int res = random.nextInt(2);
        System.out.print("Enter your choice (h/t) or (H/T): ");
        char ch=in.next().charAt(0);
        if(ch=='h' || ch=='H'){
            if(res==1)
                System.out.println("Won the toss!");
            else
                System.out.println("Loss the toss!");
        }
        else{
            if(res==0)
                System.out.println("Won the toss!");
            else
                System.out.println("Loss the toss!");
        }
        System.out.println(res);
        in.close();
    }
}
```


OUTPUT:

```
PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\"
Current Date: 2023-08-16
Current Time: 01:18:41.569796800
Name: Vijai Suria M
Register Number: (2021503568)
Enter your choice (h/t) or (H/T): t
Won the toss!
```

```
PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\" ;
Current Date: 2023-08-16
Current Time: 01:18:36.785022600
Name: Vijai Suria M
Register Number: (2021503568)
Enter your choice (h/t) or (H/T): t
Loss the toss!
```

2.3) Write a program to output the number of days of the month(1 to 12) in the given year. The value of the month February is 28 or 20 based on the leap year. Give warning for invalid input that is of digit less than 4 or greater than 5. Hint: Use Switch statement.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
import java.time.Year;

public class Days3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the month(1-12): ");
        int month = in.nextInt();
        if(month<1 || month>12){
            System.out.println("Invalid month");
            System.exit(0);
        }
    }
}
```

```

}
System.out.print("Enter the year: ");
int year = in.nextInt();
if(year<=999 || year>9999){
    System.out.println("Invalid year");
    System.exit(0);
}
switch(month){
    case 1:
        System.out.println("The number of days in the month (January) is "+31);
        break;
    case 2:
        Year y = Year.of(year);
        if(y.isLeap())
            System.out.println("The number of days in the month (Februaury) is "+29);
        else
            System.out.println("The number of days in the month (Februaury) is "+28);
        break;
    case 3:
        System.out.println("The number of days in the month (March) is "+31);
        break;
    case 4:
        System.out.println("The number of days in the month (April) is "+30);
        break;
    case 5:
        System.out.println("The number of days in the month (May) is "+31);
        break;
    case 6:
        System.out.println("The number of days in the month (June) is "+30);
        break;
    case 7:
        System.out.println("The number of days in the month (July) is "+31);
        break;
    case 8:
        System.out.println("The number of days in the month (August) is "+31);
        break;
    case 9:
        System.out.println("The number of days in the month (September) is "+30);

```

```

        break;
    case 10:
        System.out.println("The number of days in the month (October) is "+31);
        break;
    case 11:
        System.out.println("The number of days in the month (November) is "+30);
        break;
    case 12:
        System.out.println("The number of days in the month (December) is "+31);
        break;
    default:
        System.out.println("Invalid Month");
        break;
    }
    in.close();
}
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\"
Current Date: 2023-08-16
Current Time: 02:08:59.806440500
Name: Vijai Suria M
Register Number: (2021503568)
Enter the month(1-12): 6
Enter the year: 2004
The number of days in the month (June) is 30

```

2.4) Write a program that verbalize the user inputs between 1 and 9999 number. For example : Input number : 852 => Output: Eight hundred and fifty two.

CODE:

```
import java.util.Scanner;

public class Verbalize3568 {
    private static final String[] units = {"", "One", "Two", "Three", "Four", "Five", "Six",
"Seven", "Eight", "Nine"};
    private static final String[] teens = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen",
"Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};
    private static final String[] tens = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty",
"Seventy", "Eighty", "Ninety"};
    private static final String[] thousands = {"", "Thousand", "Million", "Billion"};
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Input number: ");
        int number = scanner.nextInt();

        if (number < 1 || number > 9999) {
            System.out.println("Number out of range.");
        } else {
            String verbalized = verbalizeNumber(number);
            System.out.println(verbalized);
        }
        scanner.close();
    }

    private static String verbalizeNumber(int number) {
        if (number == 0) {
            return "Zero";
        }
        String verbalized = "";
        int thousandsIndex = 0;

        while (number > 0) {
            if (number % 1000 != 0) {
                verbalized = verbalizeChunk(number % 1000) + thousands[thousandsIndex] +
" " + verbalized;
            }
            number /= 1000;
            thousandsIndex++;
        }
    }

    private static String verbalizeChunk(int number) {
        if (number < 10) {
            return units[number];
        } else if (number < 20) {
            return teens[number - 10];
        } else {
            return tens[number / 10] + (number % 10 != 0 ? verbalizeNumber(number % 10) : "");
        }
    }
}
```

```

    }
    number /= 1000;
    thousandsIndex++;
}
return verbalized.trim();
}

private static String verbalizeChunk(int number) {
    if (number == 0) {
        return "";
    }
    if (number < 10) {
        return units[number];
    } else if (number < 20) {
        return teens[number - 10];
    } else if (number < 100) {
        return tens[number / 10] + " " + units[number % 10];
    } else {
        return units[number / 100] + " Hundred " + verbalizeChunk(number % 100);
    }
}
}
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\"
Current Date: 2023-08-16
Current Time: 02:00:19.260870200
Name: Vijai Suria M
Register Number: (2021503568)
Input number: 852
Eight Hundred Fifty Two
PS Z:\2021503568_JAVA\LAB4_1608>

```

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\"
Current Date: 2023-08-16
Current Time: 02:00:04.972078200
Name: Vijai Suria M
Register Number: (2021503568)
Input number: 1234
OneThousand Two Hundred Thirty Four
PS Z:\2021503568_JAVA\LAB4_1608>

```

2.5) Write a program to create 24-hour digital watch. Get the input from the user for the day(Mon, Tue, Wed, Thu, Fri, Sat, Sun) , hour (24-hour) , minute and seconds . The input of the user is valid then the digital clock should advance the input by one second and display the new day, hour, minute and second.

Sample input:

Input day : Mon

Input hour : 23

Input minute : 59

Input second : 59

Sample output:

Tue 0 0 0

Hint: Use for statement

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class Time3568 {
    public static String nextDay(String day)
    {
        if(day.equals("Sun"))
            return "Mon";
        if(day.equals("Mon"))
            return "Tue";
        if(day.equals("Tue"))
            return "Wed";
        if(day.equals("Wed"))
            return "Thu";
        if(day.equals("Thu"))
            return "Fri";
        if(day.equals("Fri"))
```

```

        return "Sat";

        return "Sun";

    }

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");

        String day;
        int hour,minute,sec;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter day:");
        day=in.nextLine();
        System.out.print("Enter hour:");
        hour = in.nextInt();
        System.out.print("Enter minute:");
        minute = in.nextInt();
        System.out.print("Enter second:");
        sec = in.nextInt();

        if(sec+1 <=59)
            sec+=1;
        else{
            sec =0;
            if(minute+1 <= 59)
                minute++;
            else
            {
                minute =0;
                if(hour+1<=23)
                    hour++;
                else
                {
                    hour = 0;
                    day=nextDay(day);
                }
            }
        }
    }
}

```

```

    }
    }
}
System.out.println(day+" "+hour+" "+minute+" "+sec);
in.close();
}
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\" ;
Current Date: 2023-08-16
Current Time: 02:32:38.265161100
Name: Vijai Suria M
Register Number: (2021503568)
Enter day:Mon
Enter hour:23
Enter minute:59
Enter second:59
Tue 0 0 0

```

```

PS Z:\2021503568_JAVA\LAB4_1608> cd "z:\2021503568_JAVA\LAB4_1608\" ;
Current Date: 2023-08-16
Current Time: 02:39:39.046738
Name: Vijai Suria M
Register Number: (2021503568)
Enter day:Thu
Enter hour:12
Enter minute:58
Enter second:59
Thu 12 59 0

```


Result: Thus, basic concepts of Control Structures in java were implemented successfully. And thus, outputs were verified.

CONSTRUCTOR AND METHODS

Aim: To implement basic concepts of constructors and methods in Java

Algorithm:

Step 1: Circle class

Methods used:

- a) **System.out.println(String)** - To print a string to the standard output.
- b) **LocalDate.now()** - To get the current date.
- c) **LocalTime.now()** - To get the current time.
- d) **Math.PI** - To get the mathematical constant pi (π).
- e) **Circle(double x, double y, double r)** - Constructor to initialize the properties of a circle with a specified center (x, y) and radius (r).
- f) **display()** - To display the center and radius of a circle.
- g) **calculateArea()** - To calculate the area of a circle.
- h) **static String compareArea(Circle c1, Circle c2)** - To compare the areas of two circles (c1 and c2) and return a comparison result as a string.

Step 2: Usage of default constructor, parameterized constructor, this keyword and returning object

Methods used:

- a) **Student()** - Default constructor that initializes a student with default values (id = 0, name = "Unknown").
- b) **Student(int id, String name)** - Parameterized constructor that initializes a student with specified values for id and name.
- c) **display()** - To display the student's ID and name.
- d) **static Student createStudent(int id, String name)** - To return a new Student object with the specified id and name.
- e) **void updateStudent(Student newStudent)** - To update the student's information with the values from another student object.

Step 3: Count the number of instances

Methods used:

- a) **Counter()** - Constructor that increments the static instance count for each new instance and initializes the non-static instance count for each instance.

- b) **displayCounts()** - To display the total number of instances created and the non-static instance count for a specific instance.

Step 4: Method overloading

Methods used:

- a) **int sum(int x, int y)** - Method to sum two integers.
- b) **int sum(int x, int y, int z)** - Method to sum three integers.
- c) **double sum(double x, double y)** - Method to sum two doubles.
- d) **public Data(double value)** - Constructor to initialize a "Data" object with a specified value.

Step 5: Difference of Instance variables, Instance methods, static variable, static methods

Methods used:

- a) **Dummy(int instanceVar)** - Constructor to initialize the instance variable with a specified value.
- b) **void instanceMethod()** - To display information about the instance variable and call the static method.
- c) **static void staticMethod()** - To display information about the static variable.

Step 6: Immutable class Person

Methods used:

- a) **Person(String name, int age)** - Constructor to initialize the name and age of the person.
- b) **String getName()** - To get the name of the person.
- c) **int getAge()** - To get the age of the person.

Step 7: Java class Clock

Methods used:

- a) **setClock(int hours, int minutes, int seconds)** - To set the time values while checking for validity.
- b) **getHours()** - To get the hours of the time.
- c) **getMinutes()** - To get the minutes of the time.
- d) **getSeconds()** - To get the seconds of the time.
- e) **setHours(int hours)** - To set the hours while checking for validity.
- f) **setMinutes(int minutes)** - To set the minutes while checking for validity.
- g) **setSeconds(int seconds)** - To set the seconds while checking for validity.
- h) **tick()** - To increment the time by one second.

- i) **tickDown()** - To decrement the time by one second.
- j) **addClock(Clock other)** - To add time from another Clock instance.
- k) **subtractClock(Clock other)** - To subtract time from another Clock instance and return the difference as a new Clock.

3.1) Write a program to create a class circle with centre and the radius as instance variables.

Initialize and display its variables.

Modify the exercise to have a constructor in class circle to initialize its variables.

Modify the exercise to define the instance method calculateArea() to calculate the area and a static method compareArea() to compare the area of the circle and declare the result as smaller than or larger than or equal

CODE:

```
class Circle {
    double centerX;
    double centerY;
    double radius;

    public Circle(double centerX, double centerY, double radius) {
        this.centerX = centerX;
        this.centerY = centerY;
        this.radius = radius;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    public static String compareArea(Circle circle1, Circle circle2) {
        double area1 = circle1.calculateArea();
        double area2 = circle2.calculateArea();
        if (area1 < area2) {
            return "Circle 1 has a smaller area than Circle 2.";
        } else if (area1 > area2) {
            return "Circle 1 has a larger area than Circle 2.";
        }
    }
}
```

```

        } else {
            return "Circle 1 and Circle 2 have equal areas.";
        }
    }
}

public class CircleArea3568 {
    public static void main(String[] args) {
        Circle circle1 = new Circle(0, 0, 5);
        Circle circle2 = new Circle(3, 4, 7);

        System.out.println("Circle 1 - Center: (" + circle1.centerX + ", " + circle1.centerY +
            "), Radius: " + circle1.radius);
        System.out.println("Circle 2 - Center: (" + circle2.centerX + ", " + circle2.centerY +
            "), Radius: " + circle2.radius);

        System.out.println("Area of Circle 1: " + circle1.calculateArea());
        System.out.println("Area of Circle 2: " + circle2.calculateArea());

        System.out.println(Circle.compareArea(circle1, circle2));
    }
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB6_3008> cd "z:\2021503568_JAVA\LAB6_3008\" ;
Circle 1 - Center: (0.0, 0.0), Radius: 5.0
Circle 2 - Center: (3.0, 4.0), Radius: 7.0
Area of Circle 1: 78.53981633974483
Area of Circle 2: 153.93804002589985
Circle 1 has a smaller area than Circle 2.

```

3.2) Write a program to display the use of

- this keyword.
- Default constructor
- Parameterized constructor
- Pass Object as an argument
- Return object

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
class Car {
    public String myCar;
    Car(){
        this.myCar="";
        System.out.println("car object initialized with default constructor " + myCar);
    }
    Car(String myCar){
        this.myCar=myCar;
        System.out.println("car object initialized with parameterised constructor " + myCar);
    }
}
public class Question2 {
    public static Car mergeObject(Car obj1, Car obj2){
        Car result = new Car();
        result.myCar = obj1.myCar + " " + obj2.myCar;
        return result;
    }
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Car obj1 = new Car("Audi");
        Car obj2 = new Car("BMW");
        Car result = mergeObject(obj1,obj2);
        System.out.println(result.myCar);
    }
}
```

OUTPUT:

```
PS Z:\2021503568_JAVA\LAB6_3008> cd "z:\2021503568_JAVA\LAB6_3008\" ;  
Current Date: 2023-08-30  
Current Time: 14:55:29.994621500  
Name: Vijai Suria M  
Register Number: (2021503568)  
car object initialized with parameterised constructor Audi  
car object initialized with parameterised constructor BMW  
car object initialized with default constructor  
Audi BMW
```

3.3) Write a program to count the number of instances created for the class using static variable and the non static variable not visible to all the instances.

CODE:

```
import java.time.LocalDate;  
import java.time.LocalTime;  
  
class InstanceCounter {  
    private static int instanceCount = 0;  
    private int nonStaticCount = 0;  
  
    public InstanceCounter() {  
        instanceCount++;  
        nonStaticCount++;  
    }  
  
    public static int getInstanceCount() {  
        return instanceCount;  
    }  
  
    public int getNonStaticCount() {  
        return nonStaticCount;  
    }  
}
```



```

public class InstanceCounter3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        InstanceCounter obj1 = new InstanceCounter();
        InstanceCounter obj2 = new InstanceCounter();
        InstanceCounter obj3 = new InstanceCounter();

        System.out.println("Total instances created: " +
InstanceCounter.getInstanceCount());
        System.out.println("Non-static count for obj1: " + obj1.getNonStaticCount());
        System.out.println("Non-static count for obj2: " + obj2.getNonStaticCount());
        System.out.println("Non-static count for obj3: " + obj3.getNonStaticCount());
    }
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB6_3008> cd "z:\2021503568_JAVA\LAB6_3008\"
Current Date: 2023-08-30
Current Time: 15:03:23.163135500
Name: Vijai Suria M
Register Number: (2021503568)
Total instances created: 3
Non-static count for obj1: 1
Non-static count for obj2: 1
Non-static count for obj3: 1

```

3.4) Write a program that implements method overloading(multiple methods in the same class can have the same name but different parameter lists) based on the following conditions

- By changing number of arguments
- By changing the data type of the arguments
- Passing object as parameter.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
class Car {
    public String myCar;
    Car(){
        this.myCar="null";
        System.out.println("car object initialized with default constructor ");
    }
    Car(String myCar){
        this.myCar=myCar;
        System.out.println("car object initialized with parameterised constructor ");
    }
    public String getMyCar(){
        return myCar;
    }
    public String getMyCar(String name){
        return "Your name: "+ name + "\n Your car name:" + myCar;
    }
    public String getMyCar(String name,int id){
        return "Your name and ID: "+ name + "\t" + id + "\n Your car name:" +
myCar;
    }
    public String getMyCar(int id){
        return "Your Id: "+ id + "\n Your car name:" + myCar;
    }
}
public class Overloading3568 {
    public static Car mergeObject(Car obj1, Car obj2){
        Car result = new Car();
        result.myCar = obj1.myCar + " " + obj2.myCar;
        return result;
    }
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Car obj1 = new Car("Audi");
```

```

        Car obj2 = new Car("BMW");
        Car result = mergeObject(obj1,obj2);
        System.out.println(result.getMyCar("Vijai"));
        System.out.println(result.getMyCar("Vijai",2021503568));
        System.out.println(result.getMyCar(2021503568));
        System.out.println(result.getMyCar());
    }
}

```

OUTPUT:

- By changing number of arguments

```

public String getMyCar(){
    return myCar;
}
public String getMyCar(String name){
    return "Your name: " + name + "\n Your car name:" + myCar;
}
public String getMyCar(String name,int id){
    return "Your name: " + name + "\n Your car name:" + myCar;
}
public String getMyCar(int id){
    return "Your Id: " + id + "\n Your car name:" + myCar;
}

```

- By changing the data type of the arguments

```

class Car {
    public String myCar;
    Car(){
        this.myCar="null";
        System.out.println(x:"car object initialized with default constructor ");
    }
    Car(String myCar){
        this.myCar=myCar;
        System.out.println(x:"car object initialized with parameterised constructor ");
    }
    public String getMyCar(){
        return myCar;
    }
    public String getMyCar(String name){
        return "Your name: " + name + "\n Your car name:" + myCar;
    }
    public String getMyCar(int id){
        return "Your Id: " + id + "\n Your car name:" + myCar;
    }
}

```

- Passing object as parameter.

```
public static Car mergeObject(Car obj1, Car obj2){  
    Car result = new Car();  
    result.myCar = obj1.myCar + " " + obj2.myCar;  
    return result;  
}
```

- Show that the method overloading is not possible by just changing the return type

```
public String getMyCar(String name){  
    return "Your name: " + name + "\n Your car name:" + myCar;  
}  
public String getMyCar(int id){  
    return "Your Id: " + id + "\n Your car name:" + myCar;  
}  
public int getMyCar(String name){  
    return 1;  
}
```

- Can we overload java main method?
No, it is not possible.

OUTPUT:

```
Current Date: 2023-09-17  
Current Time: 22:43:56.285508300  
Name: Vijai Suria M  
Register Number: (2021503568)  
car object initialized with parameterised constructor  
car object initialized with parameterised constructor  
car object initialized with default constructor  
Your name: Vijai  
Your car name:Audi BMW  
Your name and ID: Vijai 2021503568  
Your car name:Audi BMW  
Your Id: 2021503568  
Your car name:Audi BMW  
Audi BMW
```

3.5) Write a program that show the differences of Instance variables, Instance methods, static variable, static methods

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
class Example {
    int instanceVar;
    static int staticVar;
    void instanceMethod() {
        System.out.println("This is an instance method.");
        System.out.println("Instance variable value: " + instanceVar);
        System.out.println("Static variable value: " + staticVar);
    }
    static void staticMethod() {
        System.out.println("This is a static method.");
        // Instance variables cannot be accessed directly in a static method.
        // System.out.println("Instance variable value: " + instanceVar); // This will cause an
error.
        System.out.println("Static variable value: " + staticVar);
    }
}
public class Question5 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Example.staticVar = 100;
        Example.staticMethod();
        Example obj1 = new Example();
        obj1.instanceVar = 42;
        obj1.instanceMethod();
        Example obj2 = new Example();
        obj2.instanceVar = 24;
        obj2.instanceMethod();
    }
}
```

OUTPUT:

```
PS Z:\2021503568_JAVA\LAB6_3008> cd "z:\2021503568_JAVA\LAB6_3008\"
Current Date: 2023-08-30
Current Time: 15:10:10.847719500
Name: Vijai Suria M
Register Number: (2021503568)
This is a static method.
Static variable value: 100
This is an instance method.
Instance variable value: 42
Static variable value: 100
This is an instance method.
Instance variable value: 24
Static variable value: 100
```

6) Write a program to create an immutable class Person (state cannot be changed)

- Define private final fields of name and age
- Define a constructor to set the fields and a getter method to display the values.
- Show that the state cannot be changed and enhances robustness

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;

final class Person {
    private final String name;
    private final int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }
}
```

```

    public int getAge() {
        return age;
    }
}

public class Question6 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Person person = new Person("John", 30);

        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());
    }
}

```

OUTPUT:

```

PS Z:\2021503568_JAVA\LAB6_3008> cd "z:\2021503568_JAVA\LAB6_3008\" ;
Current Date: 2023-08-30
Current Time: 15:13:35.120302300
Name: Vijai Suria M
Register Number: (2021503568)
Name: John
Age: 30

```

3.7) Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:

Three instance variables for the hours (range 0-23), minutes(range 0-59),and seconds(range 0-59).

Three constructors:

- default(with no parameters passed; initialize the represented time to12:0:0)
- a constructor with three parameters: hours, minutes, and seconds.
- a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes,and seconds)

Instance methods:

- A set-method method setClock() with one parameter seconds since midnight (to be converted into the time value in hours, minutes, and seconds as above).
- get-methods : getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values.
- set-methods : setHours(), setMinutes(), setSeconds() with one parameter each that setup the corresponding instance variables.
- method tick() with no parameters that increments the time stored in a Clock object by one second.
- method addClock() accepting an object of type Clock as a parameter.The method should add the time represented by the parameter class to the time represented in the current class.
- Add an instance method tickDown() which decrements the time stored in a Clock object by one second.
- Add an instance method subtractClock() that takes one Clock parameter and returns the difference between the time represented in the current

Clock object and the one represented by the Clock parameter. Difference of time should be returned as an clock object.

Write a separate class Clock Demo with a main() method. The program should:

- Instantiate a Clock object first Clock using one integer seconds since midnight obtained from the keyboard.
- Print both clock object

Create a reference thirdClock that should reference to object of difference of first Clock and second Clock by calling the method subtractClock()

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

class Clock{
    private int hour;
    private int minute;
    private int second;
    Clock(){
        hour = 12;
        minute = 0;
        second = 0;
    }
    Clock(int h, int m, int s){
        hour = h;
        minute = m;
        second = s;
    }
    Clock(int secondsFromMindnight){
        hour = secondsFromMindnight / 3600;
        minute = (secondsFromMindnight % 3600) / 60;
        second = (secondsFromMindnight % 3600) % 60;
    }
    public void setClock(int secondsFromMindnight){
        hour = secondsFromMindnight / 3600;
        minute = (secondsFromMindnight % 3600) / 60;
        second = (secondsFromMindnight % 3600) % 60;
    }
    public int getHours(){
        return hour;
    }
    public int getMinutes(){
        return minute;
    }
    public int getSeconds(){
        return second;
    }
}
```

```

public void setHours(int h){
    hour = h;
}
public void setMinutes(int m){
    minute = m;
}
public void setSeconds(int s){
    second = s;
}
public void tick(){
    second++;
    if(second == 60){
        second = 0;
        minute++;
        if(minute == 60){
            minute = 0;
            hour++;
            if(hour == 24){
                hour = 0;
            }
        }
    }
}
public void tickDown(){
    second--;
    if(second == -1){
        second = 59;
        minute--;
        if(minute == -1){
            minute = 59;
            hour--;
            if(hour == -1){
                hour = 23;
            }
        }
    }
}
public Clock addClock(Clock A){

```

```

    int s = this.second + A.getSeconds();
    int m = this.minute + A.getMinutes();
    int h = this.hour + A.getHours();
    while(s >= 60){
        s-=60;
        m++;
    }
    while(m >= 60){
        m-=60;
        h++;
    }
    if(h>=24)
        h=0;
    Clock res = new Clock(h,m,s);
    return res;
}

public Clock subtractClock(Clock A){
    int h = hour - A.getHours();
    int m = minute - A.getMinutes();
    int s = second - A.getSeconds();
    if(s < 0){
        s += 60;
        m--;
    }
    if(m < 0){
        m += 60;
        h--;
    }
    if(h < 0){
        h += 24;
    }
    Clock res = new Clock(h, m, s);
    return res;
}

public void show(){
    System.out.println(hour + ":" + minute + ":" + second);
}
}

```

```

public class ClockDemo3568 {
    public static void main(String args[]){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of seconds from midnight: ");
        int secondsFromMindnight = in.nextInt();
        Clock b = new Clock(secondsFromMindnight);
        Clock a = new Clock();
        System.out.print("Clock 1:");
        a.show();
        System.out.print("Clock 2:");
        b.show();
        Clock c = a.subtractClock(b);
        System.out.print("Clock 3 => Clock_1 - Clock_2 = ");
        c.show();
        in.close();
    }
}

```

OUTPUT:

```

PS C:\Users\vijai\Documents\ASSIGNMENTS\JAVA-PROGRAMMING\LAB6_3008>
?) { javac ClockDemo3568.java } ; if ($?) { java ClockDemo3568 }
Current Date: 2023-09-12
Current Time: 12:26:27.235196600
Name: Vijai Suria M
Register Number: (2021503568)
Enter the number of seconds from midnight:
3600
Clock 1:12:0:0
Clock 2:1:0:0
Clock 3 => Clock_1 - Clock_2 = 11:0:0

```

Result: Thus, basic concepts of Constructors and Methods in java were implemented successfully. And thus, outputs were verified.

ARRAYS

Aim: To implement basic concepts of arrays in Java

Algorithm:

Step 1: To displaying the sorted array format. Include $O(n^2)$ and $O(n)$ complexity sorting algorithms and print the comparisons for each.

Methods used:

- a) **System.out.println(String)** - Print a string to the standard output.
- b) **LocalDate.now()** - To get the current date.
- c) **LocalTime.now()** - To get the current time.
- d) **Scanner in = new Scanner(System.in)** - To create a Scanner object to read user input from the standard input.
- e) **in.nextInt()** - To read and return the next integer from the input.
- f) **in.nextLine()** - To consume the newline character left in the input buffer.
- g) **int[] arr = new int[n]** - To initialize an integer array to store marks for each subject.
- h) **bubbleSort(int[] arr)** - Sorts an array in ascending order using the Bubble Sort algorithm and returns the number of comparisons.
- i) **selectionSort(int[] arr)** - Sorts an array in ascending order using the Selection Sort algorithm and returns the number of comparisons.
- j) **printArray(int[] arr)** - Prints the elements of an array in a formatted manner.

Step 2: To create a Java program with static methods to read and sort n random integers, returning comparisons. Also, generate and sort a character array from the random integers.

Methods used:

- a) **RandomArray(int[] arr)**- Generates an array of random integers between 0 and 25..
- b) **Random rand = new Random()** - To create a random number generator.
- c) **rand.nextInt(int)** - To generate a random integer within the specified range.
- d) **Scanner in = a Scanner(System.in)** - To create a Scanner object to read user input from the standard input.
- e) **CharArray(int[] intArray)** - Converts an integer array to a character array, mapping integers to corresponding lowercase English letters.
- f) **OrderNSort(char[] arr)** - Sorts a character array in ascending order using the Selection Sort algorithm and returns the number of comparisons.
- g) **OrderNSort(int[] arr)** - Sorts an integer array in ascending order using the Selection Sort algorithm and returns the number of comparisons.
- h) **printArray(int[] arr)** - Prints the elements of an integer array in a formatted manner with vertical bars.
- i) **printArray(char[] arr)** - Prints the elements of a character array in a formatted manner with horizontal bars.

Step 3: To write Java program to read n random integers into arrays A and B, both of size n. Use a method to search and print the occurrences of each element from B in A.

Methods used:

- a) **Random rand = new Random()** - To create a random number generator.
- b) **rand.nextInt(int)** - To generate a random integer within the specified range.
- c) **countOccurrences(int[] arr, int target)** - Counts the occurrences of a specified target integer in an array and returns the count.

Step 4: To calculates column-wise sums using a method, and sorts the arrays based on the column sums.

Methods used:

- a) **Scanner scanner = new Scanner(System.in)** - To create a Scanner object to read user input from the standard input.

- b) **scanner.nextInt()** - To read and return the next integer from the input.
- c) **readMatrix(Scanner scanner, int[][] matrix)** - Reads input elements for a matrix from the user using a scanner.
- d) **addMatrices(int[][] matrix1, int[][] matrix2)** - Adds two matrices and returns the result.
- e) **sortMatrixByColumnSum(int[][] matrix)** - Sorts the matrix by the sum of each column in ascending order.
- f) **printMatrix(int[][] matrix)** - Prints the elements of a matrix in a formatted manner.

4.1) Sort Write a program to read n integer in a 1D array and print the sorted array in the following format. Use static methods and find the number of comparisons for the sorting algorithm whose worst-case complexity is $O(n^2)$ and $O(n)$

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class Sort3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter no of elements: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the array elements: ");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
    }
}
```

```

int comparisonsBubbleSort = bubbleSort(arr.clone());
int comparisonsSelectionSort = selectionSort(arr.clone());

    System.out.println("Number of comparisons for Bubble Sort: " +
comparisonsBubbleSort);
    System.out.println("Number of comparisons for Selection Sort: " +
comparisonsSelectionSort);
}

public static int bubbleSort(int[] arr) {
    int n = arr.length;
    int comparisons = 0;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            comparisons++;
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    System.out.println("Sorted array in ascending order(Bubble Sort):");
    printArray(arr);
    return comparisons;
}

public static int selectionSort(int[] arr) {
    int n = arr.length;
    int comparisons = 0;
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            comparisons++;
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
    }
}

```

```

    }
    int temp = arr[minIndex];
    arr[minIndex] = arr[i];
    arr[i] = temp;
}
System.out.println("Sorted array in ascending order(Selection Sort):");
printArray(arr);
return comparisons;
}

public static void printArray(int[] arr) {
    System.out.print("|");
    for (int i = 0; i < arr.length; i++) {
        System.out.printf(" %d |", arr[i]);
    }
    System.out.println();
}
}
}

```

OUTPUT:

```

PS C:\Users\vijai\Documents\ASSIGNMENTS\JAVA-PROGRAMMING\LAB5_2308>
568.java } ; if ($?) { java Sort3568 }
Current Date: 2023-08-28
Current Time: 22:18:49.261587900
Name: Vijai Suria M
Register Number: (2021503568)
Enter no of elements: 6
Enter the array elements:
3 12 67 1 0 78
Sorted array in ascending order(Bubble Sort):
| 0 | 1 | 3 | 12 | 67 | 78 |
Sorted array in ascending order(Selection Sort):
| 0 | 1 | 3 | 12 | 67 | 78 |
Number of comparisons for Bubble Sort: 15
Number of comparisons for Selection Sort: 15

```

4.2) Sort random integer/character Write a program to read n random integer in a 1D array.

