Ex.No: 06 **DATE: 04-10-23**

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 Inheritannce**

- 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.

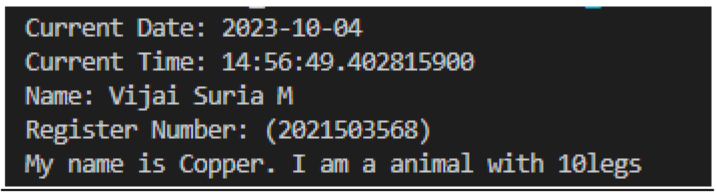
- Chat(User user1, User user2, int maxMessages) - Initializes chat participants and messages.
- addMessage(Message message) - Adds a message to the chat.
- displayChatHistory() - Displays the chat history.

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 youu!! \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:



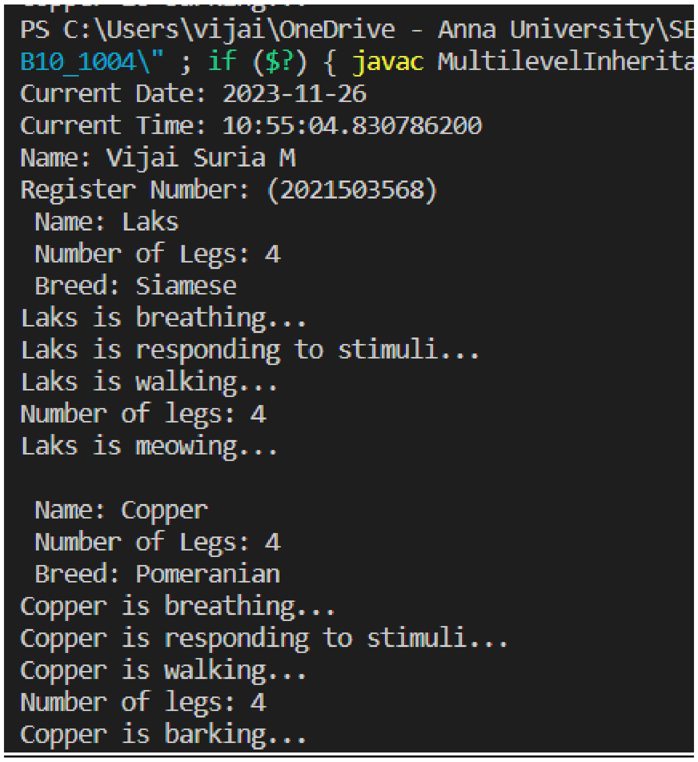
```
Current Date: 2023-10-04
Current Time: 14:56:49.402815900
Name: Vijai Suria M
Register Number: (2021503568)
My name is Copper. I am a animal with 10legs
```