- a) Apply method to sort the generated array content and return the number of comparisons done.
- b) Apply another method to generate character array using the random integer and sort the array.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Random;
import java.util.Scanner;

public class SortRandom3568 {

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter no of elements: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

        arr = RandomArray(arr);
        int comparisonsInt = OrderNSort(arr.clone());
        char[] charArray = CharArray(arr);
        int comparisonsChar = OrderNSort(charArray.clone());
        System.out.println("Number of comparisons for integer array sorting: " +
comparisonsInt);
        System.out.println("Number of comparisons for character array sorting: " +
comparisonsChar);
    }

    public static int[] RandomArray(int[] arr) {
        Random random = new Random();
```

```

        for (int i = 0; i < arr.length; i++) {
            arr[i] = random.nextInt(26);
        }
        return arr;
    }

    public static char[] CharArray(int[] intArray) {
        char[] charArray = new char[intArray.length];
        for (int i = 0; i < intArray.length; i++) {
            charArray[i] = (char) (intArray[i] + 'a');
        }
        return charArray;
    }

    public static int OrderNSort(int[] arr) {
        int comparisons = 0;
        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - i - 1; j++) {
                comparisons++;
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
        System.out.println("\nSorted array in ascending order(Selection Sort):");
        printArray(arr);
        return comparisons;
    }

    public static int OrderNSort(char[] arr) {
        int comparisons = 0;
        for (int i = 0; i < arr.length - 1; i++) {
            for (int j = 0; j < arr.length - i - 1; j++) {
                comparisons++;
                if (arr[j] > arr[j + 1]) {
                    char temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

```

```

    }
}
System.out.println("\nSorted array in ascending order(Selection Sort:");
printArray(arr);
return comparisons;
}
public static void printArray(int[] arr) {
    System.out.print("|");
    for (int i = 0; i < arr.length; i++) {
        System.out.printf(" %d |", arr[i]);
    }
}
public static void printArray(char[] arr) {
    System.out.print("-----\n");
    System.out.print("|");
    for (int i = 0; i < arr.length; i++) {
        System.out.printf(" %c |", arr[i]);
    }
    System.out.print("\n-----\n");
}
}
}

```

OUTPUT:

```

PS C:\Users\vijai\Documents\ASSIGNMENTS\JAVA-PROGRAMMING\LAB5_2308>
andom3568.java } ; if ($?) { java SortRandom3568 }
Current Date: 2023-08-28
Current Time: 22:29:17.281840300
Name: Vijai Suria M
Register Number: (2021503568)
Enter no of elements: 5

Sorted array in ascending order(Selection Sort):
| 1 | 2 | 6 | 17 | 25 |
Sorted array in ascending order(Selection Sort):
-----
| b | c | g | r | z |
-----
Number of comparisons for integer array sorting: 10
Number of comparisons for character array sorting: 10

```

4.3) Search element Occurrence Write a program to read n random integer in a 1D array of A and B of size n. Apply method to search the occurrence of element in B and print the number of B element occurrence in A.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Random;
import java.util.Scanner;

public class SortRandom3568 {

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter no of elements: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

        arr = RandomArray(arr);
        int comparisonsInt = OrderNSort(arr.clone());

        char[] charArray = CharArray(arr);
        int comparisonsChar = OrderNSort(charArray.clone());

        System.out.println("Number of comparisons for integer array sorting: " +
            comparisonsInt);
        System.out.println("Number of comparisons for character array sorting: " +
            comparisonsChar);
    }

    public static int[] RandomArray(int[] arr) {
        Random random = new Random();
        for (int i = 0; i < arr.length; i++) {
            arr[i] = random.nextInt(26);
        }
    }
}
```

```

    return arr;
}

public static char[] CharArray(int[] intArray) {
    char[] charArray = new char[intArray.length];
    for (int i = 0; i < intArray.length; i++) {
        charArray[i] = (char) (intArray[i] + 'a');
    }
    return charArray;
}

public static int OrderNSort(int[] arr) {
    int comparisons = 0;
    for (int i = 0; i < arr.length - 1; i++) {
        for (int j = 0; j < arr.length - i - 1; j++) {
            comparisons++;
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    System.out.println("\nSorted array in ascending order(Selection Sort):");
    printArray(arr);
    return comparisons;
}

public static int OrderNSort(char[] arr) {
    int comparisons = 0;
    for (int i = 0; i < arr.length - 1; i++) {
        for (int j = 0; j < arr.length - i - 1; j++) {
            comparisons++;
            if (arr[j] > arr[j + 1]) {
                char temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

```



```

    }
}
System.out.println("\nSorted array in ascending order(Selection Sort:");
printArray(arr);
return comparisons;
}

public static void printArray(int[] arr) {
    System.out.print("|");
    for (int i = 0; i < arr.length; i++) {
        System.out.printf(" %d |", arr[i]);
    }
}

public static void printArray(char[] arr) {
    System.out.print("-----\n");
    System.out.print("|");
    for (int i = 0; i < arr.length; i++) {
        System.out.printf(" %c |", arr[i]);
    }
    System.out.print("\n-----\n");
}
}

```

OUTPUT:

```

PS C:\Users\vijai\Documents\ASSIGNMENTS\JAVA-PROGRAMMING\LAB5_2308>
ence3568.java } ; if ($?) { java Occurence3568 }
Current Date: 2023-08-28
Current Time: 22:34:04.628814100
Name: Vijai Suria M
Register Number: (2021503568)
Randomly generated value of n: 1
Random integers for array A:
49
Random integers for array B:
16
Number of occurrences of 16 in array A: 0

```

4.4) Sum of arrays Write a program to read two 2D array. Apply method to perform column major sum and sort the array based on the sum of columns.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class sumArrays3524 {
    static void arraySum(int[][] a, int[][] s, int rows, int cols) {
        // Calculate the sum of corresponding elements in two arrays
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                a[i][j] += s[i][j];
            }
        }
        System.out.println("Sum of two arrays is:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }
    }
    static void colSumMax(int[][] x, int[][] y, int rows, int cols) {
        // Calculate the sum of arrays and sort based on column sum
        arraySum(x, y, rows, cols);
        int[] colsum = new int[cols];
        for (int i = 0; i < cols; i++) {
            for (int j = 0; j < rows; j++) {
                colsum[i] += x[j][i];
            }
        }
        int[] csindex = new int[cols];
        for (int j = 0; j < cols; j++) {
            csindex[j] = j;
        }
    }
}
```

```

        for (int i = 0; i < cols - 1; i++) {
            for (int j = 0; j < cols - i - 1; j++) {
                if (colsum[j] > colsum[j + 1]) {
                    int temp = colsum[j];
                    colsum[j] = colsum[j + 1];
                    colsum[j + 1] = temp;

                    int temp1 = csindex[j];
                    csindex[j] = csindex[j + 1];
                    csindex[j + 1] = temp1;
                }
            }
        }
        System.out.println("The array after sorting based on column sum is:");
        // Display the sorted array
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(x[i][csindex[j]] + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        Scanner t = new Scanner(System.in);
        System.out.print("Enter the no. of rows: ");
        int r = t.nextInt();
        System.out.print("Enter the no. of columns: ");
        int c = t.nextInt();
        int[][] a1 = new int[r][c];
        int[][] a2 = new int[r][c];
        System.out.println("Enter the values of matrix 1:");
        // Input values for matrix 1
        for (int i = 0; i < r; i++) {

```

```

        for (int j = 0; j < c; j++) {
            a1[i][j] = t.nextInt();
        }
    }
    System.out.println("Enter the values of matrix 2:");
    // Input values for matrix 2
    for (int k = 0; k < r; k++) {
        for (int l = 0; l < c; l++) {
            a2[k][l] = t.nextInt();
        }
    }
    colSumMax(a1, a2, r, c);
}
}

```

OUTPUT:

```

Name: Vijai Suria M
Register Number: (2021503568)
Enter the no. of rows: 2
Enter the no. of columns: 2
Enter the values of matrix 1:
1 2
3 4
Enter the values of matrix 2:
5 6
7 8
Sum of two arrays is:
6 8
10 12
The array after sorting based on column sum is:
6 8
10 12

```

Result: Thus, basic concepts of arrays in java were implemented successfully. And thus, outputs were verified.

STRING HANDLING

Aim: To implement basic concepts of String handling in Java

Algorithm:

Step 1: Perform string methods

Methods used:

- a) **System.out.println(String)** - Print a string to the standard output.
- b) **LocalDate.now()** - To get the current date.
- c) **LocalTime.now()** - To get the current time.
- d) **s1 == s2** - Checks if s1 and s2 are the same object in memory.
- e) **s1.equals(s2)** - Compares the content of s1 and s2 to check if they are equal.
- f) **s1.compareTo(s2)** - Compares two strings lexicographically and returns an integer based on their order.
- g) **s1.charAt(index)** - Returns the character at the specified index in the string.
- h) **s1.indexOf(char)** - Returns the first occurrence index of the specified character in the string.
- i) **s1.lastIndexOf(char)** - Returns the last occurrence index of the specified character in the string.
- j) **s1.length()** - Returns the number of characters in the string.
- k) **s1.substring(start,end)** - Extracts a substring from s1 starting at start index and ending before end index.
- l) **s1.startsWith(string)** - Checks if s1 starts with the specified string.
- m) **s1.endsWith(string)** - Checks if s1 ends with the specified string.
- n) **s1.toLowerCase()** - Converts s1 to lowercase.
- o) **s1.toUpperCase()** - Converts s1 to uppercase.
- p) **s1.replace(char,char)** - Replaces all occurrences of a character with another character in the string.
- q) **s1.replaceAll(char,char)** - Replaces all occurrences of a character with another character in the string.
- r) **s1.replaceFirst(char,char)** - Replaces the first occurrence of a character with another character in the string.
- s) **s1.toCharArray(char)** - Converts the string to a character array.

Step 2: Show that String is immutable

Methods used:

- a) **referenceCheck():** Method to check if two objects have the same reference or not.
- b) **displayStudentInfo():** Method to display the student's name and registration number.

Step 3: Reverse the String

Methods used:

- a) **ReverseString** - class with a constructor to initialize the str variable.
- b) **reverseString()** - method takes user input, reverses the input string, and prints the reversed string.

Step 4: Count the number of occurrence of the each letter in the given string.

Methods used:

- a) **LetterOccurrences** - class with a constructor to initialize the inputString variable.
- b) **countLetterOccurrences()** - method takes user input, counts the occurrences of each letter in the input string, and prints the results.

Step 5: Count the number of words in the given string

Methods used:

- a) **WordCounter** - class with a constructor to initialize the inputString variable.
- b) **countWord()** - To get the current time.

5.1.1) Write a java program to perform string methods by considering the given string inputs String s1="Welcome to Java"; String s2=s1; String s3=new String("Welcome to Java"); String s4=s1.intern();

CODE:

```
import java.util.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class String3568{
    public static void main(String[] args){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        System.out.println("\n");
        String s1 = "Welcome to java";
```



```

String s2 = s1;
String s3 = new String("Welcome to java");
String s4 = s1.intern();
if(s1 == s2)
    System.out.println("1.String s1 and s2 are equal  checked using s1 == s2 ");
else System.out.println("1.String s1 and s2 are not equal checked using s1 == s2");
if(s2 == s3)
    System.out.println("2.String s2 and s3 are equal checked using s2 == s3");
else System.out.println("2.String s2 and s3 are not equal checked using s2 == s3");
if(s1.equals(s2))
    System.out.println("3.String s1 and s2 are equal checked using s1.equals(s2) ");
else System.out.println("3.String s1 and s2 are not equal checked using s1.equals(s2) ");
if(s2.equals(s3))
    System.out.println("4.String s2 and s3 are equal checked using s2.equals(s3)");
else System.out.println("4.String s2 and s3 are not equal checked using s2.equals(s3) ");
if(s1.compareTo(s2) == 0)
    System.out.println("5.String s1 and s2 are equal checked using s1.compareTo(s2) ");
else System.out.println("5.String s1 and s2 are not equal checked using
s1.compareTo(s2) ");
if(s2.compareTo(s3) == 0)
    System.out.println("6.String s2 and s3 are equal checked using s2.compareTo(s3)");
else System.out.println("6.String s2 and s3 are not equal checked using
s2.compareTo(s3) ");
if(s1 == s4)
    System.out.println("7.String s1 and s2 are equal  checked using s1 == s4 ");
else System.out.println("7.String s1 and s2 are not equal checked using s1 == s4");
System.out.println("8. s1.charAt(0) : "+s1.charAt(0));
System.out.println("9. s1.indexOf('j') : "+s1.indexOf('j'));
System.out.println("10. s1.indexOf(\"to\") : "+s1.indexOf("to"));
System.out.println("11. s1.lastIndexOf('a') : "+s1.lastIndexOf('a'));
System.out.println("12. s1.lastIndexOf(\"o\",15) : "+s1.lastIndexOf('a'));
System.out.println("13. s1.length() : " + s1.length());
System.out.println("14. s1.substring(5) : " + s1.substring(5));
System.out.println("15. s1.substring(5,11) : " + s1.substring(5,11));
System.out.println("16. s1.startsWith(\"Wel\") : " + s1.startsWith("Wel"));
System.out.println("17. s1.endsWith(\"java\") : " + s1.startsWith("java"));
System.out.println("18. s1.toLowerCase() : " + s1.toLowerCase());
System.out.println("19. s1.toUpperCase() : " + s1.toUpperCase());

```

```

String s7 = " Welcome ";
System.out.println("20. "+ s7+".trim()+" " : " +s7.trim());
System.out.println("21. s1.replace('o','T') : " + s1.replace('o','T') );
System.out.println("22. s1.replaceAll(\"o\\\", \"T\\\") : " + s1.replaceAll("o","T") );
System.out.println("23. s1.replaceFirst(\"o\\\", \"T\\\") : " + s1.replaceFirst("o","T") );
System.out.println("24. s1.toCharArray() : ");
char ch[] = s1.toCharArray() ;
for(char c : ch)
    System.out.print(c);
}
}

```

OUTPUT:

```

Current Date: 2023-09-24
Current Time: 22:09:40.228307600
Name: Vijai Suria M
Register Number: (2021503568)

1.String s1 and s2 are equal  checked using s1 == s2
2.String s2 and s3 are not equal checked using s2 == s3
3.String s1 and s2 are equal checked using s1.equals(s2)
4.String s2 and s3 are equal checked using s2.equals(s3)
5.String s1 and s2 are equal checked using s1.compareTo(s2)
6.String s2 and s3 are equal checked using s2.compareTo(s3)
7.String s1 and s2 are equal  checked using s1 == s4
8. s1.charAt(0) : W
9. s1.indexOf('j') : 11
10. s1.indexOf("to") : 8
11. s1.lastIndexOf('a') : 14
12. s1.lastIndexOf("o",15) : 14
13. s1.length() : 15
14. s1.substring(5) : me to java
15. s1.substring(5,11) : me to
16. s1.startsWith("Wel") : true
17. s1.endsWith("java") : false
18. s1.toLowerCase() : welcome to java
19. s1.toUpperCase() : WELCOME TO JAVA
20. Welcome .trim() : Welcome
21. s1.replace('o','T') : WelcTme tT java
22. s1.replaceAll("o","T") : WelcTme tT java
23. s1.replaceFirst("o","T") : WelcTme to java
24. s1.toCharArray() :

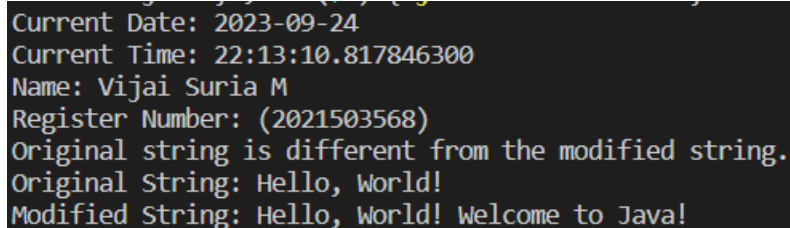
```

5.1.2) Write a program to show that String is immutable in java.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
public class Immutable3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        String originalString = "Hello, World!";
        String modifiedString = originalString.concat(" Welcome to Java!");
        if (originalString == modifiedString) {
            System.out.println("Original string is the same as the modified string.");
        } else {
            System.out.println("Original string is different from the modified
string.");
        }
        // Print the original and modified strings
        System.out.println("Original String: " + originalString);
        System.out.println("Modified String: " + modifiedString);
    }
}
```

OUTPUT:

A screenshot of a terminal window showing the output of the Java program. The text is as follows:

```
Current Date: 2023-09-24
Current Time: 22:13:10.817846300
Name: Vijai Suria M
Register Number: (2021503568)
Original string is different from the modified string.
Original String: Hello, World!
Modified String: Hello, World! Welcome to Java!
```

REASON:

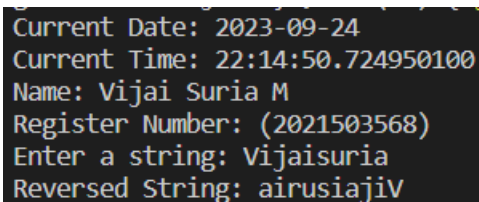
We attempt to modify the string by concatenating " Welcome to Java!" to it using the **concat** method. However, instead of modifying the **originalString**, a new string is created, and **modifiedString** references the new string.

5.2) Write a java program to read the string and displays the reverse of the string

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Scanner;;
public class StringReverse3568 {
    public static void main(String args[]){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        Scanner sc = new Scanner(System.in);
        String s;
        System.out.print("Enter a string: ");
        s = sc.nextLine();
        char[] charArray = s.toCharArray();
        int length = charArray.length;
        for (int i = 0; i < length / 2; i++) {
            char temp = charArray[i];
            charArray[i] = charArray[length - 1 - i];
            charArray[length - 1 - i] = temp;
        }
        String reversedString = new String(charArray);
        System.out.println("Reversed String: " + reversedString);
        sc.close();
    }
}
```

OUTPUT:

A screenshot of a terminal window showing the output of the Java program. The text is as follows:

```
Current Date: 2023-09-24
Current Time: 22:14:50.724950100
Name: Vijai Suria M
Register Number: (2021503568)
Enter a string: Vijaisuria
Reversed String: airusiajiv
```

5.3) Write a java program to count the number of occurrence of the each letter in the given string

CODE:

```
import java.util.Scanner;
public class LettersCount3568 {
    public static void main(String args[]){
        //Write a java program to count the occurrence of each letters in a string
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Current Time: " + java.time.LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        String s;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        s = sc.nextLine();
        int[] count = new int[26];
        for(int i = 0; i < s.length(); i++){
            if(s.charAt(i) >= 'a' && s.charAt(i) <= 'z'){
                count[s.charAt(i) - 'a']++;
            }
        }
        for(int i = 0; i < 26; i++){
            if(count[i] != 0){
                System.out.println((char)(i + 'a') + " occurs " + count[i] + " times");
            }
        }
        sc.close();
    }
}
```

Output:

```
Current Date: 2023-09-24
Current Time: 22:17:44.477670500
Name: Vijai Suria M
Register Number: (2021503568)
Enter a string: Hello
e occurs 1 times
l occurs 2 times
o occurs 1 times
```

5.4) Write a Java program to count the number of words in the given string

CODE:

```
import java.util.Scanner;
public class WordsCount3568 {
    public static void main(String args[]){
        // Write a java program to count the number of words in a string
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Current Time: " + java.time.LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        String s;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        s = sc.nextLine();
        int count = 1;
        for(int i = 0; i < s.length(); i++){
            if(s.charAt(i) == ' '){
                count++;
            }
        }
        System.out.println("Number of words in the string: " + count);
    }
}
```

OUTPUT:

```
Current Date: 2023-09-24
Current Time: 22:18:47.844729600
Name: Vijai Suria M
Register Number: (2021503568)
Enter a string: Good Morning, Vijai
Number of words in the string: 3
```

5.5.1) Write a java program to check the given string is palindrome or not (Example:Race car)

CODE:

```
import java.time.LocalDate;
```

```

import java.time.LocalDateTime;
import java.util.Scanner;
public class Palindrome3568{
    public static boolean isPalindrome(String s){
        if(s.length() == 0 || s.length() == 1){
            return true;
        }
        if(s.charAt(0) == s.charAt(s.length()-1)){
            return isPalindrome(s.substring(1, s.length()-1));
        }
        return false;
    }
    public static void main(String[] args){
        System.out.println("Current Date: " + LocalDateTime.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner sc = new Scanner(System.in);
        String s;
        System.out.print("Enter a string: ");
        s = sc.nextLine();
        System.out.println(isPalindrome(s));
        sc.close();
    }
}

```

OUTPUT:

```

Current Date: 2023-09-24
Current Time: 22:20:49.935699600
Name: Vijai Suria M
Register Number: (2021503568)
Enter a string: madam
true

```

```

Current Date: 2023-09-24
Current Time: 22:21:29.539266200
Name: Vijai Suria M
Register Number: (2021503568)
Enter a string: Vijai
false

```

**5.2) Write a java program to check the given string is anagram or not
(Example Iceman vs Cinema)**

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
import java.util.Arrays;
public class Anagram3568 {
    public static boolean areAnagrams(String str1, String str2) {
        // Remove all whitespace and convert to lowercase
        str1 = str1.replaceAll("\\s", "").toLowerCase();
        str2 = str2.replaceAll("\\s", "").toLowerCase();

        // Check if the lengths are different
        if (str1.length() != str2.length()) {
            return false;
        }

        // Convert the strings to char arrays and sort them
        char[] charArray1 = str1.toCharArray();
        char[] charArray2 = str2.toCharArray();
        Arrays.sort(charArray1);
        Arrays.sort(charArray2);

        // Compare the sorted char arrays
        return Arrays.equals(charArray1, charArray2);
    }
    public static void main(String args[]){
        //Write a java program to check the given string is anagram or not
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner sc = new Scanner(System.in);
        String s1, s2;
        System.out.print("Enter first string: ");
        s1 = sc.nextLine();
        System.out.print("Enter second string: ");
        s2 = sc.nextLine();
        if(areAnagrams(s1, s2)){
```



```

        System.out.println("The given strings are anagrams");
    }else{
        System.out.println("The given strings are not anagrams");
    }
    sc.close();
}
}

```

OUTPUT:

```

Current Date: 2023-09-24
Current Time: 20:52:07.906581
Name: Vijai Suria M
Register Number: (2021503568)
Enter first string: iceman
Enter second string: cinema
The given strings are anagrams

```

6) Write a java program that read a two string of the given format and compares the string

CODE:

```

import java.util.Scanner;

public class StringCompare3568 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the first string: ");
        String str1 = scanner.nextLine();
        System.out.println("Enter the second string: ");
        String str2 = scanner.nextLine();

        String[] parts1 = str1.split("\\.");
        String[] parts2 = str2.split("\\.");

        if (parts1.length != 3 || parts2.length != 3) {
            System.out.println("Invalid input format. Please use dd.mm.ss format.");
            return;
        }
    }
}

```

```

    }

    int day1 = Integer.parseInt(parts1[0]);
    int month1 = Integer.parseInt(parts1[1]);
    int second1 = Integer.parseInt(parts1[2]);

    int day2 = Integer.parseInt(parts2[0]);
    int month2 = Integer.parseInt(parts2[1]);
    int second2 = Integer.parseInt(parts2[2]);

    if (day1 > day2) {
        System.out.println(str1 + " is greater than " + str2 + " as " + day1 + " > " +
day2);
    } else if (day1 < day2) {
        System.out.println(str2 + " is greater than " + str1 + " as " + day2 + " > " +
day1);
    } else {
        if (month1 > month2) {
            System.out.println(str1 + " is greater than " + str2 + " as " + month1 + " > "
+ month2);
        } else if (month1 < month2) {
            System.out.println(str2 + " is greater than " + str1 + " as " + month2 + " > "
+ month1);
        } else {
            if (second1 > second2) {
                System.out.println(str1 + " is greater than " + str2 + " as " + second1 + "
> " + second2);
            } else if (second1 < second2) {
                System.out.println(str2 + " is greater than " + str1 + " as " + second2 + "
> " + second1);
            } else {
                System.out.println(str1 + " is equal to " + str2);
            }
        }
    }

    scanner.close();
}
}

```

OUTPUT:

```
geempare5500.java } } 21 (47) { java Stringgeempare5500
Current Date: 2023-09-24
Current Time: 22:26:41.397290800
Name: Vijai Suria M
Register Number: (2021503568)
Enter the first string:
15.10.10
Enter the second string:
14.20.10
15.10.10 is greater than 14.20.10 as 15 > 14
```

```
Current Date: 2023-09-24
Current Time: 22:27:11.319744100
Name: Vijai Suria M
Register Number: (2021503568)
Enter the first string:
14.12.10
Enter the second string:
14.10.50
14.12.10 is greater than 14.10.50 as 12 > 10
```

5.7) Write a java program using String methods to compare the email is valid is invalid and returns the username and domain name

i) Valid Username: numbers[0-7], alphabets[a-z][A-Z], underscore, dot, hyphen and plus characters

ii) Presence of @ symbol

iii) Presence of domainname.com or .in or .edu

Hint use givenstring.split("@") to find specific user(case-insensitive;jc_vp) and specific domain

(case-insensitive: gmail.com) for example jc_vp@gmail.com.

CODE:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Scanner;
```

```

public class EmailValidator3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an email address: ");
        String email =sc.nextLine();

        // Regular expression pattern for email validation
        String emailPattern = "^[a-zA-Z0-9_+.-]+@[a-zA-Z0-9.-
]+\\.(com|in|edu)$";

        // Create a Pattern object
        Pattern pattern = Pattern.compile(emailPattern);

        // Match the input email against the pattern
        Matcher matcher = pattern.matcher(email);
        if (matcher.matches()) {
            System.out.println("Email is valid");

            // Split the email address using "@" symbol
            String[] parts = email.split("@");

            // Extract and print the username (case-insensitive)
            String username = parts[0];
            System.out.println("Username: " + username.toLowerCase());

            // Extract and print the domain name (case-insensitive)
            String domain = parts[1];
            System.out.println("Domain: " + domain.toLowerCase());
        } else {
            System.out.println("Email is invalid");
        }
        sc.close();
    }
}

```

OUTPUT:

```
Current Date: 2023-09-24
Current Time: 22:29:49.006085500
Name: Vijai Suria M
Register Number: (2021503568)
Enter an email address: vj%@gmail.com
Email is invalid
```

```
Enter an email address: vj+@gmail.com
Email is valid
Username: vj+
Domain: gmail.com
```

5.8. Write a java program to create a dictionary using 2D string array for any 10 programming languages.

Write a method that return the definition for the input of PL name.

Java- pure object oriented programming language by James Gosling

C++ - Object oriented programming language by Stroustrup

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class DictionaryPL3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        // Define a 2D string array to store programming languages and their
definitions
        String[][] programmingLanguages = {
            {"Java", "Pure object-oriented programming language by James
Gosling"},
            {"C++", "Object-oriented programming language by Stroustrup"},
            // Add definitions for more programming languages here
        }
    }
}
```

```

        {"Python", "High-level programming language known for its
simplicity"},
        {"JavaScript", "Scripting language commonly used for web
development"},
        {"C#", "Object-oriented language developed by Microsoft"},
        {"Ruby", "Dynamic, reflective, and object-oriented language"},
        {"Swift", "Apple's programming language for iOS and macOS
development"},
        {"Kotlin", "Modern statically-typed language for Android
development"},
        {"Go", "Concurrent and statically typed language developed by
Google"},
        {"Rust", "Systems programming language focused on safety and
performance"}
    };

    Scanner sc = new Scanner(System.in);
    // Example input programming language name
    System.out.print("Enter the Programming Language: ");
    String inputLanguage = sc.nextLine(); // Change this to the language
you want to look up

    // Call the method to get the definition and print it
    String definition = getDefinition(programmingLanguages,
inputLanguage);
    if (definition != null) {
        System.out.println(inputLanguage + " - " + definition);
    } else {
        System.out.println("Programming language not found in the
dictionary.");
    }
}

// Method to retrieve the definition for a given programming language
public static String getDefinition(String[][] languages, String
languageName) {
    for (String[] language : languages) {
        if (language[0].equalsIgnoreCase(languageName)) {

```

```
        return language[1];
    }
}
return null; // Return null if the language is not found
}
}
```

OUTPUT:

```
Current Date: 2023-09-24
Current Time: 22:31:16.706170300
Name: Vijai Suria M
Register Number: (2021503568)
Enter the Programming Language: Python
Python - High-level programming language known for its simplicity
```

```
Current Date: 2023-09-24
Current Time: 22:31:38.134741300
Name: Vijai Suria M
Register Number: (2021503568)
Enter the Programming Language: Java
Java - Pure object-oriented programming language by James Gosling
```

5.9. Write a program to implement Hangman game in Java, a word-guessing game where one player thinks of a Secret word, and another player (the computer i.e., Guesser or user) tries to guess it by stating one letter at a time. If the guessed letter is in the secret word, it's revealed in the display; otherwise the number of attempt is reduced by one. The player usually has a limited number of incorrect guesses allowed before they lose the game. The game ends when the player successfully guesses the word (win) or when they run out of allowed incorrect guesses (lose). The game typically has the following components:

Input:

The guesser or user tries to guess the secret word by stating each of the character in the secret word.

Output:

A representation of the secret word is displayed with underscores for each letter that hasn't been guessed yet.

For example, if the secret word is "java" and the player has guessed 'a', the display would be "_ a _ a".

Method: Guessing

Guessing: Fix the maximum attempts. Create a guessed Letter Boolean array of secret word length size to verify the guessed character of secret word.

Process the guess [10 points each]

- i. Check input is a valid input (i.e., a character) Check if the guessed letter is in the secret word.
- ii. Update guessed Letters array.
- iii. Check if the entire word has been guessed.
- iv. Increment attempts if the guess is incorrect.
- v. Print game messages Win/Lose Conditions
- vi. The player receives feedback on their guesses, including whether the guessed letter is in the word and the current state of the word display.
- vii. Replay: After the game ends, the players may choose to play again with a new secret word.

Increase the complexity of the program by having more than one secret Word and choose it randomly. Give clues about the secret Word and reduce the points accordingly.