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 MultilevelInherita
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...
```

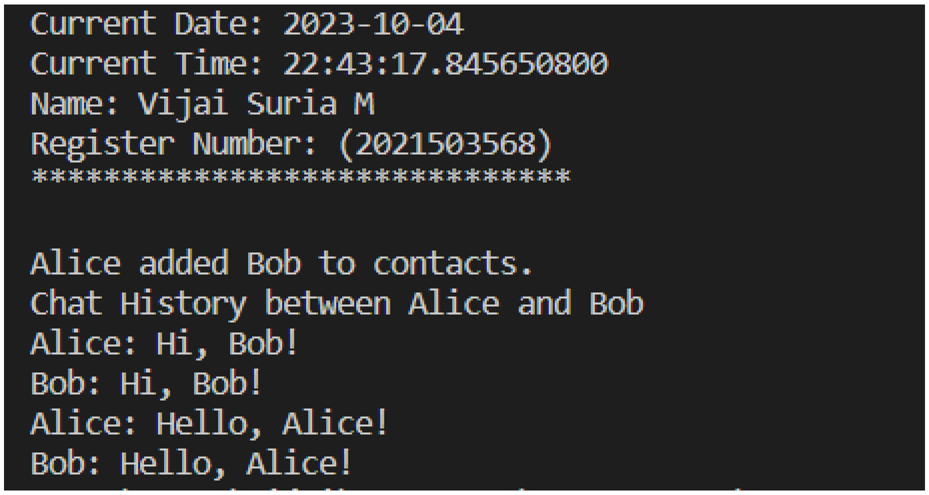
6.3 Write a program in Java to create messaging service like WhatsApp that uses single inheritance, multilevel inheritance, and hierarchical inheritance.

<u>CODE:</u>

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

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.

Ex.No: 07 **DATE: 14-10-23**

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**

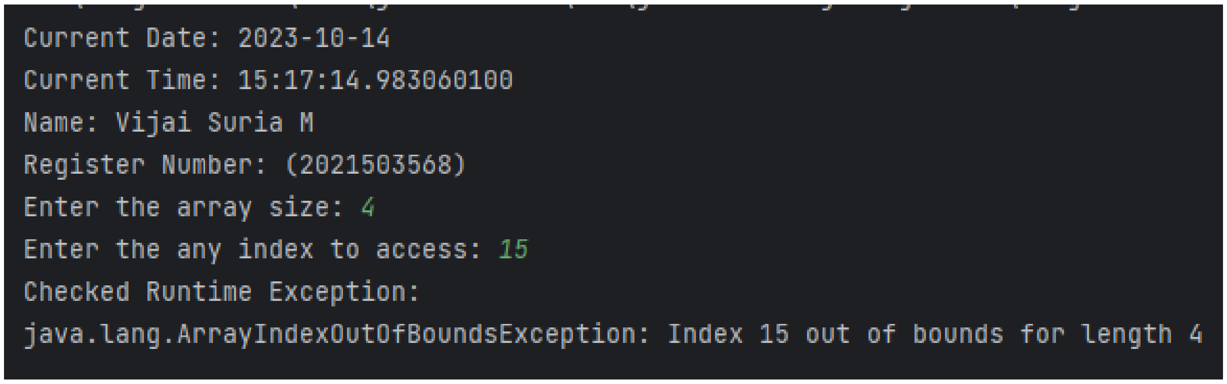
- Scanner.nextLine(): Reads a line of text from the console input.
- validateEmail(String email) throws InvalidEmailException: Checks if the given email contains the '@' symbol; if not, throws an InvalidEmailException.
- InvalidEmailException extends Exception: Represents an exception specific to invalid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



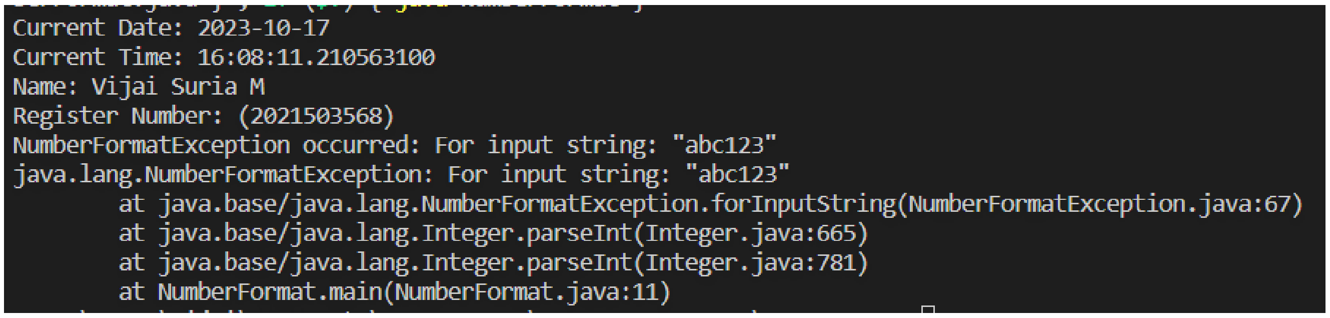
```
Current Date: 2023-10-14
Current Time: 15:17:14.983060100
Name: Vijai Suria M
Register Number: (2021503568)
Enter the array size: 4
Enter the any index to access: 15
Checked Runtime Exception:
java.lang.ArrayIndexOutOfBoundsException: Index 15 out of bounds for length 4
```

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)
```

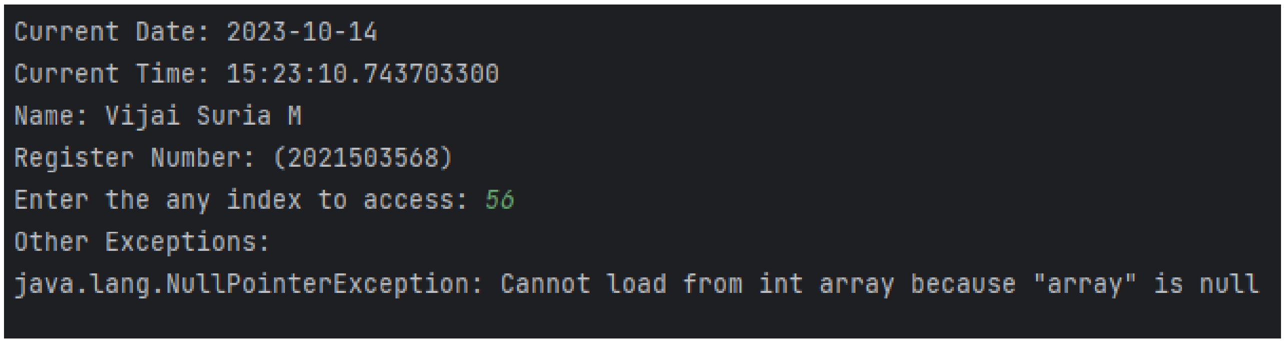
7.3) Illustrate and demonstrate the Null Pointer Exception

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:



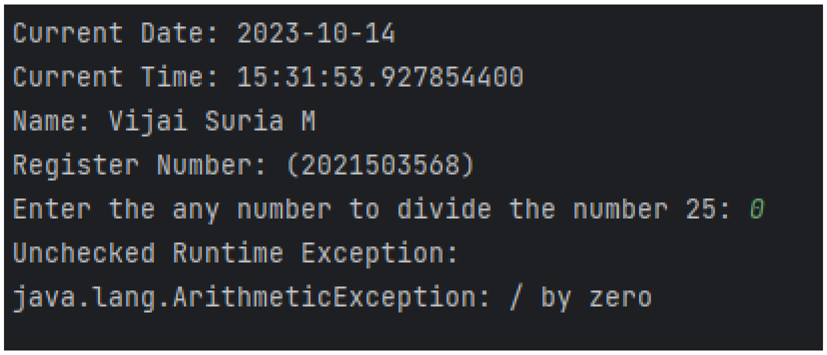
```
Current Date: 2023-10-14
Current Time: 15:23:10.743703300
Name: Vijai Suria M
Register Number: (2021503568)
Enter the any index to access: 56
Other Exceptions:
java.lang.NullPointerException: Cannot load from int array because "array" is null
```

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:



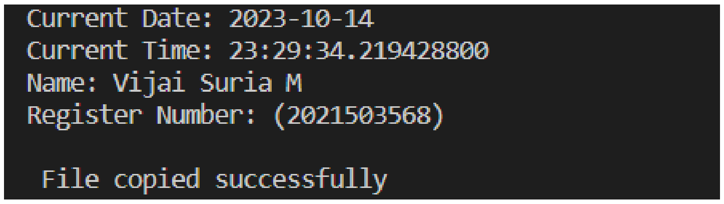
```
Current Date: 2023-10-14
Current Time: 15:31:53.927854400
Name: Vijai Suria M
Register Number: (2021503568)
Enter the any number to divide the number 25: 0
Unchecked Runtime Exception:
java.lang.ArithmeticException: / by zero
```

7.5) Illustrate and demonstrate the File Not Found Exception

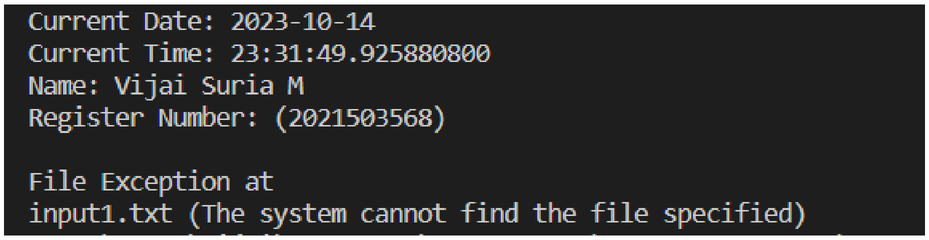
CODE:

```
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
```