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.*;

class Hangman3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:");
    }
}
```



```
(2021503568)");
```

```
String words[] = { "java", "internship", "college", "computer", "github" };
String hints[][] = {
    { "Widely used programming language", "Platform independence" },
    { "Temporary work experience", "Academic years" },
    { "Higher education institution", "Beyond high school" },
    { "Data processing device", "Calculations electronically" },
    { "Code collaboration platform", "Version control" }
};
```

```
Scanner in = new Scanner(System.in);
```

```
int chances, length, mode;
boolean k = true;
while (k) {
    int clu = 0;
    Random rand = new Random();
    int ran = rand.nextInt(5);

    String tobe = words[ran];
    length = tobe.length();
    char[] actual = tobe.toCharArray();
    char[] guessword = new char[length];
    for (int i = 0; i < length; i++)
        guessword[i] = ' ';
    chances = length + (length / 3);
```

```
    boolean t = false;
    while (chances > 0) {
        for (int i = 0; i < length; i++) {
            if (guessword[i] == ' ')
                System.out.print("_ ");
            else
                System.out.print(guessword[i]);
        }
        System.out.println("\n" + chances + " chances left");
        char guess;
```

```

        if (clu <= 1) {
            System.out.print("Enter 1 for clue, 2 for guessing:");
            mode = in.nextInt();
        } else {
            if (t == false)
                System.out.println("No clues left");
            t = true;
            mode = 2;
        }
        if (mode == 1) {

            System.out.println("Clue : " + hints[ran][clu]);

            clu++;
            chances--;
        } else {
            in.nextLine();
            System.out.println("Guess a letter : ");
            guess = in.next().charAt(0);
            chances--;
            if (!(guess >= 'a' && guess <= 'z')) {
                System.out.println("Invalid input. Only give small letter
alphabets!");
            } else {
                for (int i = 0; i < actual.length; i++) {
                    if (guess == actual[i]) {
                        guessword[i] = actual[i];
                    }
                }
            }
        }
        boolean found = true;
        for (int i = 0; i < length; i++) {
            if (guessword[i] == ' ') {
                found = false;
                break;
            }
        }
    }

```

```

    }
    if (found == true) {
        break;
    }

}
boolean f = true;
for (int i = 0; i < length; i++) {
    if (guessword[i] == ' ') {
        f = false;
        break;
    }
}
if (f == true) {
    System.out.println("Congratulations!! You have found the word!!The
word is " + tobe);
} else {
    System.out.println("Chance over...!! Better luck next time");
}
int p = 0;
System.out.print("Enter 0 to exit, 1 to play again:");
p = in.nextInt();
if (p == 0)
    k = false;

}
in.close();
}
}

```

OUTPUT:

1) Player winning against computer:

```
Current Date: 2023-09-27
Current Time: 18:48:50.657738700
Name: Vijai Suria M
Register Number: (2021503568)

_ _ _ _
5 chances left
Enter 1 for clue, 2 for guessing:1
Clue :Widely used programming language

_ _ _ _
4 chances left
Enter 1 for clue, 2 for guessing:1
Clue :Platform independence

_ _ _ _
3 chances left
No clues left
Guess a letter :
j
j _ _ _
2 chances left
Guess a letter :
a
ja _ a
1 chances left
Guess a letter :
v
Congratulations!! You have found the word!!The word is java
Enter 0 to exit, 1 to play again:█
```

2) Player losing against computer

```
Enter 1 for clue, 2 for guessing:2
Guess a letter :
t
comput _ _
3 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
e
compute _
2 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
l
compute _
1 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
l
Chance over...!! Better luck next time
Enter 0 to exit, 1 to play again:1

_ _ _ _ _ _ _ _ _ _
13 chances left
Enter 1 for clue, 2 for guessing:█
```

```
Current Date: 2023-09-27
Current Time: 18:50:50.908212800
Name: Vijai Suria M
Register Number: (2021503568)

_ _ _ _ _
10 chances left
Enter 1 for clue, 2 for guessing:1
Clue :Data processing device

_ _ _ _ _
9 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
c
c
_ _ _ _ _
8 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
o
co
_ _ _ _ _
7 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
m
com
_ _ _ _ _
6 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
p
comp
_ _ _ _ _
5 chances left
Enter 1 for clue, 2 for guessing:2
Guess a letter :
u
compu
_ _ _ _ _
4 chances left
```

Result: Thus, basic concepts of String methods in java were implemented successfully. And thus, outputs were verified.

INHERITANCE AND INTERFACE IN JAVA

Aim: To explore about Inheritance and Interface in java programming

Algorithm:

Step 1: **Single Inheritance**

- `LocalDate.now()` - Retrieves the current date using the system clock.
- `LocalTime.now()` - Retrieves the current time using the system clock.
- `Scanner.nextInt()` - Waits for user input and reads an integer from the console.
- `Breath()` and `Response()` - Methods in Parent class (Eg: Living Being).
- `Walk()` and `NoOfLegs()` - Methods in Child class (Eg: Animal).

Step 2: **Multilevel Inheritance**

- `Scanner.nextInt()` - Waits for user input and reads an integer from the console.
- `Breath()` and `Response()` - Methods in Living Being class.
- `Walk()` and `NoOfLegs()` - Methods in Animal class which inherits LivingBeing class.
- `Meow()` and `Bark()` – Methods in Classes Cat and Dog which inherits Animal class.

Step 3: **Interface and Multiple Inheritance**

- `User(String name, String phoneNumber, String status)` - Initializes user details.
- `displayStatus()` - Displays the user's status.
- `Contact(String name, String phoneNumber, String status, int maxContacts)` - Initializes contact details.
- `addContact(User user)` - Adds a user to contacts.
- `Message(User sender, User receiver, String messageContent)` - Initializes message details.
- `displayMessage()` - Displays the message content.

- Chat(User user1, User user2, int maxMessages) - Initializes chat participants and messages.
- addMessage(Message message) - Adds a message to the chat.
- displayChatHistory() - Displays the chat history.

6.1 Write a program to show single Inheritance.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;

class LivingBeing {
    String name;

    LivingBeing(String name) {
        this.name = name;
    }

    protected void breath() {
        System.out.println("I am breating.....");
    }

    protected void response() {
        System.out.println(
            "Hi, My response: My name is " + name + "I am a Living being. \n
            Thank you!! \n Hava a Nice day :)");
    }

    public String toString() {
        return "My name is " + name;
    }
}

class Animal extends LivingBeing {
    int legs;

    Animal(int legs, String name) {
        super(name);
        this.legs = legs;
    }

    void setlegs(int legs) {
        this.legs = legs;
    }
}
```



```

void getlegs(int legs) {
    System.out.println("No. of legs: " + legs);
}

void walk() {
    System.out.println("I am walking.....");
}

@Override
public String toString() {
    return super.toString() + ". I am a animal with " + legs + "legs";
}
}

public class SingleInheritance3568 {
    public static void main(String args[]) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");

        Animal obj = new Animal(10, "Copper");
        System.out.println(obj);
    }
}

```

OUTPUT:

```

Current Date: 2023-10-04
Current Time: 14:56:49.402815900
Name: Vijai Suria M
Register Number: (2021503568)
My name is Copper. I am a animal with 10legs

```

6.2 Write a program to show Multilevel Inheritance.

CODE:

```

class LivingBeing {
    String name;
    LivingBeing(String name) {
        this.name = name;
    }
    void Breath() {
        System.out.println(name + " is breathing...");
    }
}

```

```

void Response() {
    System.out.println(name + " is responding to stimuli...");
}
public String toString() {
    return " Name: " + name;
}
}

// First-level derived class
class Animal extends LivingBeing {
    int numberOfLegs;

    Animal(String name, int numberOfLegs) {
        super(name);
        this.numberOfLegs = numberOfLegs;
    }
    void Walk() {
        System.out.println(name + " is walking...");
    }
    int NoOfLegs() {
        return numberOfLegs;
    }
    @Override
    public String toString() {
        return super.toString() + "\n Number of Legs: " + numberOfLegs;
    }
}
class Cat extends Animal {
    String breed;
    Cat(String name, int numberOfLegs, String breed) {
        super(name, numberOfLegs);
        this.breed = breed;
    }
    void Meow() {
        System.out.println(name + " is meowing...");
    }
    @Override
    public String toString() {
        return super.toString() + "\n Breed: " + breed;
    }
}
class Dog extends Animal {
    String breed;
    Dog(String name, int numberOfLegs, String breed) {
        super(name, numberOfLegs);
        this.breed = breed;
    }
    void Bark() {
        System.out.println(name + " is barking...");
    }
    @Override

```

```

    public String toString() {
        return super.toString()
            + "\n Breed: " + breed;
    }
}
public class MultilevelInheritance3568 {
    public static void main(String[] args) {
        Cat cat = new Cat("Laks", 4, "Siamese");
        Dog dog = new Dog("Copper", 4, "Pomeranian");
        System.out.println(cat);
        cat.toString();
        cat.Breath();
        cat.Response();
        cat.Walk();
        System.out.println("Number of legs: " + cat.NoOfLegs());
        cat.Meow();
        System.out.println();
        System.out.println(dog);
        dog.toString();
        dog.Breath();
        dog.Response();
        dog.Walk();
        System.out.println("Number of legs: " + dog.NoOfLegs());
        dog.Bark();
    }
}

```

OUTPUT:

```

PS C:\Users\vijai\OneDrive - Anna University\SE B10_1004\> ; if ($?) { javac MultilevelInheritance3568.java
Current Date: 2023-11-26
Current Time: 10:55:04.830786200
Name: Vijai Suria M
Register Number: (2021503568)
    Name: Laks
    Number of Legs: 4
    Breed: Siamese
Laks is breathing...
Laks is responding to stimuli...
Laks is walking...
Number of legs: 4
Laks is meowing...

    Name: Copper
    Number of Legs: 4
    Breed: Pomeranian
Copper is breathing...
Copper is responding to stimuli...
Copper is walking...
Number of legs: 4
Copper is barking...

```

6.3 Write a program in Java to create messaging service like WhatsApp that uses single inheritance, multilevel inheritance, and hierarchical inheritance.

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;

class User {
    protected String name;
    protected String phoneNumber;
    protected String status;

    // Constructor for User class
    public User(String name, String phoneNumber, String status) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.status = status;
    }
}

class Contact extends User {
    protected User[] contacts;
    protected int contactCount;

    // Constructor for Contact class
    public Contact(String name, String phoneNumber, String status, int
maxContacts) {
        super(name, phoneNumber, status);
        this.contacts = new User[maxContacts];
        this.contactCount = 0;
    }

    // Method to add a contact
    public void addContact(User user) {
        if (contactCount < contacts.length) {
            contacts[contactCount] = user;
            contactCount++;
            System.out.println(name + " added " + user.name + " to contacts.");
        } else {
            System.out.println("Contact list is full. Cannot add more contacts.");
        }
    }
}

class Message {
    protected User sender;
```

```

protected User receiver;
protected String messageContent;

// Constructor for Message class
public Message(User sender, User receiver, String messageContent) {
    this.sender = sender;
    this.receiver = receiver;
    this.messageContent = messageContent;
}
}

class Chat {
    private User[] participants;
    private Message[] messages;
    private int messageCount;
    // Constructor for Chat class
    public Chat(User user1, User user2, int maxMessages) {
        this.participants = new User[]{user1, user2};
        this.messages = new Message[maxMessages];
        this.messageCount = 0;
    }
    // Method to add a message to the chat
    public void addMessage(Message message) {
        if (messageCount < messages.length) {
            messages[messageCount] = message;
            messageCount++;
        } else {
            System.out.println("Chat history is full. Cannot add more messages.");
        }
    }
    // Method to display the chat history
    public void displayChatHistory() {
        System.out.println("Chat History between " + participants[0].name + "
and " + participants[1].name);
        for (int i = 0; i < messageCount; i++) {
            System.out.println(participants[0].name + ": " +
messages[i].messageContent);
            System.out.println(participants[1].name + ": " +
messages[i].messageContent);
        }
    }
}

public class WhatsApp3568 {
    public static void main(String[] args) {
        // Displaying current date and time
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
    }
}

```

```

        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        System.out.println("***** \n");

        // Creating User, Contact, and Message objects
        User alice = new User("Alice", "+1234567890", "Available");
        User bob = new User("Bob", "+9876543210", "Away");
        Contact aliceContacts = new Contact("Alice", "+1234567890",
"Available", 10);
        aliceContacts.addContact(bob);

        Message message1 = new Message(alice, bob, "Hi, Bob!");
        Message message2 = new Message(bob, alice, "Hello, Alice!");

        // Creating a Chat object and adding messages
        Chat chat = new Chat(alice, bob, 100);
        chat.addMessage(message1);
        chat.addMessage(message2);

        // Displaying chat history
        chat.displayChatHistory();
    }
}

```

OUTPUT:

```

Current Date: 2023-10-04
Current Time: 22:43:17.845650800
Name: Vijai Suria M
Register Number: (2021503568)
*****

Alice added Bob to contacts.
Chat History between Alice and Bob
Alice: Hi, Bob!
Bob: Hi, Bob!
Alice: Hello, Alice!
Bob: Hello, Alice!

```

RESULT:

Thus, the java programs for Inheritance and Interface were executed successfully.

EXCEPTION HANDLING MECHANISM IN JAVA

Aim: To explore about exception handling mechanism in java programming

Algorithm:

Step 1: Array index out of Bound

- `LocalDate.now()` - Retrieves the current date using the system clock.
- `LocalTime.now()` - Retrieves the current time using the system clock.
- `Scanner.nextInt()` - Waits for user input and reads an integer from the console.
- `ArrayIndexOutOfBoundsException`: Exception class that signals an invalid array index access.
- `Exception`: Base class for all checked exceptions.
- `System.out.print()` - Used to print the data in output console

Step 2: Number Format Error

- `Integer.parseInt(String s)` - Parses the string argument as a signed decimal integer.
Throws `NumberFormatException` if the string does not contain a valid integer.
- `e.getMessage()` (from `Exception` class - Retrieves the error message associated with the exception (`NumberFormatException` in this case).
- `e.printStackTrace()` (from `Throwable` class - Prints the stack trace of the exception to the standard error stream, showing the sequence of method calls leading up to the exception.

Step 3: Null Pointer Exception

- `ArrayIndexOutOfBoundsException`: Indicates an attempt to access an array element at an invalid index.
- `NullPointerException`: Indicates an attempt to access an element having null reference.
- `Scanner.nextInt()`: Reads an integer value from the console input.
- `Exception`: Base class for checked exceptions, used here to catch general exceptions.

Step 4: **Arithmetic Exception**

- `ArithmeticException`: Thrown when an arithmetic operation encounters an exceptional condition (e.g., division by zero).
- `Exception`: Base class for checked exceptions, used here to catch general exceptions.

Step 5: **File Not Found Exception**

- `BufferedInputStream`: Reads input from an underlying input stream with buffering capabilities.
- `FileInputStream`: Opens an input file for reading.
- `BufferedOutputStream`: Writes output to an underlying output stream with buffering capabilities.
- `FileOutputStream`: Opens an output file for writing.
- `in.read()`: Reads a byte of data from the input stream.
- `out.write(int b)`: Writes a byte of data to the output stream.
- `IOException`: Signals an I/O error.

Step 6: **Stack Overflow Exception**

- `StackOverflowError` - Infinite recursion or looping overloads the program's memory stack, causing a stack overflow error.
- `recursiveMethod(int depth)`: A method that recursively calls itself with an incremented depth value.

Description: Prints the current depth and recursively calls itself, incrementing the depth by one each time. This leads to a stack overflow error due to infinite recursion, as there is no termination condition..

Step 7: **User-Defined Exception**

- `Scanner.nextLine()`: Reads a line of text from the console input.
- `validateEmail(String email)` throws `InvalidEmailException`: Checks if the given email contains the '@' symbol; if not, throws an `InvalidEmailException`.
- `InvalidEmailException` extends `Exception`: Represents an exception specific to invalid email addresses.

7.1) Illustrate and demonstrate the Array Index Out of Bounds Exception

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
public class ArrayIndex3568 {
    public static void main(String args[]){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the array size: ");
        int n = in.nextInt();
        int []array = new int[n];

        try {
            System.out.print("Enter the any index to access: ");
            int i = in.nextInt();
            System.out.println("Array Element array["+i+"]: "+ array[i]);
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Checked Runtime Exception: \n" + e);
        }
        catch(Exception e){
            System.out.println("Other Exceptions: \n" + e);
        }
    }
}
```

OUTPUT:

```
Current Date: 2023-10-14
Current Time: 15:17:14.983060100
Name: Vijai Suria M
Register Number: (2021503568)
Enter the array size: 4
Enter the any index to access: 15
Checked Runtime Exception:
java.lang.ArrayIndexOutOfBoundsException: Index 15 out of bounds for length 4
```

7.2) Illustrate and demonstrate the Number Format Error:

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
public class NumberFormat {
    public static void main(String[] args) {
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        try {
            String str = "abc123"; // This is not a valid number
            int number = Integer.parseInt(str);
            System.out.println("Parsed number: " + number);
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException occurred: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
Current Date: 2023-10-17
Current Time: 16:08:11.210563100
Name: Vijai Suria M
Register Number: (2021503568)
NumberFormatException occurred: For input string: "abc123"
java.lang.NumberFormatException: For input string: "abc123"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:665)
    at java.base/java.lang.Integer.parseInt(Integer.java:781)
    at NumberFormat.main(NumberFormat.java:11)
```

7.3) Illustrate and demonstrate the Null Pointer Exception

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class NullPointer3568 {
    public static void main(String args[]){
```

```

System.out.println("Current Date: " + LocalDate.now());
System.out.println("Current Time: " + LocalTime.now());
System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
    Scanner in = new Scanner(System.in);
    int []array = null;
    try {
System.out.print("Enter the any index to access: ");
        int i = in.nextInt();
System.out.println("Array Element array["+i+"]: "+ array[i]);
    }
    catch(ArrayIndexOutOfBoundsException e){
System.out.println("Unchecked Runtime Exception: \n" + e);
    }
    catch(Exception e){
System.out.println("Other Exceptions: \n" + e);
    }
    }
}

```

OUTPUT:

```

Current Date: 2023-10-14
Current Time: 15:23:10.743703300
Name: Vijai Suria M
Register Number: (2021503568)
Enter the any index to access: 56
Other Exceptions:
java.lang.NullPointerException: Cannot load from int array because "array" is null

```

7.4) Illustrate and demonstrate the Arithmetic Exception

CODE:

```

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
public class Arithmetic3568 {
    public static void main(String args[]){
System.out.println("Current Date: " + LocalDate.now());
System.out.println("Current Time: " + LocalTime.now());
System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Scanner in = new Scanner(System.in);

```

```

        try {
            System.out.print("Enter the any number to divide the number 25: ");
            int i = in.nextInt();
            System.out.println(" 25 / "+i+" = "+ (25/i));
        }
        catch(ArithmeticException e){
            System.out.println("Unchecked Runtime Exception: \n" + e);
        }
        catch(Exception e){
            System.out.println("Other Exceptions: \n" + e);
        }
    }
}

```

OUTPUT:

```

Current Date: 2023-10-14
Current Time: 15:31:53.927854400
Name: Vijai Suria M
Register Number: (2021503568)
Enter the any number to divide the number 25: 0
Unchecked Runtime Exception:
java.lang.ArithmeticException: / by zero

```

7.5) Illustrate and demonstrate the File Not Found Exception

CODE:

```

import java.io.*;
import java.util.*;

public class BufferStream {
    public static void main(String[] args) throws IOException {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        try {
            BufferedInputStream in = new BufferedInputStream(new
            FileInputStream("input.txt"));
            BufferedOutputStream out = new BufferedOutputStream(new
            FileOutputStream("output.txt"));
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
        }
    }
}

```

```

    }
    } catch (IOException e) {
        System.out.println("File Exception at \n" + e.getMessage());
    }
}

```

OUTPUT:

```

Current Date: 2023-10-14
Current Time: 23:29:34.219428800
Name: Vijai Suria M
Register Number: (2021503568)

```

File copied successfully

```

Current Date: 2023-10-14
Current Time: 23:31:49.925880800
Name: Vijai Suria M
Register Number: (2021503568)

File Exception at
input1.txt (The system cannot find the file specified)

```

7.6) Illustrate and demonstrate the User defined email validator:

CODE:

```

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Scanner;

class InvalidEmailException extends Exception {
    public InvalidEmailException(String message) {
        super(message);
    }
}

public class EmailException3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        Scanner in = new Scanner(System.in);
        try {
            String email;
            System.out.println("Enter the email: ");

```

```

        email = in.nextLine();
        validateEmail(email);
        System.out.println("Email is valid: " + email);
    } catch (InvalidEmailException e) {
        System.out.println("Invalid Email: " + e.getMessage());
    }
}

public static void validateEmail(String email) throws InvalidEmailException
{
    if (!email.contains("@")) {
        throw new InvalidEmailException("Email is missing the '@' symbol.");
    }
}
}

```

OUTPUT:

```

Current Date: 2023-10-17
Current Time: 16:00:30.217803600
Name: Vijai Suria M
Register Number: (2021503568)
Enter the email:
vijai@mit.edu
Email is valid: vijai@mit.edu

```

```

Current Date: 2023-10-17
Current Time: 16:00:19.999025200
Name: Vijai Suria M
Register Number: (2021503568)
Enter the email:
Vijai123.com
Invalid Email: Email is missing the '@' symbol.

```

7.7) Write a java program to illustrate the Stack Overflow Error:

CODE:

```

import java.time.*;
public class StackOverflow3568 {
    public static void main(String[] args) {
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        recursiveMethod(0);
    }
    public static void recursiveMethod(int depth) {

```



```
        System.out.println("Depth: " + depth);  
        recursiveMethod(depth + 1);  
    }  
}
```

OUTPUT:

```
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)  
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)  
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)  
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)  
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)  
at StackOverflow3568.recursiveMethod(StackOverflow3568.java:14)
```

Result: The basic concepts of Exception handling mechanism in Java programming were explored successfully. And thus, outputs were verified.

FILE HANDLING IN JAVA

Aim: To explore about file handling in java programming

Algorithm:

Step 1: Copy the content of files to another file

- `BufferedInputStream`: Reads input from an underlying input stream with buffering capabilities.
- `FileInputStream`: Opens an input file for reading.
- `BufferedOutputStream`: Writes output to an underlying output stream with buffering capabilities.
- `FileOutputStream`: Opens an output file for writing.
- `in.read()`: Reads a byte of data from the input stream.
- `out.write(int b)`: Writes a byte of data to the output stream.
- `IOException`: Signals an I/O error.

Step 2: Object and File serialization

- `FileOutputStream(String name)`: Opens an output stream to write to a file with the specified name.
- `ObjectOutputStream(OutputStream out)`: Constructs an `ObjectOutputStream` that writes to the specified `OutputStream`.
- `ObjectOutputStream.writeObject(Object obj)`: Writes the specified object to the `ObjectOutputStream`.
- `FileInputStream(String name)`: Opens an input stream for reading from a file with the specified name.
- `ObjectInputStream(InputStream in)`: Constructs an `ObjectInputStream` that reads from the specified `InputStream`.
- `ObjectInputStream.readObject()`: Reads an object from the `ObjectInputStream`.
- `Serializable` interface: Indicates that the class can be serialized.
- `ObjectOutputStream.writeObject(Object obj)`: Serializes the object (writes it to a file).
- `ObjectInputStream.readObject()`: Deserializes the object (reads it from a file).

8.1) Write a java program to copy the content of one file to another file.

CODE:

```
import java.io.*;
import java.util.*;

public class BufferStream {
    public static void main(String[] args) throws IOException {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");
        try {
            BufferedInputStream in = new BufferedInputStream(new
FileInputStream("input.txt"));
            BufferedOutputStream out = new BufferedOutputStream(new
FileOutputStream("output.txt"));
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
        } catch (IOException e) {
            System.out.println("File Exception at \n" + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Current Date: 2023-10-14
Current Time: 23:29:34.219428800
Name: Vijai Suria M
Register Number: (2021503568)

File copied successfully
```

8.2) Write a Java program demonstrating object serialization and deserialization for a 'Person' class. Serialize the object to a file and then deserialize it, displaying the person's information.

CODE:

```
import java.io.*;

class Person implements Serializable {
    public String name;
    public int age;
    public String address;
    public String email;
}

public class Serialize3568 {
    public static void main(String args[]) throws Exception {
        System.out.println("Current Date: " + java.time.LocalDate.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
        Person PL;
        PL = new Person();
        PL.name = "Java";
        PL.age = 19;
        PL.address = "Chennai, Tamil Nadu, India";
        PL.email = "vijaisuriam@gmail.com";
        String Filename = "File.txt";
        FileOutputStream fo = new FileOutputStream(Filename);
        ObjectOutputStream os = new ObjectOutputStream(fo);
        os.writeObject(PL);
        FileInputStream fi = new FileInputStream(Filename);
        ObjectInputStream is = new ObjectInputStream(fi);
        Person p = (Person) is.readObject();
        System.out.println(p.name + " is " + p.age + " old living in " + p.address
            + " and you can mail at " + p.email);
    }
}
```

OUTPUT:

```
Current Date: 2023-10-14
Current Time: 23:43:48.294540600
Name: Vijai Suria M
Register Number: (2021503568)
Java is 19 old living in Chennai, Tamil Nadu, India and you can mail atvijaisuriam@gmail.com
```

Result:

The basic concepts of File handling in Java programming were explored successfully. And thus, outputs were verified.

GRAPHICAL USER INTERFACE IN JAVA

Aim: To explore about Graphical User Interface in java programming.

Algorithm:

Step 1: Quiz Form Application

- setTitle(String title) - Sets the title of the JFrame.
- setSize(int width, int height) - Sets the size of the JFrame.
- setDefaultCloseOperation(int operation) - Sets the default close operation for the JFrame.
- setLayout(LayoutManager mgr) - Sets the layout manager for the JPanel.
- createTitledBorder(String title) - Creates a titled border with the specified title.
- addActionListener(ActionListener listener) - Registers an ActionListener to the submitButton.
- add(AbstractButton b):Adds a button to the container.
- JRadioButton(String text) - Constructs a radio button with the specified text.
- JCheckBox(String text) - Constructs a check box with the specified text.
- JButton(String text) - Constructs a button with the specified text.
- calculateScore() - Calculates the total score based on the selected options and user inputs.
- getText() - Returns the text contained in the JTextField or JTextArea.
- trim() - Removes leading and trailing whitespaces from a string.
- toLowerCase() - Converts all characters in a string to lowercase.
- contains(CharSequence s) - Checks if a string contains the specified sequence of characters.
- showMessageDialog(Component parentComponent, Object message) - Displays a message dialog with the specified message.

Step 2: **Shopping Cart**

- JFrame(String title) - Initializes a JFrame with the specified title.
- setDefaultCloseOperation(int operation) - Sets the default close operation for the JFrame.
- setSize(int width, int height) - Sets the size of the JFrame.
- addColumn(Object columnName) - Adds a column to the table model.
- JTable(TableModel dm) - Creates a table that is initialized with a specific TableModel.
- addActionListener(ActionListener listener) - Adds an ActionListener to the button.
- removeRow(int row) - Removes the specified row from the table model.
- parseDouble(String s) - Parses the string argument as a double.
- JTextField(int columns) - Initializes a text field with a specified number of columns.
- JButton(String text) - Creates a JButton with the specified text.
- setBackground(Color bg) - Sets the background color of the JFrame.
- setText(String text) - Sets the text of the label.
- addDocumentListener(DocumentListener listener) - Registers a DocumentListener on the text field's document model.

Step 3: **Google Form**

- setDefaultCloseOperation(int operation) - Sets the default close operation for the JFrame.
- setSize(int width, int height) - Sets the size of the JFrame.
- setBackground(Color bg) - Sets the background color of the content pane.
- JPanel(LayoutManager layout) - Creates a JPanel with the specified layout manager.
- GridBagConstraints methods - fill, insets, gridx, gridy, gridwidth: Control grid bag layout constraints for components within a panel.
- JLabel(String text) - Creates a JLabel with the specified text.

- JTextField(int columns), JPasswordField(int columns), JComboBox(E[] items) - Initialize text fields, password fields, and combo boxes with specific column width and items.
- addActionListener(ActionListener listener) - Adds an ActionListener to a button for responding to button clicks.
- setSelectedIndex(int index) - Sets the selected index in a JComboBox.
- getModel() - Returns the data model associated with a JComboBox.
- getSelectedItem() - Returns the currently selected item in a JComboBox.
- setText(String text) - Sets the text of a JTextField or JLabel component.

9.1 Create an interactive Quiz form using radio buttons, check box, text box, text area. Write a method to find the total score of the user. Display the right answer to the user at end.

CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class QuizApp_3568 extends JFrame implements ActionListener {
    JPanel qPanel, resPanel;
    JLabel q1, q2, q3, q4, q5, res;
    JTextField score;
    private JRadioButton[] radioButtons;
    private JCheckBox[] checkBoxes;
    ButtonGroup radioGroup;
    JButton submit, ok;
    JDialog result;

    QuizApp() {
        radioButtons = new JRadioButton[4];
        checkBoxes = new JCheckBox[4];
        result = new JDialog(this, "Score", false);

        JPanel q1, q2, q3, q4, q5;

        q1 = new JLabel("Who is the father of Java?");
        q1 = new JPanel(new GridLayout(2, 4));
```

```

radioButtons[0] = new JRadioButton("James Gosling");
radioButtons[1] = new JRadioButton("Mark Zuckerberg");
radioButtons[2] = new JRadioButton("Bill Gates");
radioButtons[3] = new JRadioButton("Linus Torvalds");
radioGroup = new ButtonGroup();
for (JRadioButton radioButton : radioButtons) {
    radioGroup.add(radioButton);
}
q1.add(radioButtons[0]);
q1.add(radioButtons[1]);
q1.add(radioButtons[2]);
q1.add(radioButtons[3]);

q2 = new JLabel("What are the advantages of Java?");
q2 = new JPanel(new GridLayout(2, 4));
checkboxes[0] = new JCheckBox("Platform Independent");
checkboxes[1] = new JCheckBox("Simple");
checkboxes[2] = new JCheckBox("Robust");
checkboxes[3] = new JCheckBox("Presence of Pointers");
q2.add(checkboxes[0]);
q2.add(checkboxes[1]);
q2.add(checkboxes[2]);
q2.add(checkboxes[3]);

q3 = new JLabel("What is the main feature of OOP?");
q3 = new JPanel(new GridLayout(2, 4));
radioButtons[0] = new JRadioButton("Encapsulation");
radioButtons[1] = new JRadioButton("Inheritance");
radioButtons[2] = new JRadioButton("Polymorphism");
radioButtons[3] = new JRadioButton("Abstraction");
radioGroup = new ButtonGroup();
for (JRadioButton radioButton : radioButtons) {
    radioGroup.add(radioButton);
}
q3.add(radioButtons[0]);
q3.add(radioButtons[1]);
q3.add(radioButtons[2]);
q3.add(radioButtons[3]);

q4 = new JLabel("What is a constructor in Java?");
q4 = new JPanel(new GridLayout(2, 4));
checkboxes[0] = new JCheckBox("A special method used to initialize
objects");

```

```

checkBoxes[1] = new JCheckBox("A static variable");
checkBoxes[2] = new JCheckBox("A loop structure");
checkBoxes[3] = new JCheckBox("A data type");
q4.add(checkBoxes[0]);
q4.add(checkBoxes[1]);
q4.add(checkBoxes[2]);
q4.add(checkBoxes[3]);

ql5 = new JLabel("Which of the following is a collection in Java?");
q5 = new JPanel(new GridLayout(2, 4));
checkBoxes[0] = new JCheckBox("ArrayList");
checkBoxes[1] = new JCheckBox("HashMap");
checkBoxes[2] = new JCheckBox("String");
checkBoxes[3] = new JCheckBox("int");
q5.add(checkBoxes[0]);
q5.add(checkBoxes[1]);
q5.add(checkBoxes[2]);
q5.add(checkBoxes[3]);

submit = new JButton("Submit");
submit.setBounds(30, 30, 120, 50);
submit.addActionListener(this);

qPanel = new JPanel(new GridLayout(10, 1, 20, 20));
qPanel.add(ql1);
qPanel.add(q1);
qPanel.add(ql2);
qPanel.add(q2);
qPanel.add(ql3);
qPanel.add(q3);
qPanel.add(ql4);
qPanel.add(q4);
qPanel.add(ql5);
qPanel.add(q5);
qPanel.add(submit);

resPanel = new JPanel(new FlowLayout());
res = new JLabel("Results");
score = new JTextField(30);
resPanel.add(res);
resPanel.add(score);

ok = new JButton("OK");

```

```

        ok.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                result.setVisible(false);
            }
        });
        result.add(ok);

        setVisible(true);
        setLayout(new BorderLayout());
        setSize(800, 500);

        add(qPanel, BorderLayout.NORTH);
        add(resPanel, BorderLayout.CENTER);
    }
    public void actionPerformed(ActionEvent e) {
        int c = 0;
        if (radioButtons[0].isSelected()) {
            c++;
        }
        if (checkboxes[0].isSelected() && checkboxes[1].isSelected() &&
checkboxes[2].isSelected()) {
            c++;
        }
        if (radioButtons[3].isSelected()) {
            c++;
        }
        if (checkboxes[0].isSelected()) {
            c++;
        }
        if (checkboxes[0].isSelected() && checkboxes[1].isSelected()) {
            c++;
        }
        score.setText(String.valueOf(c));
        result.add(new JLabel("Score: " + c));
        result.setLayout(new FlowLayout());
        result.setSize(300, 300);
        result.setVisible(true);
    }
    public static void main(String[] args) {
        new QuizApp();
    }
}

```

OUTPUT:

A screenshot of a Java quiz application window. The window has a title bar with a small icon and standard window controls. The main area contains five questions on the left and a grid of radio and checkbox options on the right. A 'Submit' button is at the bottom left, and a 'Results' label with an input field is at the bottom right.

Who is the father of Java?

What are the advantages of Java?

What is the main feature of OOP?

What is a constructor in Java?

Which of the following is a collection in Java?

Submit

Results

☐ James Gosling ☐ Mark Zuckerberg
☐ Bill Gates ☐ Linus Torvalds

☐ Platform Independent ☐ Simple
☐ Robust ☐ Presence of Pointers

☐ Encapsulation ☐ Inheritance
☐ Polymorphism ☐ Abstraction

☐ A special method used to ini... ☐ A static variable
☐ A loop structure ☐ A data type

☐ ArrayList ☐ HashMap
☐ String ☐ int

A screenshot of the same Java quiz application window, but with a 'Score' dialog box open. The dialog box has an 'OK' button and displays 'Score: 3'. The quiz options are now partially visible, showing the selected answers for the questions.

Score

OK Score: 3

Which of the following is a collection in Java?

Submit

☒ James Gosling ☐ Mark Zuckerberg
☐ Bill Gates ☐ Linus Torvalds

☒ Platform Independent ☐ Simple
☐ Robust ☐ Presence of Pointers

☒ Encapsulation ☐ Inheritance
☐ Polymorphism ☐ Abstraction

☒ A special method used to ini... ☐ A static variable
☐ A loop structure ☐ A data type

☒ ArrayList ☒ HashMap
☐ String ☐ int

9.2 Create a shopping cart program using Java Swing. The program should allow users to add items to the cart, display the cart's contents (including the items and their prices), and calculate the total amount to be paid. Create a Java Swing application that fulfills these requirements.

CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
class Item {
    String name;
    double price;

    public Item(String name, double price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return name + " ($" + price + ")";
    }
}

public class ShoppingCartApp_3568 {
    private JFrame frame;
    private DefaultListModel<Item> cartModel;
    private JList<Item> cartList;
    private double totalAmount;

    JLabel totalLabel;

    public ShoppingCartApp() {
        frame = new JFrame("Shopping Cart App");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);

        cartModel = new DefaultListModel<>();
        cartList = new JList<>(cartModel);

        JPanel itemPanel = createItemPanel();
```

```

JPanel cartPanel = createCartPanel();
JPanel totalPanel = new JPanel(new FlowLayout());

JLabel totalLabelT = new JLabel("Total Amount: $");
totalLabel = new JLabel("00");
totalPanel.add(totalLabelT);
totalPanel.add(totalLabel);

frame.setLayout(new BorderLayout());
frame.add(itemPanel, BorderLayout.WEST);
frame.add(cartPanel, BorderLayout.CENTER);
frame.add(totalPanel, BorderLayout.SOUTH);

frame.setVisible(true);
}

private JPanel createItemPanel() {
    JPanel panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

    JLabel label = new JLabel("Available Items:");
    panel.add(label);

    JComboBox<Item> itemComboBox = new JComboBox<>();
    itemComboBox.addItem(new Item("Samsung S23", 10.0));
    itemComboBox.addItem(new Item("Apple 15 Pro", 15.0));
    itemComboBox.addItem(new Item("Vivo T2 5g", 20.0));
    itemComboBox.addItem(new Item("Nokio 6a", 25.0));
    panel.add(itemComboBox);

    JButton addButton = new JButton("Add to Cart");
    addButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Item selected = (Item) itemComboBox.getSelectedItem();
            cartModel.addElement(selected);
            assert selected != null;
            totalAmount += selected.price;
            totalLabel.setText(String.valueOf(totalAmount));
        }
    });
    panel.add(addButton);
}

```

```

        return panel;
    }

    private JPanel createCartPanel() {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        JLabel label = new JLabel("Shopping Cart:");
        panel.add(label);

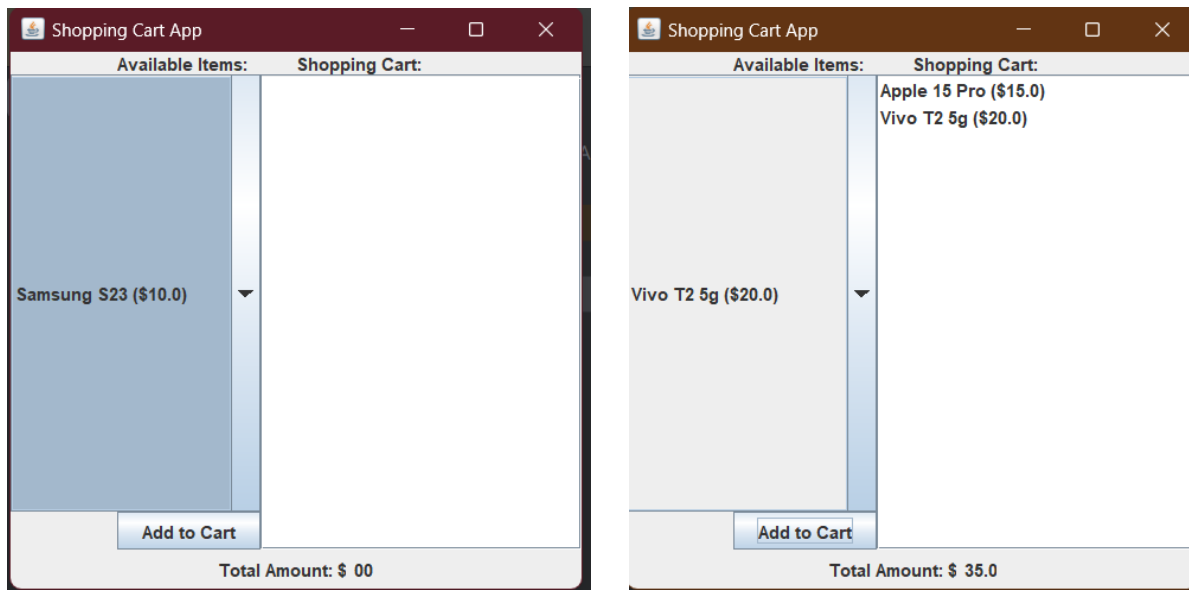
        JScrollPane scrollPane = new JScrollPane(cartList);
        panel.add(scrollPane);

        return panel;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new ShoppingCartApp();
        });
    }
}

```

OUTPUT:



9.3 Write a program to create google account with details

- Raise an exception if password and confirm password did not match
- password with atleast 8 characters.
- Raise an exception if any input box is empty
- Raise an exception an invalid emailid

CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class GoogleForm_3568 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Google Account Registration");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(5, 5, 5, 5);
        JLabel firstNameLabel = new JLabel("First Name:");
        JTextField firstNameField = new JTextField(20);
        JLabel lastNameLabel = new JLabel("Last Name:");
        JTextField lastNameField = new JTextField(20);
        JLabel emailLabel = new JLabel("Email (Username):");
        JTextField emailField = new JTextField(20);
        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordField = new JPasswordField(20);
        JLabel confirmPasswordLabel = new JLabel("Confirm Password:");
        JPasswordField confirmPasswordField = new JPasswordField(20);
        JLabel countryLabel = new JLabel("Country:");
        String[] countries = { "Select", "India", "US", "UK" };
        JComboBox<String> countryComboBox = new JComboBox<>(countries);
        JLabel phoneLabel = new JLabel("Phone number:");
        JTextField phoneField = new JTextField(20);
        JLabel recoveryEmailLabel = new JLabel("Recovery Email:");
        JTextField recoveryEmailField = new JTextField(20);
        JLabel dobLabel = new JLabel("Date of Birth:");
        JComboBox<String> monthComboBox = new JComboBox<>(new String[] {
            "Select", "January", "February", "March",
            "April", "May", "June", "July", "August", "September", "October", "November",
            "December" });
    }
}
```

```

JComboBox<String> dayComboBox = new JComboBox<>(new String[] { "Select"
});
JComboBox<String> yearComboBox = new JComboBox<>(new String[] { "Select"
});

// Populate dayComboBox with days 1 to 31 initially
String[] daysInitial = new String[32];
for (int i = 0; i <= 31; i++) {
    daysInitial[i] = Integer.toString(i);
}
dayComboBox.setModel(new DefaultComboBoxModel<>(daysInitial));
// Now, add the ActionListener for monthComboBox
monthComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedMonth = (String) monthComboBox.getSelectedItem();
        int selectedYear = Integer.parseInt((String)
yearComboBox.getSelectedItem());
        if (selectedMonth.equals("Select")) {
            dayComboBox.setModel(new DefaultComboBoxModel<>(new String[] {
"Select" }));
        } else {
            int maxDays = getMaxDaysForMonth(selectedMonth, selectedYear);
            String[] days = new String[maxDays + 1];
            days[0] = "Select";
            for (int i = 1; i <= maxDays; i++) {
                days[i] = String.valueOf(i);
            }
            dayComboBox.setModel(new DefaultComboBoxModel<>(days));
        }
    }
});

// Populate dayComboBox and yearComboBox based on the selected month
monthComboBox.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String selectedMonth = (String) monthComboBox.getSelectedItem();
        int selectedYear = Integer.parseInt((String)
yearComboBox.getSelectedItem());
        if (selectedMonth.equals("Select")) {
            dayComboBox.setModel(new DefaultComboBoxModel<>(new String[] {
"Select" }));
        } else {
            int maxDays = getMaxDaysForMonth(selectedMonth, selectedYear);
            String[] days = new String[maxDays + 1];
            days[0] = "Select";
            for (int i = 1; i <= maxDays; i++) {
                days[i] = String.valueOf(i);
            }
        }
    }
});

```

```

        }
        dayComboBox.setModel(new DefaultComboBoxModel<>(days));
    }
}
});
// Populate yearComboBox with a range of years
String[] years = new String[200];
for (int i = 1900; i <= 2018; i++) {
    years[i - 1900] = Integer.toString(i);
}
years[0] = "Select";
yearComboBox.setModel(new DefaultComboBoxModel<>(years));
JLabel genderLabel = new JLabel("Gender:");
String[] genders = { "Select", "Male", "Female", "Other" };
JComboBox<String> genderComboBox = new JComboBox<>(genders);
JButton resetButton = new JButton("Reset");
JButton submitButton = new JButton("Submit");
gbc.gridx = 0;
gbc.gridy = 0;
panel.add(firstNameLabel, gbc);
gbc.gridx = 1;
panel.add(firstNameField, gbc);
gbc.gridx = 0;
gbc.gridy = 1;
panel.add(lastNameLabel, gbc);
gbc.gridx = 1;
panel.add(lastNameField, gbc);
gbc.gridx = 0;
gbc.gridy = 2;
panel.add(emailLabel, gbc);
gbc.gridx = 1;
panel.add(emailField, gbc);
gbc.gridx = 0;
gbc.gridy = 3;
panel.add(passwordLabel, gbc);
gbc.gridx = 1;
panel.add(passwordField, gbc);
gbc.gridx = 0;
gbc.gridy = 4;
panel.add(confirmPasswordLabel, gbc);
gbc.gridx = 1;
panel.add(confirmPasswordField, gbc);
gbc.gridx = 0;
gbc.gridy = 5;
panel.add(countryLabel, gbc);
gbc.gridx = 1;
panel.add(countryComboBox, gbc);

```

```

gbc.gridx = 0;
gbc.gridy = 6;
panel.add(phoneLabel, gbc);
gbc.gridx = 1;
panel.add(phoneField, gbc);
gbc.gridx = 0;
gbc.gridy = 7;
panel.add(recoveryEmailLabel, gbc);
gbc.gridx = 1;
panel.add(recoveryEmailField, gbc);
gbc.gridx = 0;
gbc.gridy = 8;
panel.add(dobLabel, gbc);
gbc.gridx = 1;
panel.add(monthComboBox, gbc);
gbc.gridx = 2;
panel.add(dayComboBox, gbc);
gbc.gridx = 3;
panel.add(yearComboBox, gbc);
gbc.gridx = 0;
gbc.gridy = 9;
panel.add(genderLabel, gbc);
gbc.gridx = 1;
panel.add(genderComboBox, gbc);
gbc.gridx = 0;
gbc.gridy = 10;
gbc.gridwidth = 2;
panel.add(resetButton, gbc);
gbc.gridx = 2;
panel.add(submitButton, gbc);
frame.add(panel, BorderLayout.CENTER);
resetButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        resetForm(firstNameField, lastNameField, emailField, passwordField,
confirmPasswordField,
countryComboBox, phoneField, recoveryEmailField, yearComboBox,
monthComboBox, dayComboBox,
genderComboBox);
    }
});
submitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            validateForm(firstNameField, lastNameField, emailField, passwordField,
confirmPasswordField,

```

```

        countryComboBox, phoneField, recoveryEmailField, yearComboBox,
        monthComboBox, dayComboBox,
        genderComboBox);
        JOptionPane.showMessageDialog(frame, "Account successfully created!");
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
});
frame.setVisible(true);
}

private static void resetForm(JTextField firstName, JTextField lastName, JTextField
email, JPasswordField password,
        JPasswordField confirmPassword,
        JComboBox<String> country, JTextField phone, JTextField
recoveryEmail, JComboBox<String> year,
        JComboBox<String> month,
        JComboBox<String> day, JComboBox<String> gender) {
    firstName.setText("");
    lastName.setText("");
    email.setText("");
    password.setText("");
    confirmPassword.setText("");
    country.setSelectedIndex(0);
    phone.setText("");
    recoveryEmail.setText("");
    year.setSelectedIndex(0);
    month.setSelectedIndex(0);
    day.setSelectedIndex(0);
    gender.setSelectedIndex(0);
}

private static void validateForm(JTextField firstName, JTextField lastName,
JTextField email,
        JPasswordField password, JPasswordField confirmPassword,
        JComboBox<String> country, JTextField phone, JTextField
recoveryEmail, JComboBox<String> year,
        JComboBox<String> month,
        JComboBox<String> day, JComboBox<String> gender) throws
Exception {
    if (firstName.getText().isEmpty() || lastName.getText().isEmpty() ||
email.getText().isEmpty()
        || password.getPassword().length == 0 ||
confirmPassword.getPassword().length == 0
        || country.getSelectedIndex() == 0 || phone.getText().isEmpty() ||
year.getSelectedIndex() == 0
        || month.getSelectedIndex() == 0 || day.getSelectedIndex() == 0 ||

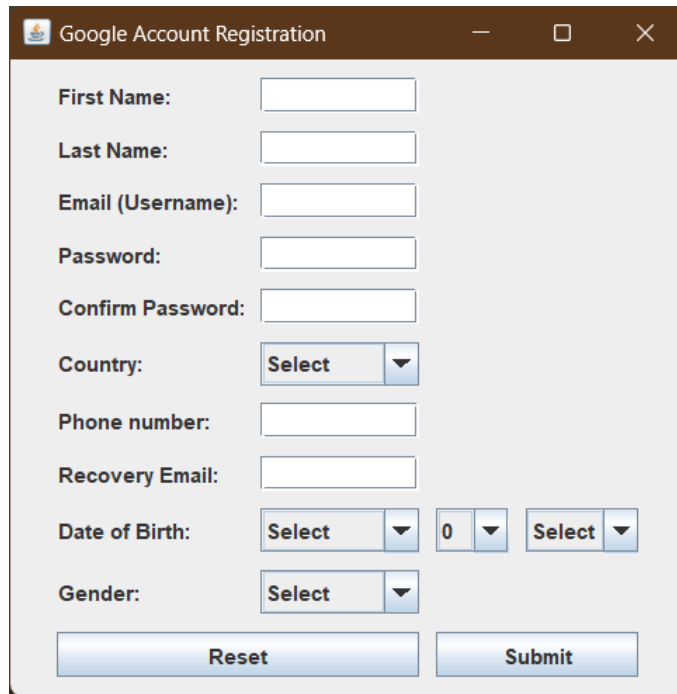
```

```

gender.getSelectedIndex() == 0) {
    throw new Exception("All fields are required.");
}
if (!isValidEmail(email.getText())) {
    throw new Exception("Invalid email address.");
}
if (!isValidEmail(recoveryEmail.getText())) {
    throw new Exception("Invalid recovery email address.");
}
String passwordStr = new String(password.getPassword());
String confirmPasswordStr = new String(confirmPassword.getPassword());
if (passwordStr.length() < 8) {
    throw new Exception("Password must be at least 8 characters long.");
}
if (!passwordStr.equals(confirmPasswordStr)) {
    throw new Exception("Passwords do not match.");
}
}
private static boolean isValidEmail(String email) {
    String regex = "^(.+)@(.+)\\.com$";
    Pattern pattern = Pattern.compile(regex);
    Matcher matcher = pattern.matcher(email);
    return matcher.matches();
}
private static int getMaxDaysForMonth(String month, int year) {
    switch (month) {
        case "April":
        case "June":
        case "September":
        case "November":
            return 30;
        case "February":
            if (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)) {
                return 29; // Leap year
            } else {
                return 28;
            }
        default:
            return 31;
    }
}
}
}

```

OUTPUT:



A screenshot of a Java Swing window titled "Google Account Registration". The window contains a registration form with the following fields: First Name, Last Name, Email (Username), Password, Confirm Password, Country (a dropdown menu with "Select" and a downward arrow), Phone number, Recovery Email, Date of Birth (a dropdown menu with "Select" and a downward arrow, followed by a year dropdown with "0" and a downward arrow, and another dropdown menu with "Select" and a downward arrow), and Gender (a dropdown menu with "Select" and a downward arrow). At the bottom of the form are two buttons: "Reset" and "Submit".



Two side-by-side screenshots of the registration form. The left screenshot shows the form with the following values: First Name: Vijai, Last Name: Suria, Email (Username): vijaisuriam@gmail.com, Password: [empty], Confirm Password: [empty], Country: [empty], Phone number: [empty], Recovery Email: vj@gmail.com, Date of Birth: June, 3, 2004, Gender: Male. A "Message" dialog box is displayed over the form, containing the text "Account successfully created!" and an "OK" button. The right screenshot shows the same form with the same values. An "Error" dialog box is displayed over the form, containing a red "X" icon and the text "All fields are required." with an "OK" button.

RESULT:

Thus, the programs to explore GUI in Java Programming was done successfully.

MULTITHREADING AND DEADLOCKS

Aim: To explore about multithreading in java programming

Algorithm:

Step 1: To display your name using thread.

1. `LocalDate.now()` - Obtains the current date from the system clock in the default time-zone
2. `LocalTime.now()` - Obtains the current time from the system clock in the default time-zone.
3. `println()` - A method used to print a line of text to the console.
4. `run()` - An abstract method from the `Runnable` interface that needs to be implemented by a class that intends to be executed by a thread.
5. `start()` - A method from the `Thread` class used to start the execution of a thread. It initiates the execution of the `run()` method of the thread.

Step 2: Difference between traditional and concurrent programming.

1. `currentTimeMillis()` - A static method of the `System` class in Java that returns the current time in milliseconds.
2. `start()` - A method from the `Thread` class used to start the execution of a thread.
3. `join()` - A method from the `Thread` class that makes sure the current thread waits until the thread on which it's called is dead.
4. `run()` - An abstract method from the `Runnable` interface that needs to be implemented by a class that intends to be executed by a thread.

Step 3: To show that return value of a thread can be caught and displayed.

1. `Scanner(System.in)` - Constructor for creating a new `Scanner` instance that takes input from the standard input stream (`System.in`). It's used to read user input from the console.
2. `Executors.newSingleThreadScheduledExecutor()` - A factory method from the `Executors` class used to create an `ExecutorService` that schedules commands to run after a given delay or to execute periodically in a single-threaded manner.
3. `submit(CallableInterface)` - Submits a `Callable` task for execution and returns a `Future` representing the pending completion of the task.

4. `Future.get()` - Waits if necessary for the computation to complete, and then retrieves the result of the computation. It's a method of the `Future` interface.
5. `task.shutdown()` - Initiates an orderly shutdown of the `ExecutorService`, previously created by `Executors.newSingleThreadScheduledExecutor()`. It allows previously submitted tasks to complete before terminating.

Step 4: To implement the thread pool concept to find prime numbers in 10 million numbers.

1. `Executors.newFixedThreadPool()` - Creates a thread pool that reuses a fixed number of threads. It's a method from the `Executors` class in the `java.util.concurrent` package.
2. `submit(Callable)` - Submits a `Callable` task for execution and returns a `Future` representing the pending result of the computation. It's a method of the `ExecutorService` interface.
3. `get()` - Waits if necessary for the computation to complete, and then retrieves the result of the computation. It's a method of the `Future` interface.
4. `shutdown()` - Initiates an orderly shutdown of the `ExecutorService`, previously created by `Executors.newFixedThreadPool()`. It allows previously submitted tasks to complete before terminating.
5. `Callable` interface - Represents a task that returns a result and may throw an exception. It's a functional interface with a method `call()`.
6. `Math.sqrt(number)` - Computes the square root of a given number. Used in the `isPrime()` method to optimize the prime checking logic.
7. `isPrime(int number)` - Helper method that checks if a given number is a prime number by applying a basic prime number checking algorithm. It returns true if the number is prime, otherwise false.

Step 5: To show the exception in main thread and user defined thread.

1. `Thread.sleep(millis)` - Causes the currently executing thread to sleep for the specified number of milliseconds. It's a static method in the `Thread` class used for pausing the execution of a thread.
2. `start()` - A method from the `Thread` class used to start the execution of a thread. It initiates the execution of the `run()` method of the thread.
3. `setName(String name)` - A method from the `Thread` class used to set the name of the thread.
4. `run()` - A method that contains the code to be executed by the thread. It is overridden from the `Thread` class in the `CarThread` class to define the behavior of the thread when started.

5. `method()` - A private method within the `CarThread` class that attempts to perform a division by zero (`5/0`). It intentionally triggers an `ArithmeticException` to demonstrate exception handling.
6. `try-catch` block - Constructs used for exception handling in Java. `try` is used to enclose the code that might throw an exception, and `catch` is used to catch and handle the exception when it occurs.
7. `e.printStackTrace()` - A method that prints the stack trace of the exception to the console. It's called on the Exception object (`e`) to display detailed information about the exception.

Step 6: To create a thread as Daemon and display the non-Daemon thread accordingly.

1. `Thread.setDaemon(boolean on)` - A method used to set a thread as a daemon or non-daemon. Daemon threads are background threads that do not prevent the JVM from exiting when they finish executing and are terminated if all non-daemon threads have finished.
2. `Thread.setName(String name)` - A method from the `Thread` class used to set the name of the thread.
3. `Thread.start()` - A method from the `Thread` class used to start the execution of a thread. It initiates the execution of the `run()` method of the thread.
4. `Thread.currentThread()` - A static method of the `Thread` class that returns a reference to the currently executing thread object.
5. `isDaemon()` - A method from the `Thread` class used to check whether the current thread is a daemon thread.
6. `run()` - A method that contains the code to be executed by the thread. It's overridden from the `Thread` class in the `Daemon3024` class to define the behavior of the thread when started.

Step 7: To access the file by more than one thread and display the content using synchronization.

1. `Files.readAllBytes(Path path)` - A method from the `Files` class in the `java.nio.file` package used to read all the bytes from a file as a byte array.
2. `String(byte[] bytes)` - A constructor for creating a new `String` object by decoding the specified array of bytes using the platform's default character set.
3. `Thread(Runnable target)` - A constructor to create a new thread that executes the specified `Runnable` target.
4. `synchronized (lock)` - A synchronized block used to provide mutual exclusion so that only one thread can execute the block at a time. Here, `lock` is used as a monitor object for synchronization.

5. `start()` - A method from the `Thread` class used to start the execution of a thread. It initiates the execution of the `run()` method of the thread.
6. `e.printStackTrace()` - A method that prints the stack trace of an exception to the console. It's called on the `IOException` object (`e`) to display detailed information about the exception occurrence if file reading fails.

10.1 To display your name using thread

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
class Name implements Runnable{
    public void run(){
        System.out.println("My name is Vijai Suria M, using Runnable interface");
    }
}
class Names extends Thread {
    public void run(){
        System.out.println("My name is Vijai Suria M, using Thread class");
    }
}
public class Name3568 {
    public static void main(String[] args){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)
\n");
        Name obj = new Name();
        Thread t1 = new Thread(obj);
        Thread t2 = new Thread(() -> {System.out.println("My name is Vijai Suria
M, using Lambda function");});
        Thread t3 = new Names();
        t1.start();
        t2.start();
        t3.start();
    }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:
Current Date: 2023-11-08
Current Time: 20:48:28.583537800
Name: Vijai Suria M
Register Number: (2021503568)

My name is Vijai Suria M, using Runnable interface
My name is Vijai Suria M, using Lambda function
My name is Vijai Suria M, using Thread class
```

10.2 Difference between traditional and concurrent programming.

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.lang.Thread;
class Myclass implements Runnable
{
    public void run()
    {
        double result=0;
        for (int i = 0; i< 100000099; i++)
        {
            result = (i / 1.234566) * 1234.567890988;
        }
        System.out.println(result);
    }
}

public class Concurrent3568
{
    public static void main(String [] arg) throws Exception
    {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        final long startTime=System.currentTimeMillis();
        double result=0;
        for(long i=0; i< 100000099; i++)
```

```

        {
            result=(i/1.234566)*1234.567890988;
        }

System.out.println(result);
System.out.println("Total Time (using traditional method) : " + (endTime-
startTime));
    final long sTime = System.currentTimeMillis();
    Thread t1 = new Thread(new Myclass());
    Thread t2 = new Thread(new Myclass());
    t1.start();
    t2.start();
    if(t1.isAlive())
        t1.join();
    if(t2.isAlive())
        t2.join();
    final long eTime = System.currentTimeMillis();
System.out.println("Total Time (using multithreading) : " + (eTime-sTime));
    }
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-08
Current Time: 16:01:50.433310200
Name: Vijai Suria M
Register Number: (2021503568)

1.0000025117041399E11
Total Time (using traditional method) : 453
1.0000025117041399E11
1.0000025117041399E11
Total Time (using multithreading) : 8

```

10.3 To show that return value of a thread can be caught and displayed.

CODE:

```
import java.util.Scanner;
import java.util.concurrent.*;
import java.time.*;
class CallableInterface implements Callable<String> {
    public String call(){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter your name: ");
        return in.nextLine();
    }
}
public class Callable3568 {
    public static void main(String[] args) throws Exception{
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) ");
        ExecutorService task = Executors.newSingleThreadScheduledExecutor();
        Future<String> name = task.submit(new CallableInterface());
        System.out.println("Name returned by Callable interface: " + name.get());
        task.shutdown();
    }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-08
Current Time: 16:29:23.530403400
Name: Vijai Suria M
Register Number: (2021503568)

Enter your name:
Vijai
Name return by Callable interface: Vijai

Process finished with exit code 0
```

10.4 To implement the thread pool concept to find prime numbers in 10 million numbers.

CODE:

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.Callable;
import java.time.*;

public class PrimeFind3568 {
    public static void main(String[] args) throws Exception {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        int totalNumbers = 1000000;
        int threadCount = 5;
        int rangeSize = totalNumbers / threadCount;
        ExecutorService executorService =
            Executors.newFixedThreadPool(threadCount);
        Future<Integer>[] results = new Future[threadCount];
        for (int i = 0; i < threadCount; i++) {
            final int startRange = i * rangeSize + 1;
            final int endRange = (i + 1) * rangeSize;
            results[i] = executorService.submit(new PrimeCounter(startRange,
                endRange));
            System.out.printf("Thread %d: Number of primes from %d to %d: %d", i+1,
                startRange, endRange, results[i].get());
            System.out.println();
        }
        int totalPrimes = 0;
        for (int i = 0; i < threadCount; i++) {
            totalPrimes += results[i].get();
        }
        executorService.shutdown();
        System.out.println("\nTotal prime numbers in the range 1 to 1,000,000: " +
            totalPrimes);
    }
    static class PrimeCounter implements Callable<Integer> {
        private final int start;
        private final int end;
```



```

PrimeCounter(int start, int end) {
this.start = start;
this.end = end;
}
@Override
public Integer call() {
    int count = 0;
    for (int num = start; num<= end; num++) {
        if (isPrime(num)) {
            count++;
        }
    }
    return count;
}
// Helper function to check if a number is prime
private boolean isPrime(int number) {
    if (number <= 1) {
        return false;
    }
    for (int i = 2; i<= Math.sqrt(number); i++) {
        if (number % i == 0) {
            return false;
        }
    }
    return true;
}
}
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Pr
Current Date: 2023-11-08
Current Time: 21:15:03.061519900
Name: Vijai Suria M
Register Number: (2021503568)

Thread 1: Number of primes from 1 to 200000: 17984
Thread 2: Number of primes from 200001 to 400000: 15876
Thread 3: Number of primes from 400001 to 600000: 15238
Thread 4: Number of primes from 600001 to 800000: 14853
Thread 5: Number of primes from 800001 to 1000000: 14547

Total prime numbers in the range 1 to 1,000,000: 78498

```

10.5 To show the exception in main thread and user defined thread.

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;

class CarThread extends Thread{
    private void method(){
        try {
            System.out.println(5/0);
        }
        catch(ArithmeticException e){
            System.out.println("Caught Arithmetic Exception inside " +
                Thread.currentThread().getName() + " thread");
            e.printStackTrace();
        }
    }
    public void run(){
        System.out.println("\nI am a thread");
        method();
    }
}

public class Exception3568 {
    public static void main(String[] args) throws InterruptedException{
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try {
            System.out.println(5/0);
        }
        catch (ArithmeticException e){
            System.out.println("caught Exception Inside the " + Thread.currentThread().getName()
                + " thread");
            e.printStackTrace();
        }
        Thread.sleep(1000);
        CarThread BMW = new CarThread();
        BMW.setName("BMW");
        BMW.start();
    }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program
Current Date: 2023-11-08
Current Time: 21:08:00.160006
Name: Vijai Suria M
Register Number: (2021503568)

caught Exception Inside the main thread
java.lang.ArithmeticException Create breakpoint : / by zero
    at Exception3568.main(Exception3568.java:25)

I am a thread
Caught Arithmetic Exception inside BMW thread
java.lang.ArithmeticException Create breakpoint : / by zero
    at CarThread.method(Exception3568.java:7)
    at CarThread.run(Exception3568.java:16)
```

10.6 To create a thread as Daemon and display the non-Daemon thread accordingly.

CODE:

```
import java.time.*;
public class Daemon3568 extends Thread{
    public void run(){
        if(Thread.currentThread().isDaemon()){//checking for daemon thread
            System.out.println("\nI am a Daemon thread\nThread name: " +
                Thread.currentThread().getName());
        }
        else{
            System.out.println("I am a user-defined thread");
        }
    }
    public static void main(String[] args){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        Daemon3568 t1=new Daemon3568();//creating thread
        Daemon3568 t2=new Daemon3568();
        Daemon3568 t3=new Daemon3568();
        t1.setDaemon(true);//now t1 is daemon thread
        t1.setName("Daemon");
        t1.start();//starting threads
        t2.start();
        t3.start();
    }
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "  
Current Date: 2023-11-08  
Current Time: 21:26:33.701937400  
Name: Vijai Suria M  
Register Number: (2021503568)  
  
I am a user-defined thread  
I am a user-defined thread  
  
I am a Daemon thread  
Thread name: Daemon
```

10.7 To access the file by more than one thread and display the content using synchronization.

CODE:

```
import java.nio.file.*;  
import java.io.IOException;  
import java.time.*;  
public class FileThread3568 {  
    private static Path filePath= Paths.get("C:\\Users\\vijai\\Documents\\JAVA-  
PROGRAMMING/Threads/src/demo.txt");  
    private static final Object lock = new Object();  
    public static void main(String[] args) {  
        System.out.println("Current Date: " + LocalDate.now());  
        System.out.println("Current Time: " + LocalTime.now());  
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");  
        Thread t1 = new Thread(() -> {  
            synchronized (lock) {  
                try {  
                    byte[] fileContent = Files.readAllBytes(filePath);  
                    String content = new String(fileContent);  
                    System.out.println("Thread t1: File Content -\n" + content);  
                } catch (IOException e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
        Thread t2 = new Thread(() -> {  
            synchronized (lock) {  
                try {  
                    byte[] fileContent = Files.readAllBytes(filePath);
```

```

        String content = new String(fileContent);
        System.out.println("Thread t2: File Content -\n" + content);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
});
t1.start();
t2.start();
}
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-08
Current Time: 21:48:48.150595600
Name: Vijai Suria M
Register Number: (2021503568)

Thread t1: File Content -
My name is Vijai Suria
Thread t2: File Content -
My name is Vijai Suria

```

10.8 To write a java program to demonstrate the multi-threading concept in movie ticket booking scenario.

CODE:

```

import static java.lang.Thread.currentThread;
import java.time.*;
public class MovieTicket {
    public static int ticketCount = 10;
    public static void main(String[] a) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        Thread user1 = new Thread(() -> bookTicket("J", 2));
        Thread user2 = new Thread(() -> bookTicket("S", 2));
        Thread user3 = new Thread(() -> bookTicket("K", 2));
        Thread user31 = new Thread(() -> bookTicket("E", 2));
        Thread user32 = new Thread(() -> bookTicket("P", 2));
        Thread user323 = new Thread(() -> bookTicket("U", 2));
        user1.start();
    }
}

```

```

        user2.start();
        user3.start();
        user31.start();
        user32.start();
        user323.start();
    }
    public static void bookTicket(String user, int bookingCount) {
        if (bookingCount > ticketCount) {
            System.out.println("Ticket Full");
            System.out.println(user + " " + "Sorry ticket not booked count " + bookingCount);
        }
        else {
            if (ticketCount >= bookingCount) {
                ticketCount -= bookingCount;
                System.out.println(user + " " + "ticket booked count " + bookingCount);
                System.out.println("total remaining" + ticketCount);
            }
        }
    }
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:16:40.145512800
Name: Vijai Suria M
Register Number: (2021503568)

Ticket Full
S ticket booked count 2
J ticket booked count 2
U Sorry ticket not booked count 2
E ticket booked count 2
K ticket booked count 2
P ticket booked count 2
total remaining0
total remaining0
total remaining0
total remaining0

```

10.9 To demonstrate produce-consumer problem of a shopping mall

CODE:

```
import java.time.*;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

class Customer {
    private String name;
    private String item;

    public Customer(String name, String item) {
        this.name = name;
        this.item = item;
    }

    public String getName() {
        return name;
    }

    public String getItem() {
        return item;
    }
}

public class ShoppingExample {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        BlockingQueue<Customer> shoppingCart = new ArrayBlockingQueue<>(5);
        // Producer thread (customers adding items to the shopping cart)
        Thread producer = new Thread(() -> {
            try {
                shoppingCart.put(new Customer("Alice", "Shoes"));
                shoppingCart.put(new Customer("Bob", "T-shirt"));
                shoppingCart.put(new Customer("Charlie", "Jeans"));
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        });
        // Consumer thread (checkout process)
        Thread consumer = new Thread(() -> {
            try {
                while (true) {
```

```

        Customer customer = shoppingCart.take();
        System.out.println("Checkout for " + customer.getName() + ": " +
customer.getItem());
        // Simulate the checkout process (payment and order processing)
        Thread.sleep(1000);
    }
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}
});

producer.start();
consumer.start();
}
}

```

-

OUTPUT:

```

"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:19:06.676815800
Name: Vijai Suria M
Register Number: (2021503568)

Checkout for Alice: Shoes
Checkout for Bob: T-shirt
Checkout for Charlie: Jeans

```

10.10 To demonstrate the deadlock condition in java (choosing a coin in multi-player mode)

CODE:

```

import java.time.*;
class Coin{
    public static boolean pickUp=false;
}
public class DeadLock {
    public static void main(String a[]){
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        Thread T1=new Thread()->{Player1("Player 1");
        });
    }
}

```



```

Thread T2=new Thread(()->{Player2("Player 2");
});
T1.start();
T2.start();
}
public static void Player1(String Player) {
    while(!Coin.pickUp) {
        Coin.pickUp=true;
        System.out.println(Player + " trying to pickup Coin");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
        }
        System.out.println(Player + " have pickedup the Coin");
        Coin.pickUp=false;
    }
    System.out.println(Player + "waiting for");
}
public static void Player2(String Player) {
    while(!Coin.pickUp) {
        Coin.pickUp=true;
        System.out.println(Player + " trying to pickup Coin");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
        }
        System.out.println(Player + " have pickedup the Coin");
        Coin.pickUp=false;
    }
    System.out.println(Player + "waiting for");
}
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:21:22.361082200
Name: Vijai Suria M
Register Number: (2021503568)

Player 1waiting for
Player 2 trying to pickup Coin
Player 2 have pickedup the Coin
Player 2 trying to pickup Coin

```

10.11 To illustrate producer consumer problem in java programming

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

class User {
    private String name;
    private String item;

    public User(String name, String item) {
        this.name = name;
        this.item = item;
    }

    public String getName() {
        return name;
    }

    public String getItem() {
        return item;
    }
}

public class ProducerConsumer {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");

        BlockingQueue<User> shoppingCart = new ArrayBlockingQueue<>(5);

        // Producer thread (customers adding items to the shopping cart)
        Thread producer = new Thread(() -> {
            try {
                shoppingCart.put(new User("User 1", "Shoes"));
                shoppingCart.put(new User("User 2", "T-shirt"));
                shoppingCart.put(new User("User 3", "Jeans"));
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        });
    }
}
```

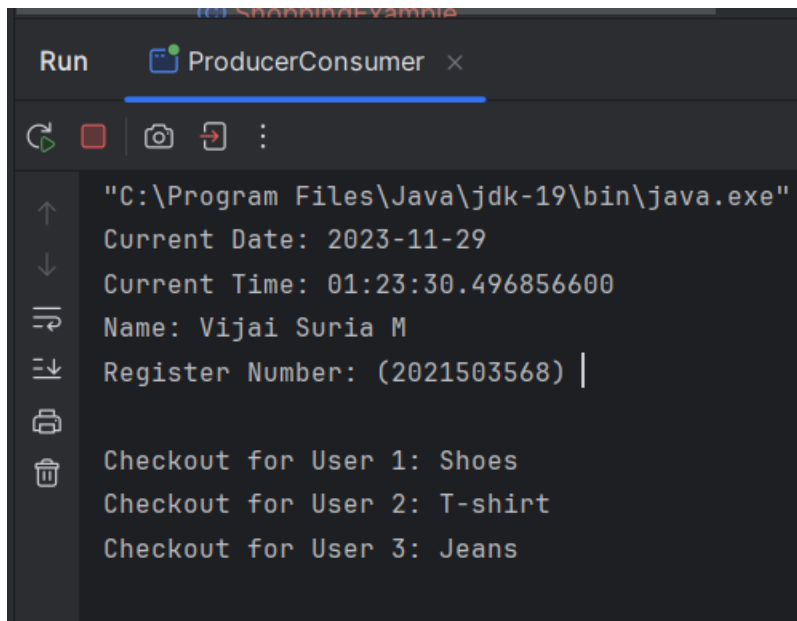
```

// Consumer thread (checkout process)
Thread consumer = new Thread() -> {
    try {
        while (true) {
            User customer = shoppingCart.take();
            System.out.println("Checkout for " + customer.getName() + ": " +
customer.getItem());
            Thread.sleep(1000);
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
});

producer.start();
consumer.start();
}
}

```

OUTPUT:



```

Run  ProducerConsumer x
"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:23:30.496856600
Name: Vijai Suria M
Register Number: (2021503568) |
Checkout for User 1: Shoes
Checkout for User 2: T-shirt
Checkout for User 3: Jeans

```

10.12 To make use of semaphore to control synchronization in java

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.concurrent.Semaphore;
public class SemaPoresExample {
    public static Semaphore share=new Semaphore(2);

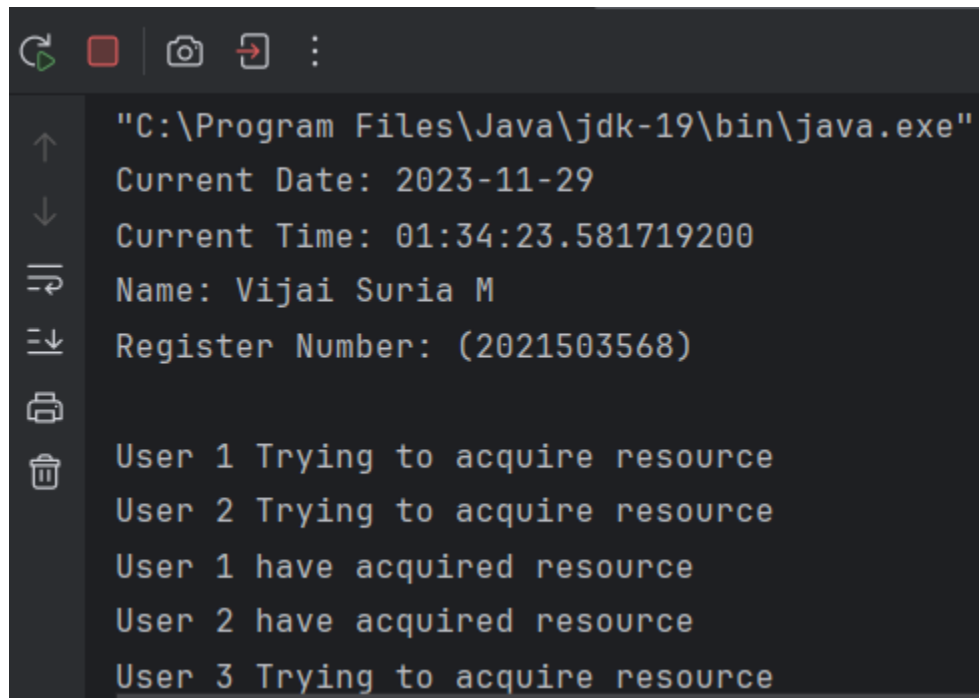
    public static void main(String a[]){
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)
\n");

        new Thread(()->ShareResource(),"User 1").start();
        new Thread(()->ShareResource(),"User 2").start();
        new Thread(()->ShareResource(),"User 3").start();
        new Thread(()->ShareResource(),"User 4").start();
        new Thread(()->ShareResource(),"User 5").start();
        new Thread(()->ShareResource(),"User 6").start();
        new Thread(()->ShareResource(),"User 7").start();
    }

    public static void ShareResource(){
        try {
            share.acquire();
            System.out.println(Thread.currentThread().getName() + " Trying to acquire
resource");
            Thread.sleep(1000);
            System.out.println(Thread.currentThread().getName() + " have acquired
resource");
            Thread.sleep(1000);
            share.release();
            System.out.println(Thread.currentThread().getName() + " have released
resource");
        }
        catch (InterruptedException e){
            Thread.currentThread().interrupt();
        }
    }
}
```

```
}  
}
```

OUTPUT:



The screenshot shows a Java IDE console window with a dark background. The output text is as follows:

```
"C:\Program Files\Java\jdk-19\bin\java.exe"  
Current Date: 2023-11-29  
Current Time: 01:34:23.581719200  
Name: Vijai Suria M  
Register Number: (2021503568)  
  
User 1 Trying to acquire resource  
User 2 Trying to acquire resource  
User 1 have acquired resource  
User 2 have acquired resource  
User 3 Trying to acquire resource
```

RESULT:

Thus, the concepts of multithreading in java programming is implemented and executed successfully.

SOCKET PROGRAMMING USING TCP & UDP

Aim: To explore about Socket Programming in Java

Algorithm:

Step 1: TCP Socket Client for square root

1. **Socket(String, int)** - Connects to a server at the specified address and port.
2. **Scanner(System.in)** - Reads user input from the console.
3. **DataInputStream(InputStream)** - Reads primitive data types from an input stream.
4. **DataOutputStream(OutputStream)** - Writes primitive data types to an output stream.
5. **writeDouble(double)** - Writes a double value to the output stream.
6. **nextDouble()** - Reads a double value from the console.

Step 2: TCP Socket Server for square root

1. **ServerSocket(int)** - Creates a server socket on the specified port.
2. **accept()** - Listens for a connection to be made to this socket and accepts it.
3. **DataInputStream(InputStream)** - Reads primitive data types from an input stream.
4. **DataOutputStream(OutputStream)** - Writes primitive data types to an output stream.
5. **readDouble()** - Reads a double value from the input stream.
6. **writeDouble(double)** - Writes a double value to the output stream.
7. **Math.sqrt(double)** - Returns the square root of a double value.

Step 3: UDP Socket Client for square root

1. **DatagramSocket(int)** - Creates a DatagramSocket to listen on the specified port.
2. **DatagramPacket(byte[], int)** - Initializes a DatagramPacket for receiving data.
3. **sk.receive(DatagramPacket)** - Receives a DatagramPacket from the client.
4. **Double.parseDouble(String)** - Converts the received data to a double.
5. **Math.sqrt(double)** - Calculates the square root of a double.
6. **InetAddress** - Represents the client's address.
7. **send(DatagramPacket)** - Sends a DatagramPacket back to the client.

Step 4: UDP Socket Server for square root

1. **DatagramSocket()** - Creates a DatagramSocket for the client.
2. **DatagramPacket(byte[], int, InetAddress, int)** - Initializes a DatagramPacket for sending data.
3. **send(DatagramPacket)** - Sends a DatagramPacket to the server.
4. **receive(DatagramPacket)** - Receives a DatagramPacket from the server.
5. **Double.parseDouble(String)** - Converts the received data to a double.
6. **Scanner(System.in)** - Reads user input from the console.
7. **InetAddress** - Represents the server's address.

Step 5: TCP Client Socket to sort the array of inputs

1. **Socket(String, int)** - Connects to a server at the specified address and port.
2. **Scanner(System.in)** - Reads user input from the console.
3. **DataInputStream(InputStream)** - Reads primitive data types from an input stream.
4. **DataOutputStream(OutputStream)** - Writes primitive data types to an output stream.
5. **writeInt(int)** - Writes an integer value to the output stream.
6. **nextInt()** - Reads an integer value from the console.

Step 6: TCP Server Socket to sort the array of inputs

1. **ServerSocket(int)** - Creates a server socket on the specified port.
2. **serverSocket.accept()** - Listens for a connection to be made to this socket and accepts it.
3. **DataInputStream(InputStream)** - Reads primitive data types from an input stream.
4. **DataOutputStream(OutputStream)** - Writes primitive data types to an output stream.
5. **readInt()** - Reads an integer value from the input stream.
6. **Arrays.sort(int[])** - Sorts an array of integers in ascending order.
7. **writeInt(int)** - Writes an integer value to the output stream.

Step 7: UDP Client Socket to sort the array of inputs

1. **DatagramSocket()** - Creates a DatagramSocket for the client.
2. **InetAddress.getLocalHost()** - Gets the local host address.
3. **DatagramPacket(byte[], int, InetAddress, int)** - Initializes a DatagramPacket for sending data.
4. **send(DatagramPacket)** - Sends a DatagramPacket to the server.
5. **DatagramPacket(byte[], int)** - Initializes a DatagramPacket for receiving data.
6. **socket.receive(DatagramPacket)** - Receives a DatagramPacket from the server.
7. **String(byte[], int, int)** - Converts received data to a string.

Step 8: UDP Server Socket to sort the array of inputs

1. **DatagramSocket(int)** - Creates a DatagramSocket for the server on the specified port.
2. **socket.receive(DatagramPacket)** - Receives a DatagramPacket from the client.
3. **String(byte[], int, int)** - Converts received data to a string.
4. **parseInt(String)** - Parses the string representation of an integer.
5. **Arrays.sort(int[])** - Sorts an array of integers in ascending order.
6. **Arrays.toString(int[])** - Converts an array of integers to a string representation.

7. **DatagramPacket(byte[], int, InetAddress, int)** - Initializes a DatagramPacket for sending data.
8. **socket.send(DatagramPacket)** - Sends a DatagramPacket back to the client.

Step 9: TCP Socket Client Chat Using Multiclient

1. **Socket(String, int)** - Connects to a server at the specified address and port.
2. **BufferedReader(InputStreamReader)** - Reads text from an input stream.
3. **PrintWriter(OutputStream, boolean)** - Prints formatted representations of objects to an output stream.
4. **readLine()** - Reads a line of text from the console or input stream.
5. **println(String)** - Prints a string to the output stream.

Step 10: TCP Socket Server Chat Using Multiclient

1. **ServerSocket(int)** - Creates a server socket on the specified port.
2. **serverSocket.accept()** - Listens for a connection to be made to this socket and accepts it.
3. **BufferedReader(InputStreamReader)** - Reads text from an input stream.
4. **PrintWriter(OutputStream, boolean)** - Prints formatted representations of objects to an output stream.
5. **handleClient(Socket)** - Handles communication with a connected client in a separate thread.

Step 11: UDP Socket Client Chat Using Multiclient

1. **DatagramSocket()** - Creates a DatagramSocket for the client.
2. **InetAddress.getLocalHost()** - Gets the local host address.
3. **DatagramPacket(byte[], int, InetAddress, int)** - Initializes a DatagramPacket for sending data.
4. **socket.receive(DatagramPacket)** - Receives a DatagramPacket from the server.
5. **String(byte[], int, int)** - Converts received data to a string.

Step 12: UDP Socket Client Chat Using Multiclient

1. **DatagramSocket(int)** - Creates a DatagramSocket for the server on the specified port.
2. **DatagramPacket(byte[], int)** - Initializes a DatagramPacket for receiving data.
3. **socket.receive(DatagramPacket)** - Receives a DatagramPacket from the client.
4. **String(byte[], int, int)** - Converts received data to a string.
5. **BufferedReader(InputStreamReader)** - Reads text from an input stream.
6. **new Thread(() -> { ... }).start()** - Starts a new thread to handle server input independently.

Step 13: TCP Socket Client for Movie ticket booking system using multi-client

1. **Socket(String, int)** - Connects to the server at the specified IP address and port.
2. **BufferedReader(InputStreamReader)** - Reads text from an input stream.
3. **PrintWriter(OutputStream, boolean)** - Prints formatted representations of objects to an output stream.
4. **readLine()** - Reads a line of text from the console.
5. **println(String)** - Prints a string to the output stream.

Step 14: TCP Socket Server for Movie ticket booking system using multi-client

1. **ServerSocket(int)** - Creates a server socket on the specified port.
2. **serverSocket.accept()** - Listens for a connection to be made to this socket and accepts it.
3. **ExecutorService** - Manages and controls the execution of threads.
4. **Executors.newFixedThreadPool(int)** - Creates a thread pool with a fixed number of threads.
5. **Runnable.run()** - The method to be executed in a separate thread.
6. **BufferedReader(InputStreamReader)** - Reads text from an input stream.
7. **PrintWriter(OutputStream, boolean)** - Prints formatted representations of objects to an output stream.

11.1) TCP/UDP Socket to get the square root of the given number,

TCP Server (finding square root):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class SquareRootServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        try {
            ServerSocket serverSocket = new ServerSocket(9999); // Port to listen on
            System.out.println("Server started. Waiting for a client...");
            Socket clientSocket = serverSocket.accept(); // Accept incoming connection
            System.out.println("Client connected.");
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                double number = Double.parseDouble(inputLine);
                double squareRoot = Math.sqrt(number);
                out.println("Square root of " + number + " is: " + squareRoot);
            }
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TCP Client (finding square root):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class SquareRootClient3568 {
    public static void main(String[] args) {
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try {
```

```

        Socket socket = new Socket("localhost", 9999); // Connect to server
        BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        System.out.print("Enter a number to find its square root: ");
        String number = userInput.readLine();
        out.println(number); // Send number to server
        String serverResponse = in.readLine(); // Receive square root from server
        System.out.println("Server response: " + serverResponse);
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

OUTPUT (finding square root):

TCP Client:

```

Run    SquareRootServer3568 x    SquareRootClient3568 x
"\"C:\\Program Files\\Java\\jdk-18.0.2.1\\bin\\java.exe\" \"-javaagent:
Current Date: 2023-11-22
Current Time: 16:27:36.413275200
Name: Vijai Suria M
Register Number: (2021503568)
Enter a number to find its square root: 81
Server response: Square root of 81.0 is: 9.0
Process finished with exit code 0

```

TCP Server:

```

Run    SquareRootServer3568 x    SquareRootClient3568 x
"\"C:\\Program Files\\Java\\jdk-18.0.2.1\\bin\\java.exe\" \"-javaagent:
Current Date: 2023-11-22
Current Time: 16:27:30.149811300
Server started. Waiting for a client...
Client connected.
Process finished with exit code 0

```

UDP Server (finding square root):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;

public class SquareRootUDPServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");

        try {
            DatagramSocket serverSocket = new DatagramSocket(9999); // Port to listen on
            System.out.println("Server started. Waiting for a client...");

            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];

            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);
                String numberString = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                double number = Double.parseDouble(numberString);

                double squareRoot = Math.sqrt(number);
                String response = "Square root of " + number + " is: " + squareRoot;
                sendData = response.getBytes();

                InetAddress clientIP = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();
                DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, clientIP, clientPort);

                serverSocket.send(sendPacket);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

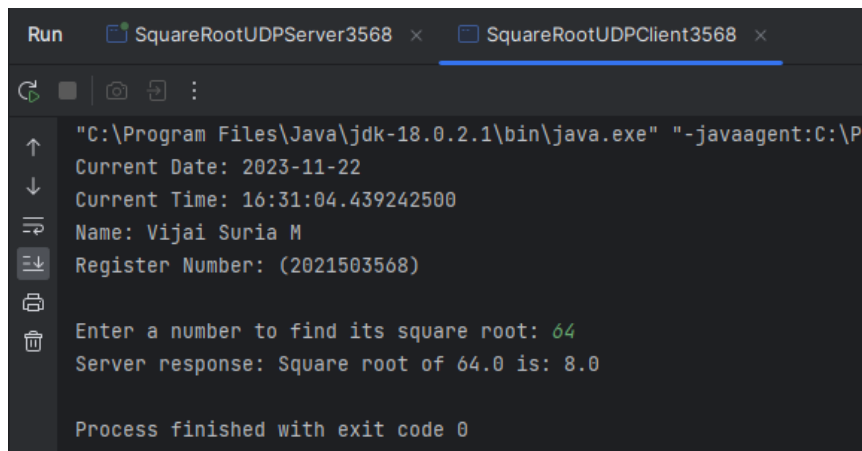
UDP Client (finding square root):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;

public class SquareRootUDPCClient3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress serverIP = InetAddress.getByName("localhost");
            byte[] sendData = new byte[1024];
            byte[] receiveData = new byte[1024];
            BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Enter a number to find its square root: ");
            String number = userInput.readLine();
            sendData = number.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverIP, 9999);
            clientSocket.send(sendPacket);
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            clientSocket.receive(receivePacket);
            String serverResponse = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Server response: " + serverResponse);
            clientSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT (finding square root):

UDP Client:

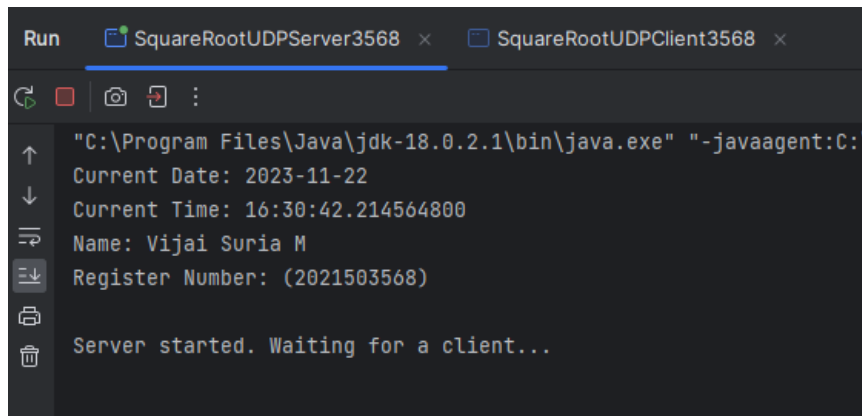


```
Run  SquareRootUDPServer3568 x  SquareRootUDPClient3568 x
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Pr
Current Date: 2023-11-22
Current Time: 16:31:04.439242500
Name: Vijai Suria M
Register Number: (2021503568)

Enter a number to find its square root: 64
Server response: Square root of 64.0 is: 8.0

Process finished with exit code 0
```

UDP Server:



```
Run  SquareRootUDPServer3568 x  SquareRootUDPClient3568 x
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\
Current Date: 2023-11-22
Current Time: 16:30:42.214564800
Name: Vijai Suria M
Register Number: (2021503568)

Server started. Waiting for a client...
```

11.2) TCP/UDP socket to sort the array of inputs,

TCP Server (sorting array of inputs):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.Arrays;
public class TCPSortServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try {
            ServerSocket serverSocket = new ServerSocket(9999); // Port to listen on
```



```

        System.out.println("Server started. Waiting for a client...");
        Socket clientSocket = serverSocket.accept(); // Accept incoming connection
        System.out.println("Client connected.");
        BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
        String inputLine = in.readLine();
        String[] numbersArray = inputLine.split(" ");
        int[] intArray =
Arrays.stream(numbersArray).mapToInt(Integer::parseInt).toArray();
        Arrays.sort(intArray);
        String sortedNumbers = Arrays.toString(intArray);
        out.println(sortedNumbers);
        clientSocket.close();
        serverSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

TCP Client (sorting array of inputs):

```

import java.io.*;
import java.net.*;
import java.util.Arrays;
import java.time.LocalDate;
import java.time.LocalTime;
public class TCPSortClient3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try {
            Socket socket = new Socket("localhost", 9999); // Connect to server
            BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in));
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            System.out.print("Enter numbers separated by spaces to be sorted: ");
            String numbers = userInput.readLine();
            out.println(numbers); // Send numbers to server

```

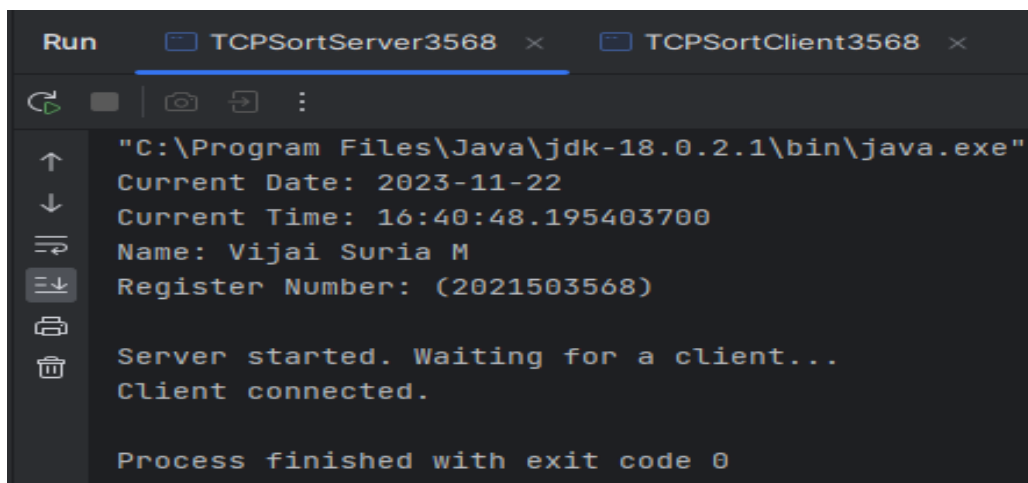
```

        String sortedNumbers = in.readLine(); // Receive sorted numbers from server
        int[] sortedArray = Arrays.stream(sortedNumbers.substring(1,
sortedNumbers.length() - 1).split(", "))
            .mapToInt(Integer::parseInt).toArray();
        System.out.print("Sorted array received from server: ");
        for (int num : sortedArray) {
            System.out.print(num + " ");
        }
        System.out.println();
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

OUTPUT (sorting array of inputs):

TCP Server:



```

Run    TCP Sort Server 3568 x    TCP Sort Client 3568 x
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-22
Current Time: 16:40:48.195403700
Name: Vijai Suria M
Register Number: (2021503568)

Server started. Waiting for a client...
Client connected.

Process finished with exit code 0

```

TCP Client:



```

Run    TCP Sort Server 3568 x    TCP Sort Client 3568 x
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Program Fil
Current Date: 2023-11-22
Current Time: 16:40:53.938203600
Name: Vijai Suria M
Register Number: (2021503568)

Enter numbers separated by spaces to be sorted: 9 89 99 -100 0 1 2 3
Sorted array received from server: -100 0 1 2 3 9 89 99

```

UDP Server (sorting array of inputs):

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Arrays;
import java.time.LocalDate;
import java.time.LocalTime;

public class UDPSortServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)
\n");
        try {
            DatagramSocket serverSocket = new DatagramSocket(9999); // Port to
listen on
            System.out.println("Server started. Waiting for a client...");

            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];

            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);
                ByteArrayInputStream byteStream = new
ByteArrayInputStream(receivePacket.getData());
                ObjectInputStream objInput = new ObjectInputStream(byteStream);
                int[] receivedArray = (int[]) objInput.readObject();
                Arrays.sort(receivedArray);
                ByteArrayOutputStream byteStreamOut = new
ByteArrayOutputStream();
                ObjectOutputStream objOutput = new
ObjectOutputStream(byteStreamOut);
                objOutput.writeObject(receivedArray);
                sendData = byteStreamOut.toByteArray();

                InetAddress clientIP = receivePacket.getAddress();
```

```

        int clientPort = receivePacket.getPort();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
        sendData.length, clientIP, clientPort);

        serverSocket.send(sendPacket);
    }
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
}
}
}
}

```

UDP Client (sorting array of inputs):

```

import java.io.*;
import java.net.*;
import java.util.Arrays;
import java.time.LocalDate;
import java.time.LocalTime;

public class UDPSortClient3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)
\n");
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress serverIP = InetAddress.getByName("localhost");
            byte[] sendData = new byte[1024];
            byte[] receiveData = new byte[1024];

            BufferedReader userInput = new BufferedReader(new
            InputStreamReader(System.in));

            System.out.print("Enter numbers separated by spaces to be sorted: ");
            String numbers = userInput.readLine();
            String[] numbersArray = numbers.split(" ");
            int[] intArray =
            Arrays.stream(numbersArray).mapToInt(Integer::parseInt).toArray();

            ByteArrayOutputStream byteStream = new ByteArrayOutputStream();
            ObjectOutputStream objOutput = new ObjectOutputStream(byteStream);
            objOutput.writeObject(intArray);
            sendData = byteStream.toByteArray();

```

```

        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverIP, 9999);
        clientSocket.send(sendPacket);

        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);

        ByteArrayInputStream byteStreamIn = new
ByteArrayInputStream(receivePacket.getData());
        ObjectInputStream objInput = new ObjectInputStream(byteStreamIn);

        int[] sortedArray = (int[]) objInput.readObject();
        System.out.print("Sorted array received from server: ");
        for (int num : sortedArray) {
            System.out.print(num + " ");
        }
        System.out.println();

        clientSocket.close();
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
}
}

```

OUTPUT (sorting array of inputs):

UDP Server:

```

Run  UDPSortServer3568 x  UDPSortClient3568 x
"\"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe\" \"-javaager
Current Date: 2023-11-22
Current Time: 16:34:16.405691200
Name: Vijai Suria M
Register Number: (2021503568)

Server started. Waiting for a client...
|

```

UDP Client:



```
Run  UDPSortServer3568  x  UDPSortClient3568  x

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Progra
Current Date: 2023-11-22
Current Time: 16:34:21.752367300
Name: Vijai Suria M
Register Number: (2021503568)

Enter numbers separated by spaces to be sorted: 45 67 -1 0 9
Sorted array received from server: -1 0 9 45 67

Process finished with exit code 0
```

11.3) TCP/UDP chat using multiclient

TCP Multi-client chat Server:

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
public class TCPServer3568 {
    private static final int PORT = 8888;
    private static final Set<Socket> clientSockets = new HashSet<>();
    private static final Map<Socket, String> clientNames = new HashMap<>();
    public static void main(String[] args) {
        System.out.println("Date:" + LocalDate.now());
        System.out.println("Time:" + LocalDateTime.now());
        System.out.println("NAME: Vijai Suria .M \nReg No:2021503568");
        System.out.println("\n");
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server started. Waiting for clients...");
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected: " + clientSocket);
                clientSockets.add(clientSocket);
                Thread clientThread = new Thread(new ClientHandler(clientSocket));
                clientThread.start();
            }
        }
    }
}
```

```

    } catch (IOException e) {
        e.printStackTrace();
    }
}

static class ClientHandler implements Runnable {
    private final Socket clientSocket;
    private PrintWriter writer;
    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
    }
    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            writer = new PrintWriter(clientSocket.getOutputStream(), true);
            writer.println("Enter your name:");
            String name = reader.readLine();
            clientNames.put(clientSocket, name);
            broadcastMessage(name + " joined the chat");
            String clientMessage;
            while ((clientMessage = reader.readLine()) != null) {
                broadcastMessage(name + ": " + clientMessage);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            clientNames.remove(clientSocket);
            clientSockets.remove(clientSocket);
            broadcastMessage(clientNames.getOrDefault(clientSocket, "Anonymous") + "
left the chat");
            try {
                clientSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    private void broadcastMessage(String message) {
        for (Socket socket : clientSockets) {
            if (socket != clientSocket) {
                try {
                    PrintWriter socketWriter = new PrintWriter(socket.getOutputStream(),
true);
                    socketWriter.println(message);

```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}
}
}
}
}
}
}
}

```

TCP Multi-client chat Client:

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class TCPClient3568 {
    private static final String SERVER_IP = "127.0.0.1"; // Change this to the server's IP
    address
    private static final int PORT = 8888;
    public static void main(String[] args) {
        System.out.println("NAME: Vijai Suria.M \nReg No:2021503568");
        System.out.println("Date:" + LocalDate.now());
        System.out.println("Time:" + LocalTime.now());
        System.out.println("\n");
        try (Socket socket = new Socket(SERVER_IP, PORT);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader serverReader = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {
            System.out.println("Connected to the chat server.");
            Thread serverListener = new Thread(() -> {
                String serverMessage;
                try {
                    while ((serverMessage = serverReader.readLine()) != null) {
                        System.out.println(serverMessage);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            });
            serverListener.start();

```



```

        String name = serverReader.readLine();
        System.out.println(name);
        while (true) {
            String message = reader.readLine();
            writer.println(message);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

OUTPUT (Multi-client chat Server):

TCP Server:

```

Run  TCPServer3568 x TCPClient3568 x TCPClient3568 x
"\"C:\Program Files\Java\jdk-19\bin\java.exe\" \"-javaagent:C:\Program Files\
Date:2023-11-22
Time:18:14:22.266814100
NAME: Vijai Suria .M
Reg No:2021503568

Server started. Waiting for clients...
New client connected: Socket[addr=/127.0.0.1,port=61839,localport=8888]
New client connected: Socket[addr=/127.0.0.1,port=61843,localport=8888]

```

CLIENT-1

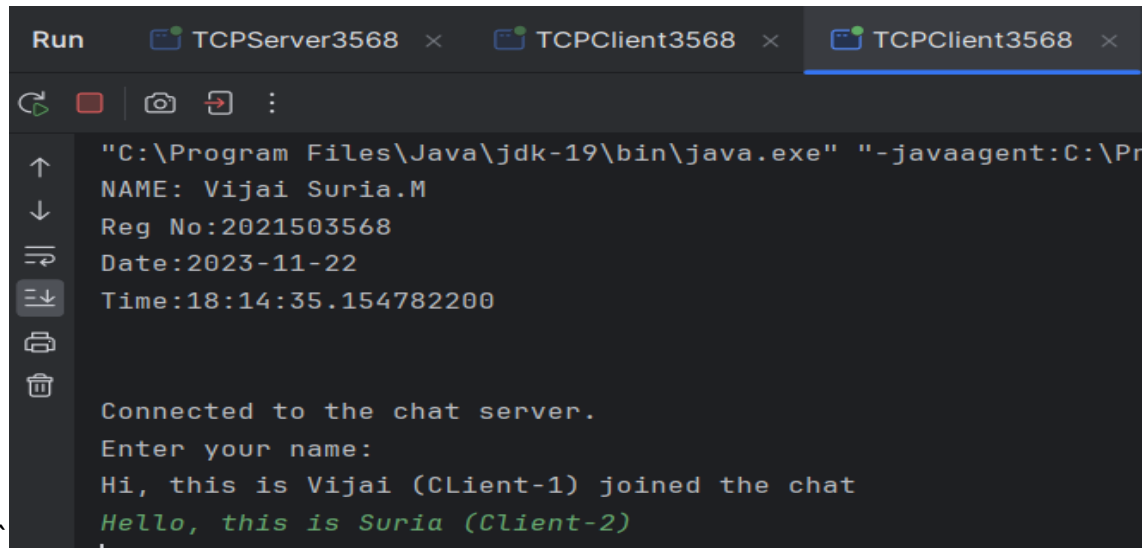
```

Run  TCPServer3568 x TCPClient3568 x TCPClient3568 x
"\"C:\Program Files\Java\jdk-19\bin\java.exe\" \"-javaagent:C:\Pr
NAME: Vijai Suria.M
Reg No:2021503568
Date:2023-11-22
Time:18:14:30.912583700

Connected to the chat server.
Enter your name:
Hi, this is Vijai (Client-1)
Hello, this is Suria (Client-2) joined the chat

```

CLIENT-2



```
Run TCPServer3568 x TCPClient3568 x TCPClient3568 x
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Pr
NAME: Vijai Suria.M
Reg No:2021503568
Date:2023-11-22
Time:18:14:35.154782200

Connected to the chat server.
Enter your name:
Hi, this is Vijai (Client-1) joined the chat
Hello, this is Suria (Client-2)
```

UDP Multi-client chat Server:

```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class UDPServer3568 {
    private static final int PORT = 8888;
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try (DatagramSocket serverSocket = new DatagramSocket(PORT)) {
            System.out.println("Server started. Waiting for clients...");
            // Receiving messages from clients
            while (true) {
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                serverSocket.receive(receivePacket);

                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();
                String receivedMessage = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                System.out.println("Client [" + clientAddress + ":" + clientPort + "]: " +
receivedMessage);
            }
        }
    }
}
```

```

        // Sending received message back to the sender client
        sendData(serverSocket, clientAddress, clientPort,
receivedMessage.getBytes());
    }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private static void sendData(DatagramSocket socket, InetAddress address, int port,
byte[] data) throws IOException {
    DatagramPacket sendPacket = new DatagramPacket(data, data.length, address,
port);
    socket.send(sendPacket);
}
}

```

UDP Multi-client chat Client:

```

import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class UDPClient3568 {
    private static final String SERVER_IP = "127.0.0.1"; // Change this to the server's IP
address
    private static final int PORT = 8888;
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        try (DatagramSocket clientSocket = new DatagramSocket();
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in))) {
            InetAddress serverAddress = InetAddress.getByName(SERVER_IP);
            Thread serverListener = new Thread(() -> {
                try {
                    while (true) {
                        byte[] receiveData = new byte[1024];
                        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
                        clientSocket.receive(receivePacket);

                        String receivedMessage = new String(receivePacket.getData(), 0,

```

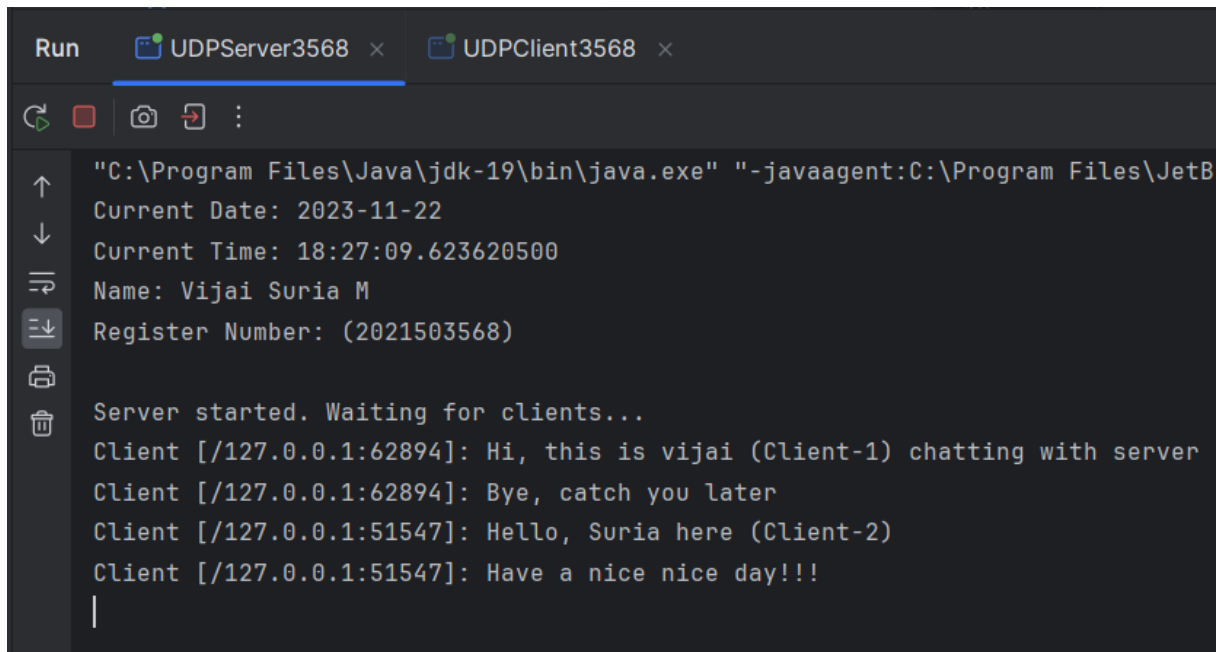
```

receivePacket.getLength());
        System.out.println(receivedMessage);
    }
} catch (IOException e) {
    e.printStackTrace();
}
});
serverListener.start();
System.out.println("Connected to the chat server.");
while (true) {
    String message = reader.readLine();
    byte[] sendData = message.getBytes();
    DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, serverAddress, PORT);
    clientSocket.send(sendPacket);
}
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

OUTPUT (Multi-client chat)

UDP Multi-chat Server:



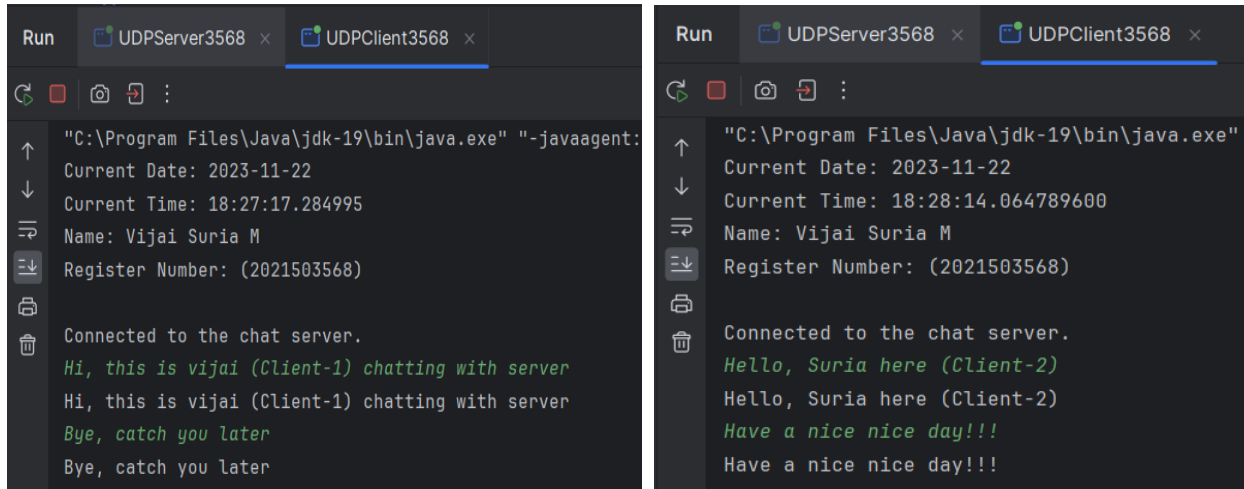
```

Run  UDPServer3568 x  UDPClient3568 x
"\"C:\\Program Files\\Java\\jdk-19\\bin\\java.exe\" \"-javaagent:C:\\Program Files\\JetB
Current Date: 2023-11-22
Current Time: 18:27:09.623620500
Name: Vijai Suria M
Register Number: (2021503568)

Server started. Waiting for clients...
Client [/127.0.0.1:62894]: Hi, this is vijai (Client-1) chatting with server
Client [/127.0.0.1:62894]: Bye, catch you later
Client [/127.0.0.1:51547]: Hello, Suria here (Client-2)
Client [/127.0.0.1:51547]: Have a nice nice day!!!
|

```

UDP Multi-chat Client-1:



```
Run  UDPServer3568 x  UDPCClient3568 x
↑ "C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:
↓ Current Date: 2023-11-22
↵ Current Time: 18:27:17.284995
⇓ Name: Vijai Suria M
⇓ Register Number: (2021503568)
⇓
⇓ Connected to the chat server.
⇓ Hi, this is vijai (Client-1) chatting with server
⇓ Hi, this is vijai (Client-1) chatting with server
⇓ Bye, catch you later
⇓ Bye, catch you later

Run  UDPServer3568 x  UDPCClient3568 x
↑ "C:\Program Files\Java\jdk-19\bin\java.exe"
↓ Current Date: 2023-11-22
↵ Current Time: 18:28:14.064789600
⇓ Name: Vijai Suria M
⇓ Register Number: (2021503568)
⇓
⇓ Connected to the chat server.
⇓ Hello, Suria here (Client-2)
⇓ Hello, Suria here (Client-2)
⇓ Have a nice nice day!!!
⇓ Have a nice nice day!!!
```

7.4) Write a java code for movie ticket booking system using multi-client socket programming. Implement the synchronization mechanism of ticket booking operations at server side to handle multiple clients concurrently.

SERVER CODE (for movie ticket booking):

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class MovieServer_3568 {
    private static final int PORT = 8888;
    private static final int MAX_AVAILABLE_TICKETS = 100;
    private static int availableTickets = MAX_AVAILABLE_TICKETS;
    private static final Object lock = new Object();

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalDateTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        System.out.println("\n");
        ExecutorService executor = Executors.newFixedThreadPool(10);
```

```

try (ServerSocket serverSocket = new ServerSocket(PORT)) {
    System.out.println("Server started. Waiting for clients...");
    while (true) {
        Socket clientSocket = serverSocket.accept();
        System.out.println("New client connected: " + clientSocket);
        Runnable clientHandler = new ClientHandler(clientSocket);
        executor.execute(clientHandler);
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    executor.shutdown();
}
}

static class ClientHandler implements Runnable {
    private final Socket clientSocket;
    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
    }
    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);

            String clientMessage;
            while ((clientMessage = reader.readLine()) != null) {
                if (clientMessage.equalsIgnoreCase("bookTicket")) {
                    bookTicket(writer);
                } else {
                    writer.println("Invalid command.");
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void bookTicket(PrintWriter writer) {
        synchronized (lock) {
            if (availableTickets > 0) {
                availableTickets--;
                writer.println("Ticket booked successfully. Remaining tickets: " +

```

```

availableTickets);
    } else {
        writer.println("Sorry, tickets are sold out.");
    }
}
}
}
}
}
}
}
}

```

CLIENT CODE (for movie ticket booking):

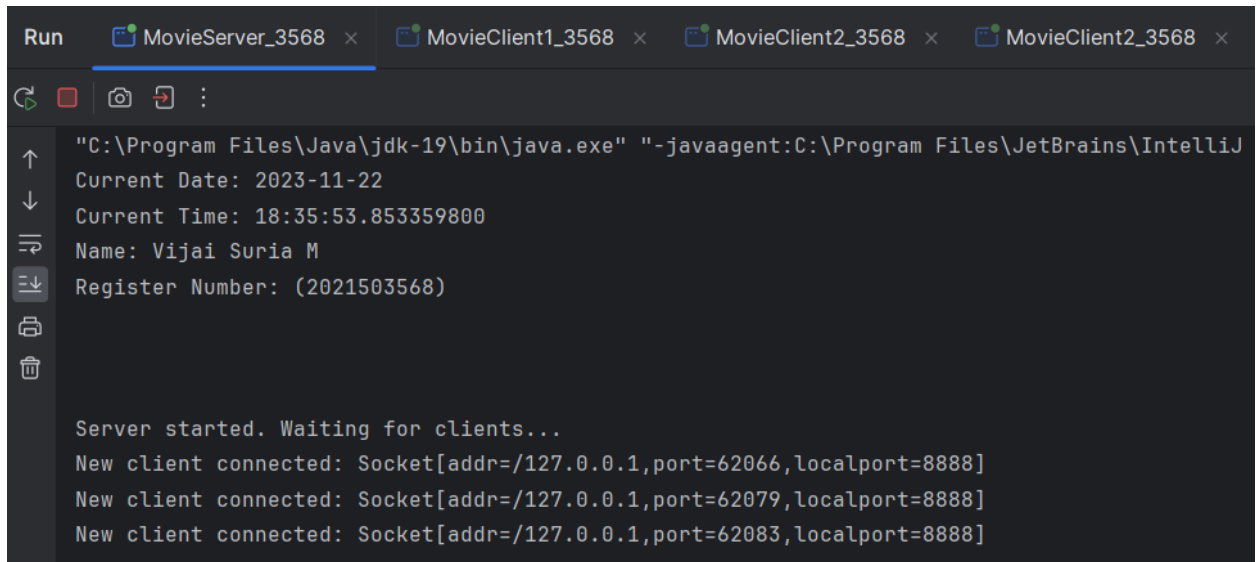
```

import java.io.*;
import java.net.*;
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.*;
public class MovieClient1_3568 {
    private static final String SERVER_IP = "127.0.0.1"; // Change this to the server's IP
    private static final int PORT = 8888;
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568) \n");
        System.out.println("\n");
        try (Socket socket = new Socket(SERVER_IP, PORT);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader serverReader = new BufferedReader(new
InputStreamReader(socket.getInputStream())) {
            System.out.println("Connected to the Movie Ticket Booking Server.");
            while (true) {
                System.out.println("Enter 'bookTicket' to book a ticket or 'exit' to quit:");
                String userInput = reader.readLine();
                if (userInput.equalsIgnoreCase("exit")) {
                    break;
                }
                writer.println(userInput);
                String serverResponse = serverReader.readLine();
                System.out.println("Server Response: " + serverResponse);
            }
        } catch (IOException e) {

```

```
e.printStackTrace();  
    }  
}  
}
```

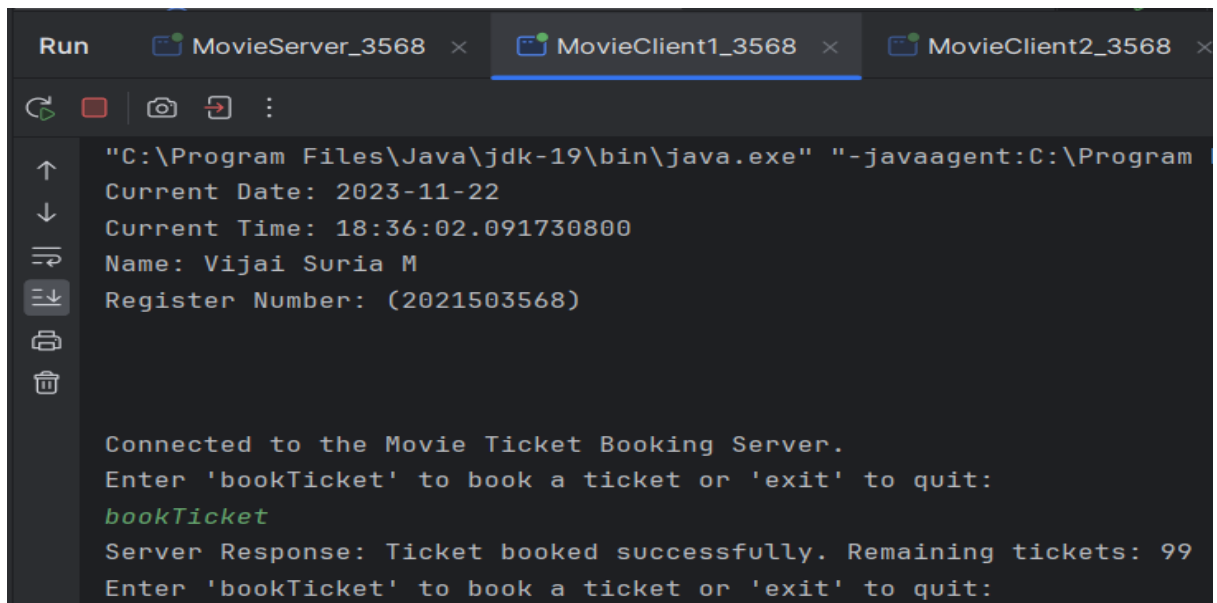
Server (for movie ticket booking):



The screenshot shows the Run console for the MovieServer_3568 application. The console output displays the Java command used to run the application, followed by the current date and time, the user's name (Vijai Suria M), and the register number (2021503568). The server then starts and waits for clients. Three new clients are connected, each with their respective IP addresses and ports.

```
Run  MovieServer_3568 x  MovieClient1_3568 x  MovieClient2_3568 x  MovieClient2_3568 x  
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ  
Current Date: 2023-11-22  
Current Time: 18:35:53.853359800  
Name: Vijai Suria M  
Register Number: (2021503568)  
  
Server started. Waiting for clients...  
New client connected: Socket[addr=/127.0.0.1,port=62066,localport=8888]  
New client connected: Socket[addr=/127.0.0.1,port=62079,localport=8888]  
New client connected: Socket[addr=/127.0.0.1,port=62083,localport=8888]
```

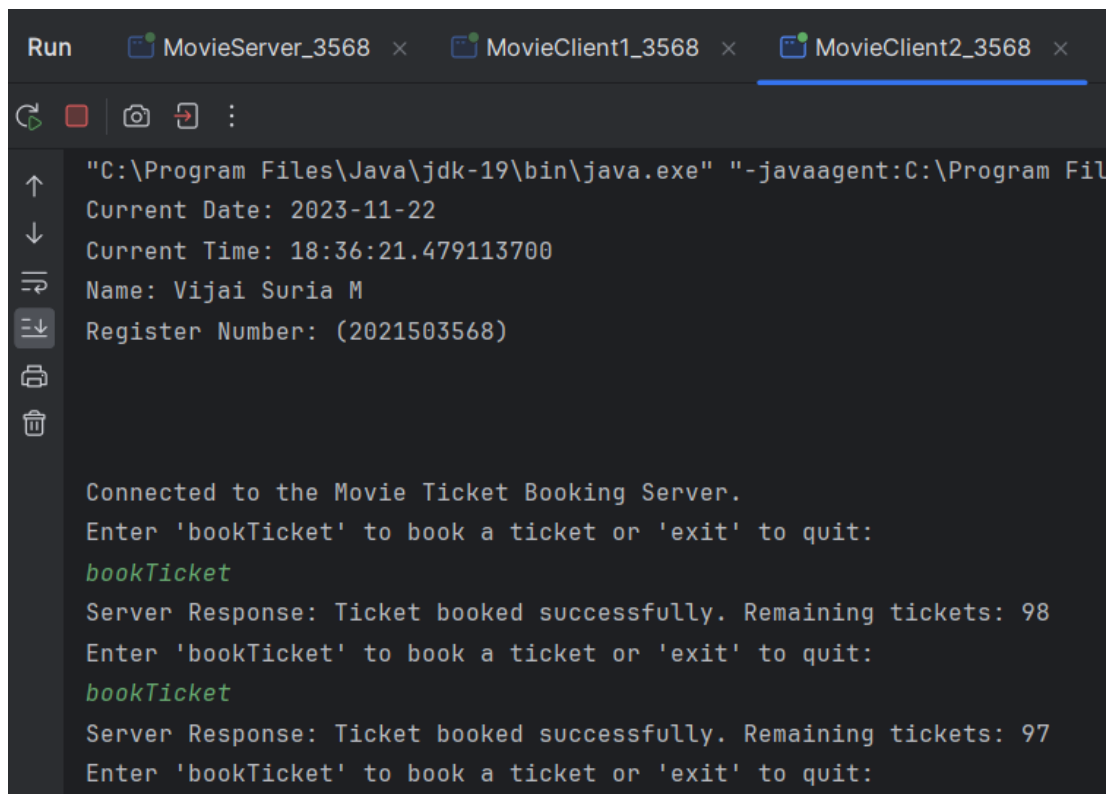
Clients (for movie ticket booking):-



The screenshot shows the Run console for the MovieClient1_3568 application. The console output displays the Java command used to run the application, followed by the current date and time, the user's name (Vijai Suria M), and the register number (2021503568). The client then connects to the Movie Ticket Booking Server. The user enters 'bookTicket' to book a ticket, and the server responds with 'Ticket booked successfully. Remaining tickets: 99'.

```
Run  MovieServer_3568 x  MovieClient1_3568 x  MovieClient2_3568 x  
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program  
Current Date: 2023-11-22  
Current Time: 18:36:02.091730800  
Name: Vijai Suria M  
Register Number: (2021503568)  
  
Connected to the Movie Ticket Booking Server.  
Enter 'bookTicket' to book a ticket or 'exit' to quit:  
bookTicket  
Server Response: Ticket booked successfully. Remaining tickets: 99  
Enter 'bookTicket' to book a ticket or 'exit' to quit:
```

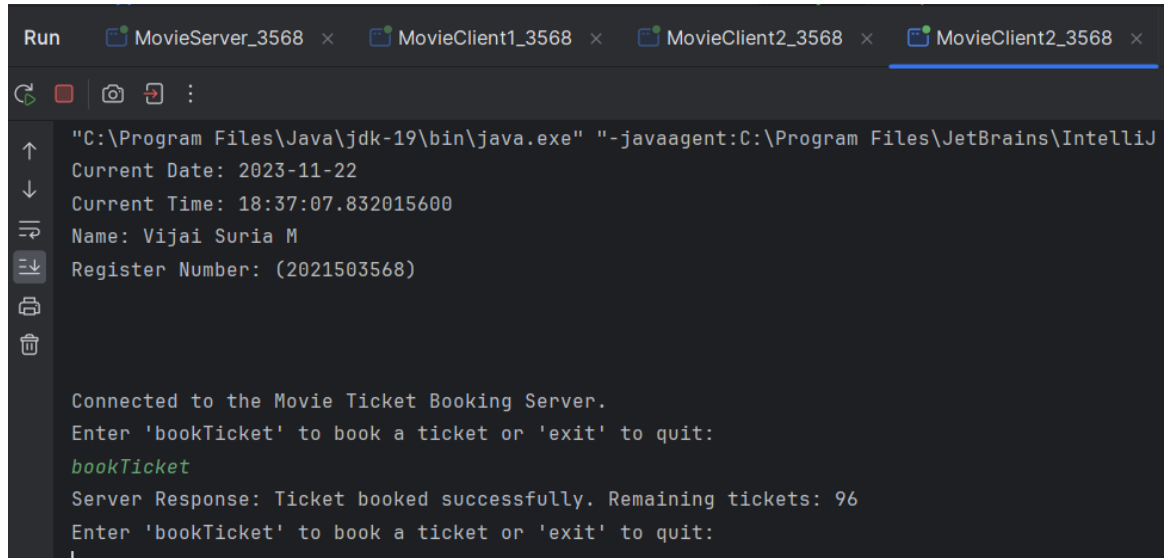

Client-2 (for movie ticket booking):-



```
Run  MovieServer_3568 x  MovieClient1_3568 x  MovieClient2_3568 x
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Fil
Current Date: 2023-11-22
Current Time: 18:36:21.479113700
Name: Vijai Suria M
Register Number: (2021503568)

Connected to the Movie Ticket Booking Server.
Enter 'bookTicket' to book a ticket or 'exit' to quit:
bookTicket
Server Response: Ticket booked successfully. Remaining tickets: 98
Enter 'bookTicket' to book a ticket or 'exit' to quit:
bookTicket
Server Response: Ticket booked successfully. Remaining tickets: 97
Enter 'bookTicket' to book a ticket or 'exit' to quit:
```

Client-3 (for movie ticket booking):-



```
Run  MovieServer_3568 x  MovieClient1_3568 x  MovieClient2_3568 x  MovieClient2_3568 x
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Current Date: 2023-11-22
Current Time: 18:37:07.832015600
Name: Vijai Suria M
Register Number: (2021503568)

Connected to the Movie Ticket Booking Server.
Enter 'bookTicket' to book a ticket or 'exit' to quit:
bookTicket
Server Response: Ticket booked successfully. Remaining tickets: 96
Enter 'bookTicket' to book a ticket or 'exit' to quit:
|
```

RESULT:

Thus, the concept of multithreading and deadlocks in java programming were implemented and executed successfully.

JAVA DATABASE CONNECTION

Aim: To explore about java database connection

Algorithm:

Step 1: To store student details in DB and calculate GPA

1. `in.next()`: Reads the next input token as a String.
2. `in.nextInt()`: Reads the next input token as an int.
3. `LocalDate.now()`: Obtains the current date.
4. `LocalTime.now()`: Obtains the current time.
5. `DriverManager.getConnection(String url, String user, String password)`: Establishes a connection to a database using the provided URL, username, and password.
6. `Connection.createStatement()`: Creates a Statement object for sending SQL statements to the database.
7. `Statement.executeUpdate(String sql)`: Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement, returning the count of affected rows.
8. `Statement.executeQuery(String sql)`: Executes the given SQL SELECT statement, returning a ResultSet object containing the resulting data.
9. `ResultSet.getString(String columnLabel)`: Retrieves the value of the designated column in the current row of the ResultSet as a String.
10. `ResultSet.getInt(String columnLabel)`: Retrieves the value of the designated column in the current row of the ResultSet as an int.
11. `ResultSet.getDouble(String columnLabel)`: Retrieves the value of the designated column in the current row of the ResultSet as a double.
12. `BigDecimal(String val)`: Constructs a BigDecimal object from a String representation.
13. `BigDecimal.setScale(int scale, RoundingMode roundingMode)`: Sets the scale of this BigDecimal to the specified number of digits after the decimal point using the specified rounding mode.
14. `getCreditPoints(int mark)`:
 - Calculates credit points based on the provided marks.
 - Returns an integer value representing credit points.
15. `calculateGPA(int[] creditPoints)`
 - Calculates the GPA based on an array of credit points for each course..

Step 2: Movie ticket booking using multithreading, synchronization and JDBC

1. run(): Within ClientHandler class, handles client requests by processing incoming messages.
2. bookTicket(PrintWriter writer): Attempts to book a ticket for the client and updates the available ticket count.
3. insertBookingIntoDB(): Inserts booking details into a MySQL database: client ID, ticket count, and booking time.
4. println(String x): Prints a string to the output stream, sending it to the client.
5. accept() (From ServerSocket): Listens and accepts client connections.
6. getInputStream(): Retrieves the input stream of the socket to read data.
7. getOutputStream(): Retrieves the output stream of the socket to send data.
8. readLine() (From BufferedReader): Reads a line of text from the input stream.
9. executeUpdate(): Executes an SQL INSERT, UPDATE, or DELETE statement.
10. now(): Retrieves the current date and time.
11. ServerSocket, Socket, BufferedReader, PrintWriter

- Handles server-client communication over sockets, reading input, and writing output.

12. ExecutorService, Executors

- Manages a thread pool for handling multiple client connections concurrently.

13. Connection, DriverManager, PreparedStatement, SQLException

- Manages database connections, executes SQL queries, and handles SQL-related exceptions.

12.1) Write a java JDBC code for storing student details such as name, regno, gender, current semester course mark and GPA(using CalculateGPA method):

CODE:

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.Statement;
```

```

import java.util.Scanner;
import java.time.LocalDate;
import java.time.LocalTime;
import java.math.BigDecimal;
import java.math.RoundingMode;

public class connectionJDBC {
    public static int getCreditPoints(int mark) {
        if (mark >= 91 && mark <= 100) {
            return 10;
        } else if (mark >= 81 && mark <= 90) {
            return 9;
        } else if (mark >= 71 && mark <= 80) {
            return 8;
        } else if (mark >= 61 && mark <= 70) {
            return 7;
        } else if (mark >= 51 && mark <= 60) {
            return 6;
        } else {
            return 0;
        }
    }

    public static float calculateGPA(int[] creditPoints){
        int[] credits = {4,4,4,5,6};
        float res=0.0F;
        int total=0;
        for(int i=0;i<credits.length;i++) {
            res += (credits[i]*creditPoints[i]);
            total+=credits[i];
        }
        res = res/total;
        return res;
    }

    public static void main(String[] args) throws Exception {
        Scanner in = new Scanner(System.in);
        Connection con = null;
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
    }
}

```

```

\n");
    try {
        con =
DriverManager.getConnection("jdbc:oracle:thin:@192.168.109.28:1521:orcl",
"ct2021503568", "ct2021503568");
        if (con == null){
            System.out.println("Database not Connected");
            System.exit(0);
        }
        else{
            System.out.println("Connected to Database 2021503568");
        }
    }
    catch (Exception e) {
        System.out.println(e);
    }

    try{
        Statement stmt = con.createStatement();
        boolean flag = true;

        while(flag){
            System.out.println("Menu Board");
            System.out.println("1 --> Insert record");
            System.out.println("2 --> View records");
            System.out.println("3 --> Exit");
            System.out.print("Enter your choice: ");
            int ch = in.nextInt();

            switch (ch){
                case 1:
                    System.out.println("Enter the Student name");
                    String name = in.next();
                    System.out.println("Enter the Student reg no.: ");
                    String regno = in.next();
                    System.out.println("Enter the Student Gender: ");
                    String gender = in.next();
                    System.out.println("Enter the course-1 marks: ");
                    int c1 = in.nextInt();
                    System.out.println("Enter the course-2 marks: ");
                    int c2 = in.nextInt();

```

```

        System.out.println("Enter the course-3 marks: ");
        int c3 = in.nextInt();
        System.out.println("Enter the course-4 marks: ");
        int c4 = in.nextInt();
        System.out.println("Enter the course-5 marks: ");
        int c5 = in.nextInt();

        int[] creditPoints = new int[5]; // Assuming 5 courses
        creditPoints[0] = getCreditPoints(c1);
        creditPoints[1] = getCreditPoints(c2);
        creditPoints[2] = getCreditPoints(c3);
        creditPoints[3] = getCreditPoints(c4);
        creditPoints[4] = getCreditPoints(c5);

        float gpa = calculateGPA(creditPoints);
        BigDecimal gpaBigDecimal = new
        BigDecimal(Float.toString(gpa));
        gpaBigDecimal = gpaBigDecimal.setScale(2,
        RoundingMode.HALF_UP);
        int count = stmt.executeUpdate("Insert into student(name, regno,
        gender, c1, c2, c3, c4, c5, gpa) VALUES (" + name + "," + regno + "," + gender
        + "," + c1 + "," + c2 + "," + c3 + "," + c4 + "," + c5 + "," + gpaBigDecimal + ")");
        System.out.println("No. of rows inserted: " + count);
        break;

    case 2:
        ResultSet rs = stmt.executeQuery("Select * from student");
        System.out.println("Student Table: ");
        System.out.println("-----");
        -----");
        while(rs.next()){
            System.out.println(
                rs.getString("name") + "\t\t" + rs.getString("regno") +
                "\t\t" + rs.getString("gender") + "\t\t" + rs.getInt("c1") + "\t\t" + rs.getInt("c2") +
                "\t\t" + rs.getInt("c3") + "\t\t" + rs.getInt("c4") + "\t\t" + rs.getInt("c5") + "\t\t" +
                rs.getDouble("gpa"));
        }
        System.out.println("-----");
        -----");
        break;

```

```

        case 3:
            System.out.println("Thank you.....");
            flag=false;
            break;

        default:
            System.out.println("Enter the valid choice....");
    }
}

}
catch (Exception e) {
    e.printStackTrace();
}

}

}

```

OUTPUT:

```

Menu Board
1 --> Insert record
2 --> View records
3 --> Exit
Enter your choice: 2
Student Table:
-----
Vijai | 68 | M | 98 | 78 | 88 | 95 | 94 | 9.48
Suria | 68 | M | 98 | 97 | 85 | 92 | 89 | 9.57
-----
Menu Board
1 --> Insert record
2 --> View records
3 --> Exit
Enter your choice: 3
Thank you.....

```



```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-22
Current Time: 16:16:33.756881600
Name: Vijai Suria M
Register Number: (2021503568)

Connected to Database 2021503568
Menu Board
1 --> Insert record
2 --> View records
3 --> Exit
Enter your choice: 1
Enter the Student name
Suria
Enter the Student reg no.:
68
Enter the Student Gender:
M
Enter the course-1 marks:
98
Enter the course-2 marks:
97
Enter the course-3 marks:
85
Enter the course-4 marks:
92
Enter the course-5 marks:
89
No. of rows inserted: 1
```

12.2) Write a java code for movie ticket booking system using multi-client socket programming. Implement the synchronization mechanism of ticket booking operations at server side to handle multiple clients concurrently.

CODE:

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalTime;
```

```

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class MovieServer_JDBC3568 {
    private static final String JDBC_URL = "jdbc:mysql://localhost:3308/vijai-java";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASSWORD = "";
    private static final int PORT = 8888;
    private static final int MAX_AVAILABLE_TICKETS = 100;
    private static int availableTickets = MAX_AVAILABLE_TICKETS;
    private static final Object lock = new Object();

    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)
\n");
        System.out.println("\n");
        ExecutorService executor = Executors.newFixedThreadPool(10);
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server started. Waiting for clients...");
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("New client connected: " + clientSocket);

                Runnable clientHandler = new ClientHandler(clientSocket);
                executor.execute(clientHandler);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            executor.shutdown();
        }
    }

    static class ClientHandler implements Runnable {
        private final Socket clientSocket;

```

```

public ClientHandler(Socket socket) {
    this.clientSocket = socket;
}
@Override
public void run() {
    try {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(),
true);

        String clientMessage;
        while ((clientMessage = reader.readLine()) != null) {
            if (clientMessage.equalsIgnoreCase("bookTicket")) {
                bookTicket(writer);
            } else {
                writer.println("Invalid command.");
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void insertBookingIntoDB() {
    try (Connection connection =
DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD))
    {
        int clientPort = clientSocket.getPort();
        String insertQuery = "INSERT INTO bookings (client_id, ticket_count,
booking_time) VALUES (?, ?, NOW())";
        PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery);
        preparedStatement.setInt(1, clientPort); // Replace with the actual
client ID
        preparedStatement.setInt(2, 1); // Assuming booking one ticket per
int rowsAffected = preparedStatement.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Booking inserted into the database.");
        } else {
            System.out.println("Failed to insert booking.");
        }
    }
}

```

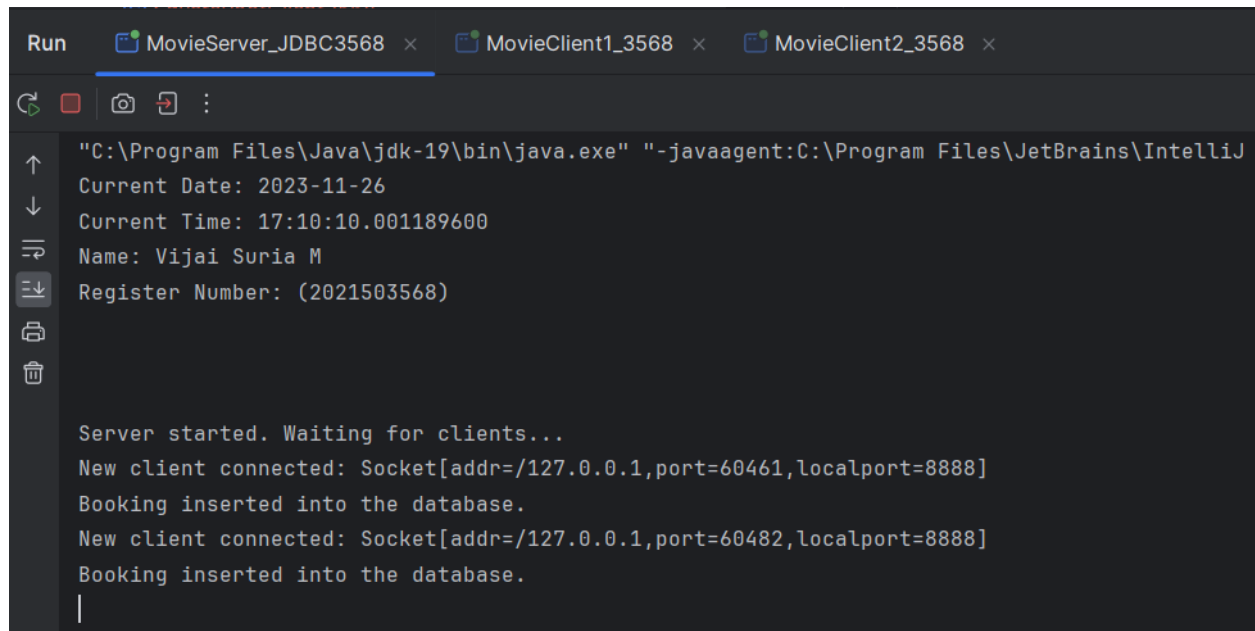
```

    }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
private void bookTicket(PrintWriter writer) {
    synchronized (lock) {
        if (availableTickets > 0) {
            availableTickets--;
            writer.println("Ticket booked successfully. Remaining tickets: " +
availableTickets);
            insertBookingIntoDB();
        } else {
            writer.println("Sorry, tickets are sold out.");
        }
    }
}
}
}
}
}

```

OUTPUT:

Movie Database Server: -



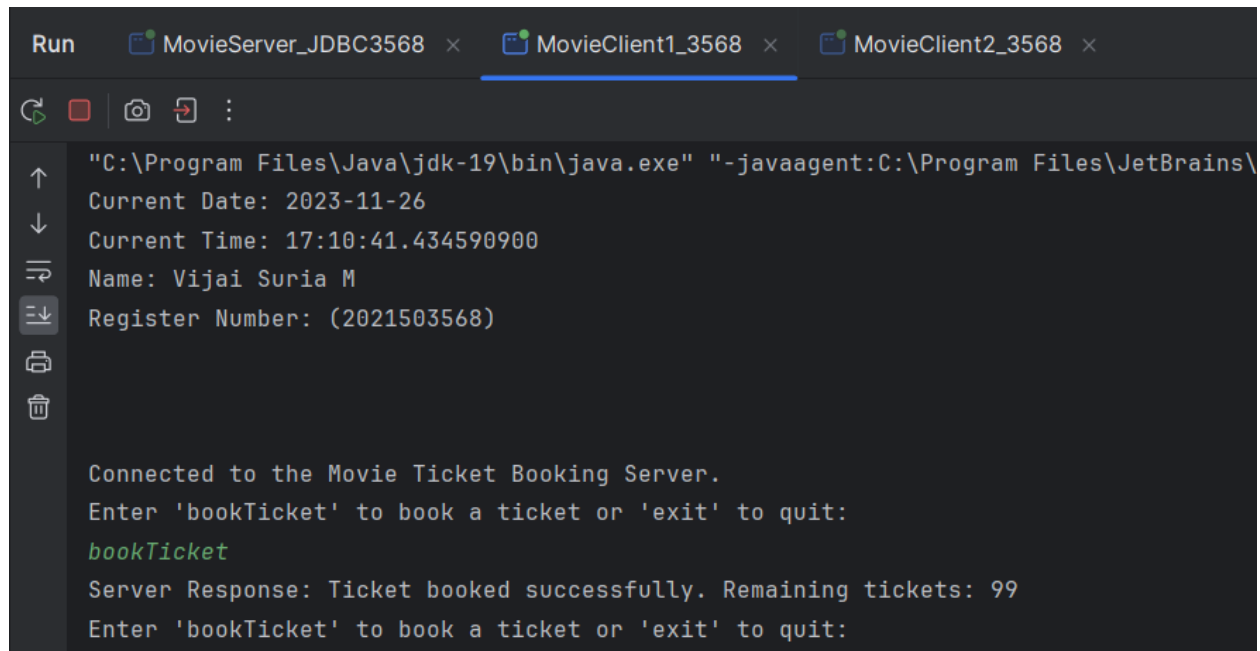
```

Run  MovieServer_JDBC3568 x MovieClient1_3568 x MovieClient2_3568 x
C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Current Date: 2023-11-26
Current Time: 17:10:10.001189600
Name: Vijai Suria M
Register Number: (2021503568)

Server started. Waiting for clients...
New client connected: Socket[addr=/127.0.0.1,port=60461,localport=8888]
Booking inserted into the database.
New client connected: Socket[addr=/127.0.0.1,port=60482,localport=8888]
Booking inserted into the database.
|

```

Client-1: -



```
Run  MovieServer_JDBC3568 x  MovieClient1_3568 x  MovieClient2_3568 x
C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\
Current Date: 2023-11-26
Current Time: 17:10:41.434590900
Name: Vijai Suria M
Register Number: (2021503568)

Connected to the Movie Ticket Booking Server.
Enter 'bookTicket' to book a ticket or 'exit' to quit:
bookTicket
Server Response: Ticket booked successfully. Remaining tickets: 99
Enter 'bookTicket' to book a ticket or 'exit' to quit:
```

Database Server: (Booking table)

booking_id	client_id	ticket_count	booking_time
1	123	1	2023-11-26 17:06:31
2	60461	1	2023-11-26 17:10:49
3	60482	1	2023-11-26 17:11:21

RESULT:

Thus, the concept of JDBC in java programming is implemented and executed successfully.

COLLECTIONS

Aim: To explore about collections and generics in java programming

Algorithm:

Step 1: Use ArrayList to manage products

16. LocalDate and LocalTime - Classes from the java.time package used to retrieve the current date and time, respectively.
17. ArrayList<Product> - Utilizes the ArrayList class from java.util to store Product objects.
18. Scanner - A class from java.util used for obtaining user input from the console.
19. Product(int id, String name, double price) - A constructor method within the Product class used to create Product objects with provided attributes (id, name, price).
20. pro_Obj.add(new Product(id, name, price)) - Adds a new Product object to the ArrayList pro_Obj.
21. pro_Obj.size() - Returns the size of the ArrayList pro_Obj.
22. pro_Obj.get(i) - Retrieves the element at the specified index i from the ArrayList pro_Obj.
23. pro_Obj.remove(i) - Removes the element at the specified index i from the ArrayList pro_Obj.
24. System.out.println() - Prints messages to the console.
25. System.exit(0) - Terminates the program with a status code of 0.

Step 2: Autoboxing and unboxing.

1. autobox(int value) - A method that performs autoboxing for an int value by converting it into an Integer object and returning the boxed value.
2. autobox(double value) - A method that performs autoboxing for a double value by converting it into a Double object and returning the boxed value.
3. autobox(char value) - A method that performs autoboxing for a char value by converting it into a Character object and returning the boxed value.
4. autobox(boolean value) - A method that performs autoboxing for a boolean value by converting it into a Boolean object and returning the boxed value.
5. unbox(Integer value) - A method that performs unboxing for an Integer object by extracting the int value and returning it.
6. unbox(Double value) - A method that performs unboxing for a Double object by extracting the double value and returning it.
7. unbox(Character value) - A method that performs unboxing for a Character object by extracting the char value and returning it.
8. unbox(Boolean value) - A method that performs unboxing for a Boolean object by extracting the boolean value and returning it.

Step 3: Manage an Amazon store to handling customer orders

1. addProd(Product p) - A method to add products to the inventory. It checks if the product already exists and adds it to the inventory map or throws a custom exception if the product already exists.
2. addToCart(String name, int quantity) - A method to add products to the shopping cart. It checks if the product exists, validates the requested quantity against available quantity, and adds the product to the cart or throws exceptions for product not found or insufficient quantity.
3. UserDefinedException(String message) - A custom exception class used to handle user-defined exceptions with specific error messages.
4. Product(String name, Double price, int q) - A constructor for the Product class to initialize product attributes (name, price, quantity).
5. getName() - A method in the Product class to retrieve the name of the product.
6. getPrice() - A method in the Product class to retrieve the price of the product.
7. getQuantity() - A method in the Product class to retrieve the quantity of the product.

13.1) Write a Java program to implement a Product class with id, name , and price attributes. Use an ArrayList to store Product objects. Prompt the user to add products (id, name, price) and provide a menu to list all products, search by ID, remove a product, and update its price.

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Product {
    int id;
    String name;
    Double price;

    Product(int id, String name, Double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Price: " + price;
    }
}
```



```

    }
}

public class ProductList3568 {
    public static void main(String args[]) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");

        List<Product> products = new ArrayList<Product>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add a product");
            System.out.println("2. List all products");
            System.out.println("3. Search for a product by ID");
            System.out.println("4. Remove a product");
            System.out.println("5. Update the price of a product");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    // Add a product
                    System.out.print("Enter product ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter product name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter product price: ");
                    double price = scanner.nextDouble();
                    Product product = new Product(id, name, price);
                    products.add(product);
                    break;

                case 2:
                    // List all products
                    for (Product p : products) {
                        System.out.println(p);
                    }
                    break;
            }
        }
    }
}

```

```

        case 3:
            // Search for a product by ID
            System.out.print("Enter product ID to search: ");
            int searchId = scanner.nextInt();
            for (Product p : products) {
                if (p.id == searchId) {
                    System.out.println("Found: " + p);
                    break;
                }
            }
            break;

        case 4:
            // Remove a product by ID
            System.out.print("Enter product ID to remove: ");
            int removeId = scanner.nextInt();
            products.removeIf(p -> p.id == removeId);
            break;

        case 5:
            // Update the price of a product by ID
            System.out.print("Enter product ID to update price: ");
            int updateId = scanner.nextInt();
            System.out.print("Enter the new price: ");
            double newPrice = scanner.nextDouble();
            for (Product p : products) {
                if (p.id == updateId) {
                    p.price = newPrice;
                    System.out.println("Price updated.");
                    break;
                }
            }
            break;

        case 6:
            // Exit the program
            scanner.close();
            System.exit(0);
            default:
                System.out.println("Invalid choice. Please try again.");
                break;
    }

```

```
}  
}  
}
```

OUTPUT:

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"  
Current Date: 2023-10-18  
Current Time: 14:54:23.657576800  
Name: Vijai Suria M  
Register Number: (2021503568)
```

Menu:

1. Add a product
2. List all products
3. Search for a product by ID
4. Remove a product
5. Update the price of a product
6. Exit

Menu:

1. Add a product
2. List all products
3. Search for a product by ID
4. Remove a product
5. Update the price of a product
6. Exit

Enter your choice: 2

ID: 101, Name: Vivo, Price: 17899.0

ID: 108, Name: Apple, Price: 350000.0

10.9) Write a Java program to perform Autoboxing and unboxing for all datatypes in java.

CODE:

```
import java.time.LocalDate;  
import java.time.LocalDateTime;  
  
public class Wrapper3568 {
```

```

public static void main(String[] args) {
    System.out.println("Current Date: " + LocalDate.now());
    System.out.println("Current Time: " + LocalTime.now());
    System.out.println("Name: Vijai Suria M \nRegister Number: (2021503568)");
    int a = 10;
    Integer aWrapper = a; // Autoboxing (int to Integer)
    int aUnboxed = aWrapper; // Unboxing (Integer to int)
    boolean b = true;
    Boolean bWrapper = b; // Autoboxing (boolean to Boolean)
    boolean bUnboxed = bWrapper; // Unboxing (Boolean to boolean)
    char c = 'a';
    Character cWrapper = c;
    char cUnboxed = cWrapper;
    System.out.println("Autoboxing and Unboxing Examples:");
    System.out.println("a: " + a + ", aWrapper: " + aWrapper + ", aUnboxed: " +
aUnboxed);
    System.out.println("d: " + d + ", dWrapper: " + dWrapper + ", dUnboxed: " +
dUnboxed);
    System.out.println("f: " + f + ", fWrapper: " + fWrapper + ", fUnboxed: " +
fUnboxed);
    System.out.println("b: " + b + ", bWrapper: " + bWrapper + ", bUnboxed: " +
bUnboxed);
    System.out.println("c: " + c + ", cWrapper: " + cWrapper + ", cUnboxed: " +
cUnboxed);
    }
}

```

OUTPUT:

```

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-10-18
Current Time: 15:08:22.859430300
Name: Vijai Suria M
Register Number: (2021503568)
Autoboxing and Unboxing Examples:
a: 10, aWrapper: 10, aUnboxed: 10
d: 97.09, dWrapper: 97.09, dUnboxed: 97.09
f: 9.83, fWrapper: 9.83, fUnboxed: 9.83
b: true, bWrapper: true, bUnboxed: true
c: a, cWrapper: a, cUnboxed: a

```

13.3) Implement a Java program featuring Product, AmazonStore, and Main classes to manage an Amazon store, handling customer orders with dedicated functionalities and serving as the program's entry point.

CODE:

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.*;
class ProductAlreadyExistsException extends Exception {
    public ProductAlreadyExistsException(String message) {
        super(message);
    }
}
class ProductNotFoundException extends Exception {
    public ProductNotFoundException(String message) {
        super(message);
    }
}
class InsufficientQuantityException extends Exception {
    public InsufficientQuantityException(String message) {
        super(message);
    }
}
class Product {
    public String name;
    public double price;
    public int quantity;
    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public int getQuantity() {
        return quantity;
    }
}
class AmazonStore {
    private Map<String, Product>productMap;
    private Set<String>productNames;
    private List<Product>cart;
```

```

    public AmazonStore() {
        productMap = new HashMap<>();
        productNames = new HashSet<>();
        cart = new ArrayList<>();
    }

    public void addProduct(String name, double price, int quantity) throws
    ProductAlreadyExistsException {
        if (productNames.contains(name)) {
            throw new ProductAlreadyExistsException("Product with the same
            name already exists.");
        }
        Product product = new Product(name, price, quantity);
        productMap.put(name, product);
        productNames.add(name);
    }

    public void addToCart(String name, int quantity) throws
    ProductNotFoundException, InsufficientQuantityException {
        if (!productNames.contains(name)) {
            throw new ProductNotFoundException("Product not found.");
        }
        Product product = productMap.get(name);
        if (product.getQuantity() < quantity) {
            throw new InsufficientQuantityException("Insufficient quantity of " +
            name + " in stock.");
        }
        cart.add(new Product(name, product.getPrice(), quantity));
        product.quantity -= quantity;
    }

    public void listProducts() {
        System.out.println("Available Products:");
        for (Product product : productMap.values()) {
            System.out.println(
            product.getName() + " - Price: $" + product.getPrice() + " - Quantity: " +
            product.getQuantity());
        }
    }

    public void viewCart() {
        System.out.println("Shopping Cart:");
        double total = 0;
        for (Product product : cart) {
            System.out.println(
            product.getName() + " - Price: $" + product.getPrice() + " - Quantity: " +
            product.getQuantity());
            total += product.getPrice() * product.getQuantity();
        }
    }

```

```

        System.out.println("Total Price: $" + total);
    }
}

public class Main {
    public static void main(String[] args) {
        AmazonStore store = new AmazonStore();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        System.out.println("Name: Vijai Suria M \nRegister Number:
(2021503568)");

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add a product");
            System.out.println("2. List products");
            System.out.println("3. Add products to the cart");
            System.out.println("4. View cart");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    // Add a product
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter product name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter product price: $");
                    double price = scanner.nextDouble();
                    System.out.print("Enter product quantity: ");
                    int quantity = scanner.nextInt();
                    try {
                        store.addProduct(name, price, quantity);
                    } catch (ProductAlreadyExistsException e) {
                        System.out.println("Product already exists.");
                        break;
                    }
                    break;

                case 2:
                    // List available products
                    store.listProducts();
                    break;

                case 3:
                    // Add products to the cart

```

```

scanner.nextLine(); // Consume newline
System.out.print("Enter product name to add to cart: ");
    String cartName = scanner.nextLine();
System.out.print("Enter quantity: ");
    int cartQuantity = scanner.nextInt();
    try {
store.addToCart(cartName, cartQuantity);
    } catch (ProductNotFoundException e) {
System.out.println("Product not found.");
break;
    } catch (InsufficientQuantityException e) {
System.out.println("Insufficient quantity of " + cartName + " in stock.");
break;
    }
break;

    case 4:
        // View the shopping cart
store.viewCart();
break;

    case 5:
        // Exit the program
scanner.close();
System.exit(0);

    default:
System.out.println("Invalid choice. Please try again.");
break;
    }
}
}
}
}

```


OUTPUT:

```
Current Date: 2023-10-19  
Current Time: 11:54:29.964906300  
Name: Vijai Suria M  
Register Number: (2021503568)
```

```
Menu:  
1. Add a product  
2. List products  
3. Add products to the cart  
4. View cart  
5. Exit  
Enter your choice: 1  
Enter product name: Samsung  
Enter product price: $200  
Enter product quantity: 5
```

```
Menu:  
1. Add a product  
2. List products  
3. Add products to the cart  
4. View cart  
5. Exit  
Enter your choice: 2  
Available Products:  
Samsung - Price: $200.0 - Quantity: 5
```

RESULT:

Thus, the collections and generics in java in implemented and executed successfully.

SERVLETS AND JSP

Aim: To explore about servlets and JSP in java programming

Algorithm:

Step 1: Servlet and JSP code for calculator program

HTML Code:

<form>: Defines the form for user input.

<input>: Creates input fields for numbers and the submit button.

<select>: Creates a dropdown menu for selecting the operation.

<label>: Provides labels for input fields.

<h2>: Adds a heading for the calculator.

Servlet Code:

request.getParameter("parameterName"): This method is used to retrieve values from the form parameters.

Integer.parseInt(string): It's used to convert the string inputs obtained from the form to integers for mathematical operations.

response.getWriter(): This method retrieves the output stream of the response, allowing the servlet to write content that will be sent back to the client.

JSP Code:

<%= ... %>: JSP expression tag for outputting content to the client.

<% ... %>: JSP scriptlet tag for embedding Java code within the HTML.

request.getParameter("parameterName"): Retrieves values from the form parameters.

out.println("..."): Outputs content to the client.

Step 2: Servlet and JSP code for student detail Insertion

HTML CODE:

<form>: Used to create a form for collecting student information and marks.

<label>: Provides labels for input fields, improving accessibility.

<input>: Various input fields such as text for name and registration number, and number for marks with specified minimum and maximum values.

<style>: CSS styling for formatting and visual appeal.

SERVLET CODE:

request.getParameter("num1"): Retrieves the value of the specified parameter ("num1") from the HTTP request.

request.getParameter("num2"): Retrieves the value of the specified parameter ("num2") from the HTTP request.

request.getParameter("operation"): Retrieves the value of the specified parameter ("operation") from the HTTP request.

response.getWriter(): Returns a PrintWriter object that can send character text to the client.

JSP CODE:

<%@ page import="java.sql.*, java.io.*, java.util.*" %>: Imports necessary Java packages for the JSP file, including SQL, I/O, and utility classes.

<%@ page import="javax.servlet.*, javax.servlet.http.*" %>: Imports Servlet and HTTP packages for handling servlet-related functionality.

<% ... %>: Scriptlet tags enclose Java code within a JSP page. In this case, they handle data retrieval, GPA calculation, JDBC connection, SQL preparation, execution, and result output.

<%= ... %>: Expression tags evaluate the expression within and output the result to the client. Here, it's used to output dynamic content, such as success or error messages.

Step 3: Maintain the visitors count on the website.

HTML CODE:

<html>: Root element of the HTML document.

<body>: Contains the content of the HTML document.

<script>: May be used to manipulate cookies in JavaScript.

SERVLET CODE:

doGet(HttpServletRequest request, HttpServletResponse response): Handles the HTTP GET request. Retrieves the visitor count from the cookie, increments it, and sets the updated count in the response.

JSP CODE:

<%@ page import="javax.servlet.http.Cookie" %>: Imports the Cookie class for cookie manipulation.

<% ... %>: Scriptlet tags for embedding Java code in the JSP.

Step 4: Session management

HTML CODE:

`<form>`: Defines the login form.

`<label>`: Labels for input fields.

`<input>`: Input fields for username and password.

`<button>`: Submits the form for authentication.

SERVLET CODE:

`doPost(HttpServletRequest request, HttpServletResponse response)`: Handles the HTTP POST request to authenticate the user.

`createSession(String username)`: Creates a session for the authenticated user.

JSP CODE:

`<%@ page import="javax.servlet.http.HttpSession" %>`: Imports HttpSession class for session management.

`<c:if>`: Conditional tag to check if the user is authenticated.

`<c:out>`: Outputs dynamic content, such as the username.

`<a>`: Provides a link to log out and end the session.

Step 5: Redirecting to the Welcome Page

HTML CODE:

`<form>`: Defines an HTML form for user input.

`<label>`: Associates a label with a form element and provides a user-readable description.

`<input>`: Defines an input field within a form.

`type="text"`: Specifies the input field to accept single-line text input.

`type="submit"`: Specifies the input field as a submit button.

SERVLET CODE:

`protected void doPost(HttpServletRequest request, HttpServletResponse response)`: Handles the HTTP POST request. Retrieves data from the form, sets the response content type, and uses a PrintWriter to output HTML displaying the received user information.

JSP CODE:

`<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>`: JSP directive for setting language and page encoding.

`<title>`: Sets the title of the HTML document.

<div class="result">: Defines a styled container for the result.

<h2>: Defines a level-two heading for displaying the welcome message.

<p>: Defines a paragraph for displaying information.

<%= ... %>: Expression tags for embedding dynamic content in the HTML.

14.1) Write a Servlet and JSP code for calculator program

CODE:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Calculator</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    h2 {
      text-align: center;
      color: #333;
    }

    form {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    label {
      display: block;
      margin-bottom: 8px;
      color: #333;
    }
```

```

    input, select {
        width: 100%;
        padding: 8px;
        margin-bottom: 16px;
        box-sizing: border-box;
    }

    input[type="submit"] {
        background-color: Red;
        color: #fff;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
<form action="CalculatorServlet" method="post">
    <h2>Simple Calculator</h2>
    <label>Number 1:</label>
    <input type="text" name="num1" required>
    <br>
    <label>Number 2:</label>
    <input type="text" name="num2" required>
    <br>
    <label>Operation:</label>
    <select name="operation" required>
        <option value="add">Add</option>
        <option value="subtract">Subtract</option>
        <option value="multiply">Multiply</option>
        <option value="divide">Divide</option>
    </select>
    <br>
    <input type="submit" value="Calculate">
</form>
</body>
</html>

```

CalculatorServlet.java

```

import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/CalculatorServlet")
public class CalculatorServlet extends HttpServlet {

```

```

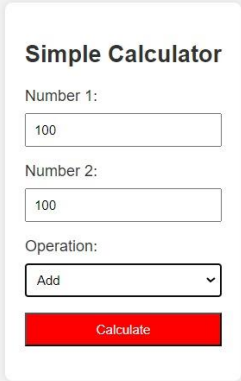
private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    int num1 = Integer.parseInt(request.getParameter("num1"));
    int num2 = Integer.parseInt(request.getParameter("num2"));
    String operation = request.getParameter("operation");
    int result = 0;
    switch (operation) {
        case "add":
            result = num1 + num2;
            break;
        case "subtract":
            result = num1 - num2;
            break;
        case "multiply":
            result = num1 * num2;
            break;
        case "divide":
            if (num2 != 0) {
                result = num1 / num2;
            } else {
                response.getWriter().println("Cannot divide by zero");
                return;
            }
            break;
    }
    response.getWriter().println("Result: " + result);
}
}

```

OUTPUT:

localhost:8080/WebApp



Simple Calculator

Number 1:

Number 2:

Operation:

Result: 200

JSP:

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Calculator</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    h2 {
      text-align: center;
      color: #333;
    }

    form {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    label {
      display: block;
      margin-bottom: 8px;
      color: #333;
    }
```

```

    input, select {
        width: 100%;
        padding: 8px;
        margin-bottom: 16px;
        box-sizing: border-box;
    }

    input[type="submit"] {
        background-color: #4caf50;
        color: #fff;
        cursor: pointer;
    }

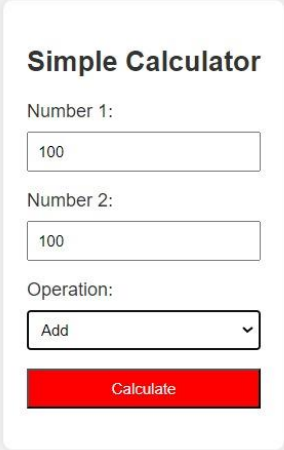
    input[type="submit"]:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
    <form action="" method="post">
        <h2>Simple Calculator</h2>
        <label for="num1">Number 1:</label>
        <input type="text" name="num1" value="<%= request.getParameter("num1") %>"
required>
        <label for="num2">Number 2:</label>
        <input type="text" name="num2" value="<%= request.getParameter("num2") %>"
required>
        <label for="operation">Operation:</label>
        <select name="operation" required>
            <option value="add" <%= "add".equals(request.getParameter("operation")) ?
"selected" : "" %>>Add</option>
            <option value="subtract" <%=
"subtract".equals(request.getParameter("operation")) ? "selected" : ""
%>>Subtract</option>
            <option value="multiply" <%=
"multiply".equals(request.getParameter("operation")) ? "selected" : ""
%>>Multiply</option>
            <option value="divide" <%= "divide".equals(request.getParameter("operation")) ?
"selected" : "" %>>Divide</option>
        </select>
        <br>
        <input type="submit" value="Calculate">
    </form>
    <!-- Java code for calculation --%>
    <%
        if (request.getMethod().equalsIgnoreCase("POST")) {
            int num1 = Integer.parseInt(request.getParameter("num1"));
            int num2 = Integer.parseInt(request.getParameter("num2"));
            String operation = request.getParameter("operation");

            int result = 0;
            switch (operation) {

```

```
        case "add":
            result = num1 + num2;
            break;
        case "subtract":
            result = num1 - num2;
            break;
        case "multiply":
            result = num1 * num2;
            break;
        case "divide":
            if (num2 != 0) {
                result = num1 / num2;
            } else {
                out.println("Cannot divide by zero");
                return;
            }
            break;
    }
    out.println("<p>Result: " + result + "</p>");
}
%>
</body>
</html>
```

OUTPUT:



Simple Calculator

Number 1:

Number 2:

Operation:

localhost:8080/WebApp

Result: 200

2) Write a Servlet and JSP code to fetch and store student details (Name, regno, subject name and marks, gpa) using database

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Details</title>
</head>
<body>
<h2>Enter Student Details</h2>
<form action="add" method="post">
  Name: <input type="text" name="name"><br>
  Registration Number: <input type="text" name="regNo"><br>
  Subject 1 Name: <input type="text" name="subjectName1"><br>
  Marks for Subject 1: <input type="text" name="marks1"><br>
  Subject 2 Name: <input type="text" name="subjectName2"><br>
  Marks for Subject 2: <input type="text" name="marks2"><br>
  Subject 3 Name: <input type="text" name="subjectName3"><br>
  Marks for Subject 3: <input type="text" name="marks3"><br>
  Subject 4 Name: <input type="text" name="subjectName4"><br>
  Marks for Subject 4: <input type="text" name="marks4"><br>
  GPA: <input type="text" name="gpa"><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

StudentServlet.java

```
package webapp;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

@WebServlet("/add")
public class StudentServlet extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
```

```

IOException, ServletException {
    final String JDBC_URL = "jdbc:oracle:thin:@localhost:XE";
    final String DB_USERNAME = "system";
    final String DB_PASSWORD = "kiran";
    // Retrieving parameters from the form
    String name = request.getParameter("name");
    int regNo = Integer.parseInt(request.getParameter("regNo"));
    String subjectName1 = request.getParameter("subjectName1");
    int marks1 = Integer.parseInt(request.getParameter("marks1"));
    String subjectName2 = request.getParameter("subjectName2");
    int marks2 = Integer.parseInt(request.getParameter("marks2"));
    String subjectName3 = request.getParameter("subjectName3");
    int marks3 = Integer.parseInt(request.getParameter("marks3"));
    String subjectName4 = request.getParameter("subjectName4");
    int marks4 = Integer.parseInt(request.getParameter("marks4"));
    double gpa = Double.parseDouble(request.getParameter("gpa"));

    request.setAttribute("name", name);
    request.setAttribute("regNo", regNo);
    request.setAttribute("subjectName1", subjectName1);
    request.setAttribute("marks1", marks1);
    request.setAttribute("subjectName2", subjectName2);
    request.setAttribute("marks2", marks2);
    request.setAttribute("subjectName3", subjectName3);
    request.setAttribute("marks3", marks3);
    request.setAttribute("subjectName4", subjectName4);
    request.setAttribute("marks4", marks4);
    request.setAttribute("gpa", gpa);

    try (Connection conn = DriverManager.getConnection(JDBC_URL, DB_USERNAME,
DB_PASSWORD)) {
        // SQL INSERT statement
        String insertQuery = "INSERT INTO student (name, reg_no, subject1, marks1, subject2,
marks2, subject3, marks3, subject4, marks4, gpa) "
            + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(insertQuery)) {
            // Set parameters for the INSERT statement
            pstmt.setString(1, name);
            pstmt.setInt(2, regNo);
            pstmt.setString(3, subjectName1);
            pstmt.setInt(4, marks1);
            pstmt.setString(5, subjectName2);
            pstmt.setInt(6, marks2);
            pstmt.setString(7, subjectName3);
            pstmt.setInt(8, marks3);

```

```

        pstmt.setString(9, subjectName4);
        pstmt.setInt(10, marks4);
        pstmt.setDouble(11, gpa);
        pstmt.executeUpdate();
    }
} catch (SQLException e) {
    // Handle any database errors
    e.printStackTrace();
    // Redirect to an error page or display an error message as needed
    response.sendRedirect("error.jsp");
    return;
}

response.setContentType("text/html");

// Creating a PrintWriter object
PrintWriter out = response.getWriter();

// Printing student details in HTML format
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Student Details</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Successfully submitted!!!</h1>");
out.println("<h2>Student Details</h2>");
out.println("<p>Name: " + name + "</p>");
out.println("<p>Registration Number: " + regNo + "</p>");
out.println("<p>Subject 1: " + subjectName1 + " - Marks: " + marks1 + "</p>");
out.println("<p>Subject 2: " + subjectName2 + " - Marks: " + marks2 + "</p>");
out.println("<p>Subject 3: " + subjectName3 + " - Marks: " + marks3 + "</p>");
out.println("<p>Subject 4: " + subjectName4 + " - Marks: " + marks4 + "</p>");
out.println("<p>GPA: " + gpa + "</p>");
out.println("<p><em>Data inserted in database</em></p>");
out.println("</body>");
out.println("</html>");
// Redirecting to a new page to display the details
response.sendRedirect("success.jsp");
}
}

```

success.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>

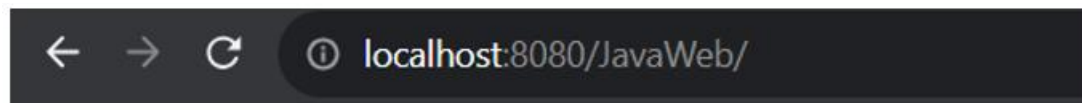
```

```

<html>
<head>
  <title>Student Details</title>
</head>
<body>
  <h2>Student Details</h2>
  <p>Name: <%= request.getAttribute("name") %></p>
  <p>Registration Number: <%= request.getAttribute("regNo") %></p>
  <p>Subject 1: <%= request.getAttribute("subjectName1") %> - Marks: <%=
request.getAttribute("marks1") %></p>
  <p>Subject 2: <%= request.getAttribute("subjectName2") %> - Marks: <%=
request.getAttribute("marks2") %></p>
  <p>Subject 3: <%= request.getAttribute("subjectName3") %> - Marks: <%=
request.getAttribute("marks3") %></p>
  <p>Subject 4: <%= request.getAttribute("subjectName4") %> - Marks: <%=
request.getAttribute("marks4") %></p>
  <p>GPA: <%= request.getAttribute("gpa") %></p>
</body>
</html>

```

OUTPUT:



Enter Student Details

Name:

Registration Number:

Subject 1 Name:

Marks for Subject 1:

Subject 2 Name:

Marks for Subject 2:

Subject 3 Name:

Marks for Subject 3:

Subject 4 Name:

Marks for Subject 4:

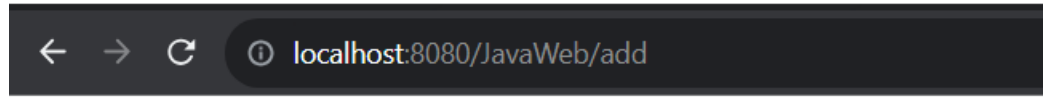
GPA:

```

1 row created.
SQL> select * from student;

```

ID	NAME	REG_NO	SUBJECT1	MARKS1	SUBJECT2	MARKS2	SUBJECT3
RKS3	SUBJECT4	MARKS4	GPA				
4	Vijai Suria M	2021503568	Java	99	OOAD	100	Math
88	DBMS	99	8				



Successfully submitted!!!

Student Details

Name: Vijai Suria M

Registration Number: 2021503568

Subject 1: Java - Marks: 99

Subject 2: OOAD - Marks: 100

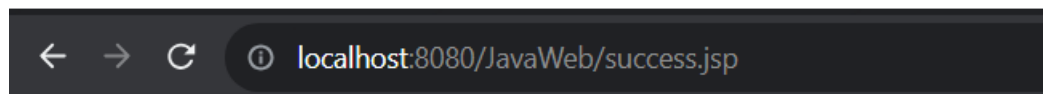
Subject 3: Math - Marks: 88

Subject 4: DBMS - Marks: 99

GPA: 8.0

Data inserted in database

OUTPUT: JSP



Student Details

Name: Vijai Suria M

Registration Number: 2021503568

Subject 1: Java - Marks: 99

Subject 2: OOAD - Marks: 100

Subject 3: Math - Marks: 88

Subject 4: DBMS - Marks: 99

GPA: 8.0

Data inserted in database

14.3) Write a Servlet and JSP code to maintain visitors count of a website using cookies

Servlet:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Visitor Count Example</title>
<style>
  body {
    font-family: 'Arial', sans-serif;
    background-color: #f8f9fa;
    margin: 20px;
    padding: 20px;
    text-align: center;
  }

  h1 {
    color: red;
  }
  input[type="submit"] {
    padding: 10px 20px;
    font-size: 18px;
    background-color: #007bff;
    color: #fff;
    border: none;
    cursor: pointer;
  }

  input[type="submit"]:hover {
    background-color: #0056b3;
  }
</style>
</head>
<body>
<div class="container">
  <h1>Welcome to V Priyadarshni's Website!</h1>
  <h1>2021503538</h1>
  <p>This is a simple example to demonstrate visitor count using cookies.</p>
  <p>Click the button to view the visitor count:</p>
  <form action="VisitorCountServlet" method="GET">
    <input type="submit" value="Get Visitor Count">
  </form>
</div>
</body>
</html>

```

VisitorCountServlet.java

```

import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.*;

@WebServlet("/VisitorCountServlet")
public class VisitorCountServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;
  protected void doGet(HttpServletRequest request, HttpServletResponse response)

```

```

        throws ServletException, IOException {
// Get the current visitor count from the cookie
int count = 1;
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for (Cookie cookie : cookies) {
        if (cookie.getName().equals("visitorCount")) {
            count = Integer.parseInt(cookie.getValue());
            count++;
        }
    }
}
// Set the new visitor count in the cookie
Cookie cookie = new Cookie("visitorCount", String.valueOf(count));
cookie.setMaxAge(24 * 60 * 60); // Cookie will expire in 1 day
response.addCookie(cookie);
// Set the response content type
response.setContentType("text/html");
// Write the HTML response
PrintWriter out = response.getWriter();
out.println("<html><head><title>Visitor Count</title></head><body style=\"text-align: center; font-family: 'Arial', sans-serif; background-color: #f8f9fa; margin: 20px; padding: 20px;\">");
out.println("<h2 style=\"color: #007bff;\">Visitor Count: " + count + "</h2>");
out.println("</body></html>");

    }
}

```

OUTPUT:

🔗 localhost:8080/CalcServlet/

Welcome to Vijai's Website!

2021503568

This is a simple example to demonstrate visitor count using cookies.

Click the button to view the visitor count:

Get Visitor Count

🔗 localhost:8080/CalcServlet/VisitorCountServlet?

Visitor Count: 12

JSP:

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Visitor Count Example</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background-color: #f8f9fa;
      margin: 20px;
      padding: 20px;
      text-align: center;
    }
    h1 {
      color: red;
    }
    input[type="submit"] {
      padding: 10px 20px;
      font-size: 18px;
      background-color: #007bff;
      color: #fff;
      border: none;
      cursor: pointer;
    }

    input[type="submit"]:hover {
      background-color: #0056b3;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Welcome to V Priyadarshni's Website made using JSP!</h1>
    <h1>2021503538</h1>
    <p>This is a simple example to demonstrate visitor count using cookies.</p>
    <p>Click the button to view the visitor count:</p>
    <form action="visitorCount.jsp" method="GET">
      <input type="submit" value="Get Visitor Count">
    </form>
  </div>
</body>
</html>
```

visitorCount.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.io.*, java.util.*"%>
<%@ page import="javax.servlet.*, javax.servlet.http.*"%>

<%
    // Get the current visitor count from the cookie
    int count = 1;
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("visitorCount")) {
                count = Integer.parseInt(cookie.getValue());
                count++;
            }
        }
    }

    // Set the new visitor count in the cookie
    Cookie cookie = new Cookie("visitorCount", String.valueOf(count));
    cookie.setMaxAge(24 * 60 * 60); // Cookie will expire in 1 day
    response.addCookie(cookie);
%>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visitor Count</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f8f9fa;
            margin: 20px;
            padding: 20px;
            text-align: center;
        }

        h2 {
            color: #007bff;
        }
    </style>
</head>
<body>
    <h2>Visitor Count: <%= count %></h2>
</body>
</html>
```

OUTPUT:

localhost:8080/CalcServlet/

Welcome to Vijai's Website made using JSP!

2021503568

This is a simple example to demonstrate visitor count using cookies.

Click the button to view the visitor count:

Get Visitor Count

localhost:8080/CalcServlet/visitorCount.jsp?

Visitor Count: 15

14.4) Write a Servlet and JSP code to maintain a session for user authentication and to welcome the user

Servlet:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    form {
      max-width: 300px;
      margin: 20px auto;
      border: 1px solid #A9A9A9;
      padding: 15px;
    }

    label {
      display: block;
      margin-bottom: 5px;
    }
  </style>

```

```

    input {
        width: 80%;
        padding: 8px;
        margin-bottom: 10px;
    }

    input[type="submit"] {
        background-color: #007bff;
        color: #fff;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #0056b3;
    }
</style>
</head>
<body>
<form action="authenticate" method="post">
    <label for="user">Enter username:</label>
    <input type="text" id="user" name="user" required>

    <label for="pwd">Enter password:</label>
    <input type="password" id="pwd" name="pwd" required>

    <input type="submit" value="Sign in">
</form>
</body>
</html>

```

Session.java

```

package webapp;
import java.io.IOException;
import java.io.PrintWriter;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.*;

@WebServlet("/authenticate")
public class Session extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res) {
        try {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            String name = req.getParameter("user");
            String pwd = req.getParameter("pwd");

            HttpSession session=req.getSession();
            session.setAttribute("Name", name);
            if (name.equals("Vijai") && pwd.equalsIgnoreCase("123")) {

```


OUTPUT:

localhost:8080/UserAuth/

Enter username:

Enter password:

localhost:8080/UserAuth/session

Hello Vijai

JSP:

index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <title>Login Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    form {
      max-width: 300px;
      margin: 20px auto;
      border: 1px solid #A9A9A9;
      padding: 15px;
    }

    label {
      display: block;
      margin-bottom: 5px;
    }
    input[type="submit"]:hover {
      background-color: #0056b3;
```



```

    }
</style>
</head>
<body>
  <form action="authenticate.jsp" method="post">
    <label for="user">Enter username:</label>
    <input type="text" id="user" name="user" required>

    <label for="pwd">Enter password:</label>
    <input type="password" id="pwd" name="pwd" required>

    <input type="submit" value="Sign in">
  </form>
</body>
</html>

```

authenticate.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>

<%
  String name = request.getParameter("user");
  String pwd = request.getParameter("pwd");

  HttpSession sessionObject = request.getSession();

  if ("Vijai".equals(name) && "123".equals(pwd)) {
    sessionObject.setAttribute("Name", name);
    response.sendRedirect("session.jsp");
  } else {
    out.println("<font color='red'><b>You have entered incorrect password</b></font>");
    request.getRequestDispatcher("index.jsp").include(request, response);
  }
%>

```

session.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>

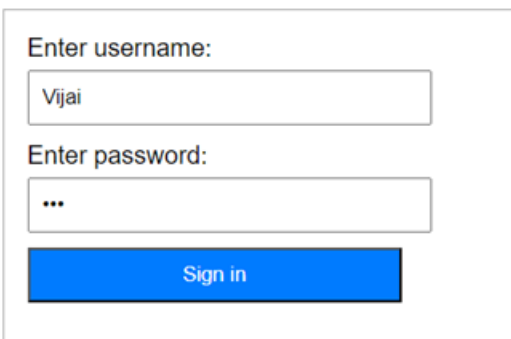
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Session Page</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background-color: #f8f9fa;
      text-align: center;
      margin: 20px;
    }
  </style>

```

```
        h2 {
            color: #007bff;
        }
    </style>
</head>
<body>
    <h2>Hello <%= session.getAttribute("Name") %></h2>
</body>
</html>
```

OUTPUT:

localhost:8080/UserAuth/



Enter username:

Vijai

Enter password:

...

Sign in

localhost:8080/UserAuth/session.jsp

Hello Vijai

14.5) Write a Servlet and JSP code to redirect to welcome page based on user credentials

CODE: (SERVLET)

```
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Information Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
```

```

        background-color: #f4f4f4;
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
    }

    form {
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        width: 300px;
        text-align: center;
    }

    label {
        display: block;
        margin-bottom: 8px;
        color: #333;
    }

    input[type="submit"] {
        background-color: #4caf50;
        color: #fff;
        cursor: pointer;
    }

    input[type="submit"]:hover {
        background-color: #45a049;
    }
</style>
</head>
<body>
<form action="Welcome" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <br>
    <label for="regno">Registration Number:</label>
    <input type="text" id="regno" name="regno" required>
    <br>
    <input type="submit" value="Submit">
</form>
</body>
</html>

```

SERVLET CODE:

```

import java.io.*;

import jakarta.servlet.annotation.WebServlet;

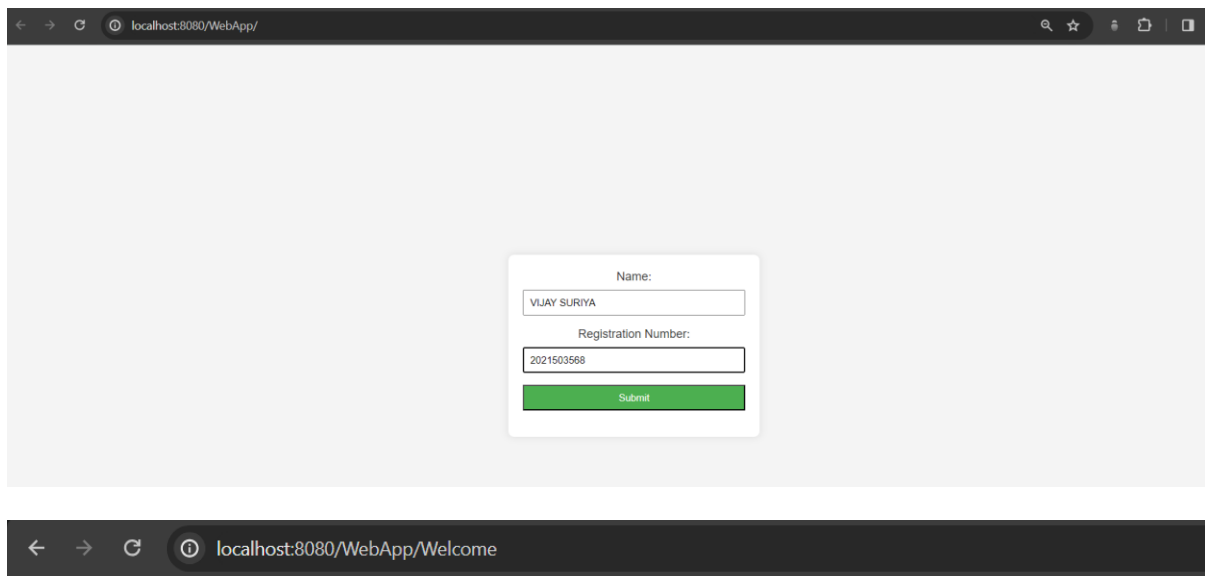
```

```

import jakarta.servlet.http.*;
@WebServlet("/Welcome")
public class Welcome extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        // Retrieve data from the form
        String name = request.getParameter("name");
        String regno = request.getParameter("regno");
        // Display the received data using HTML tags
        out.println("<html>");
        out.println("<head><title>User Information Display</title></head>");
        out.println("<body>");
        out.println("<h2>User Information:</h2>");
        out.println("<p><strong>Name:</strong> " + name + "</p>");
        out.println("<p><strong>Registration Number:</strong> " + regno + "</p>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
}

```

OUTPUT:



localhost:8080/WebApp/

Name: VIJAY SURIYA

Registration Number: 2021503568

Submit

localhost:8080/WebApp/Welcome

User Information:

Name: VIJAY SURIYA

Registration Number: 2021503568

JSP:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Information Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    form {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      width: 300px;
      text-align: center;
    }

    label {
      display: block;
      margin-bottom: 8px;
      color: #333;
    }

    input {
      width: 100%;
      padding: 8px;
      margin-bottom: 16px;
      box-sizing: border-box;
    }

    input[type="submit"] {
      background-color: #4caf50;
      color: #fff;
      cursor: pointer;
    }

    input[type="submit"]:hover {
      background-color: #45a049;
    }
  </style>
```

```

</head>
<body>
<form action="Welcome.jsp" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required>
  <br>
  <label for="regno">Registration Number:</label>
  <input type="text" id="regno" name="regno" required>
  <br>
  <input type="submit" value="Submit">
</form>
</body>
</html>

```

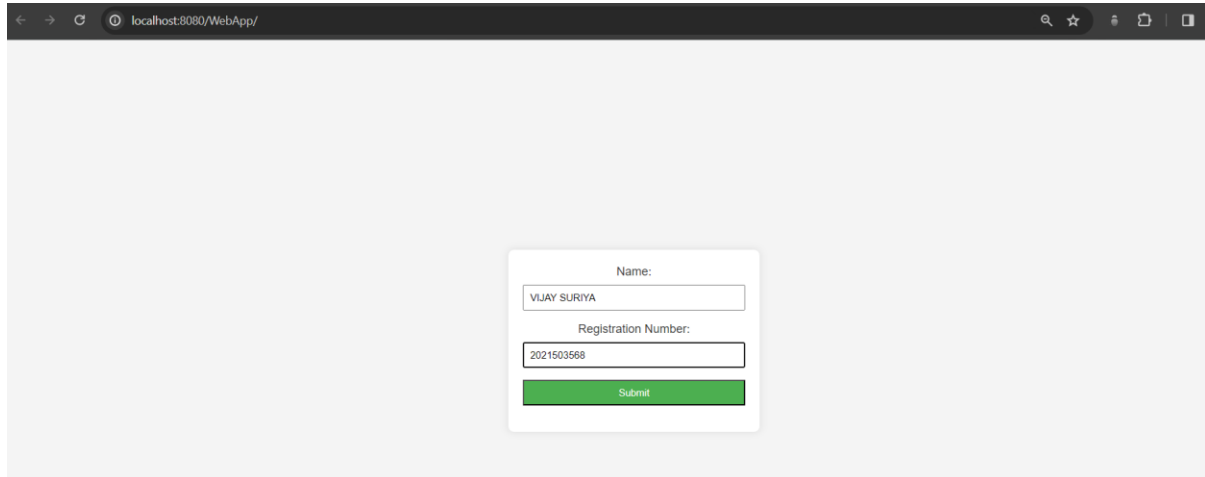
JSP

```

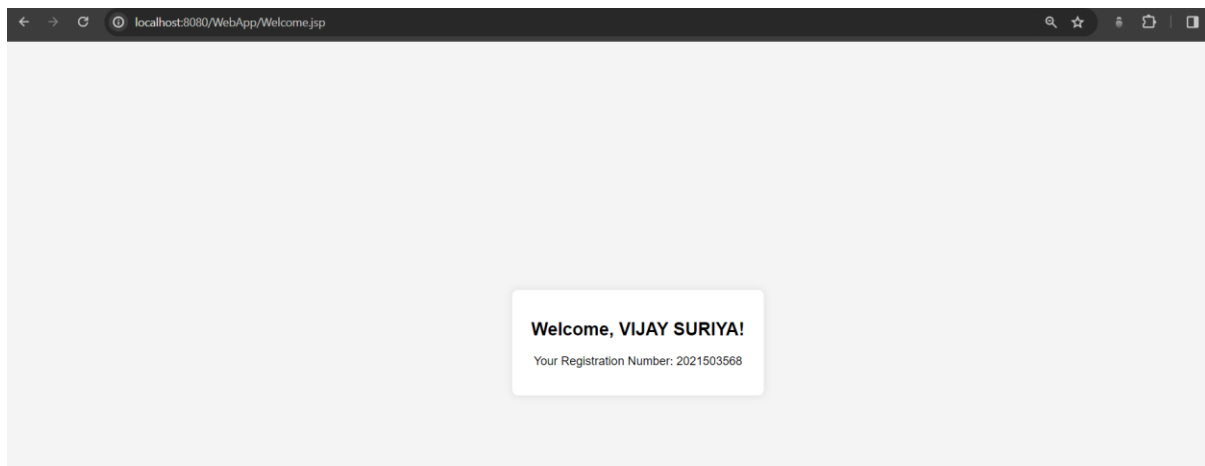
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Welcome Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .result {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      width: 300px;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="result">
    <h2>Welcome, <%= request.getParameter("name") %>!</h2>
    <p>Your Registration Number: <%= request.getParameter("regno") %></p>
  </div>
</body>
</html>

```

OUTPUT:



A screenshot of a web browser window with the address bar showing "localhost:8080/WebApp/". The browser interface includes back, forward, and refresh buttons. The main content area is light gray and contains a white registration form. The form has two input fields: "Name:" with the value "VIJAY SURIYA" and "Registration Number:" with the value "2021503568". Below the fields is a green "Submit" button.



RESULT:

Thus, the Servlets and JSP in java in implemented and executed successfully.