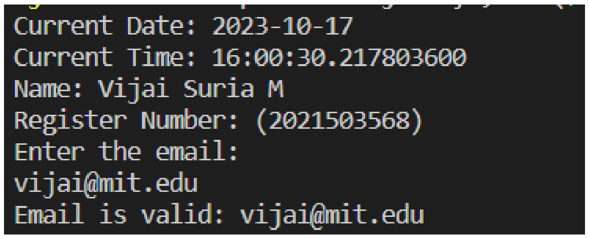
```
Current Date: 2023-10-14
Current Time: 23:31:49.925880800
Name: Vijai Suria M
Register Number: (2021503568)

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

7.6) Illustrate and demonstrate the User defined email validator:

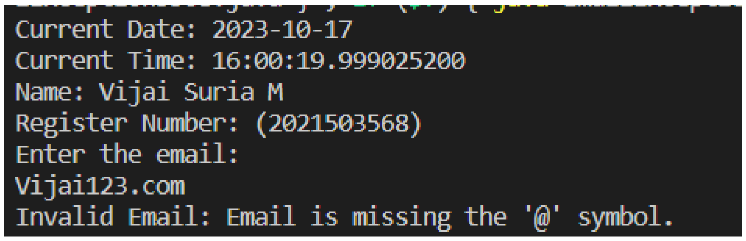
CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
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: " + LocalTime.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
```



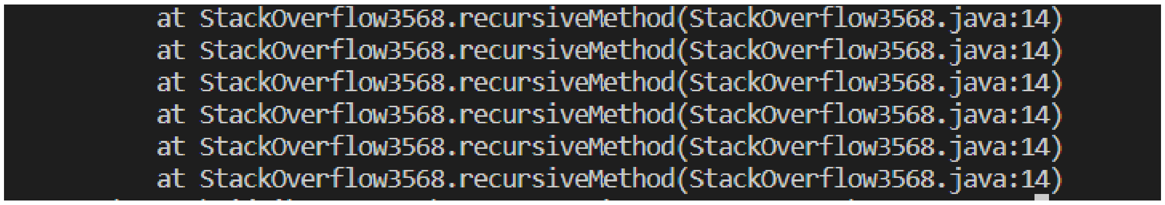
```
Current Date: 2023-10-17
Current Time: 16:00:19.999025200
Name: Vijai Suria M
Register Number: (2021503568)
Enter the email:
Vijai123.com
Invalid Email: Email is missing the '@' symbol.
```

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.

Ex.No: 08 **DATE: 14-10-23**

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**

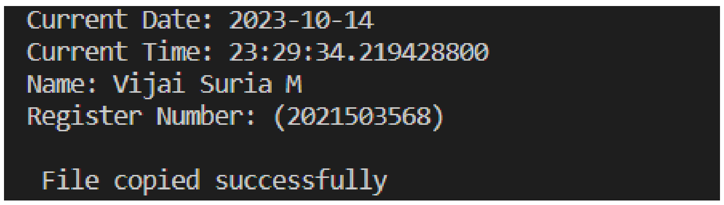
- FileOutputStream(String name): Opens an output stream to write to a file with the specified name.
- ObjectOutputStream(OutputStream out): Constructs an ObjectOutputStream that writes to the specified OutputStream.
- ObjectOutputStream.writeObject(Object obj): Writes the specified object to the ObjectOutputStream.
- 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:



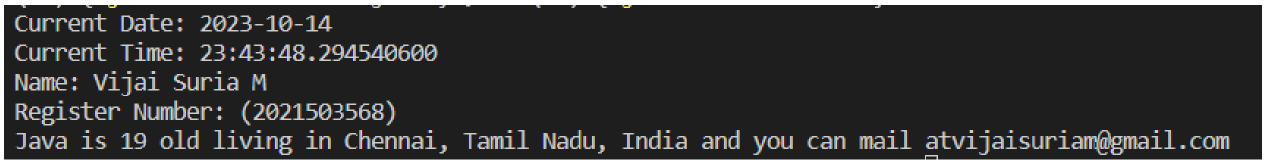
```
Current Date: 2023-10-14
Current Time: 23:29:34.219428800
Name: Vijai Suria M
Register Number: (2021503568)

 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";
        FileOutputStreamfo = new FileOutputStream(Filename);
        ObjectOutputStreamos = 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.

Ex.No: 09 **DATE: 01-11-23**

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:

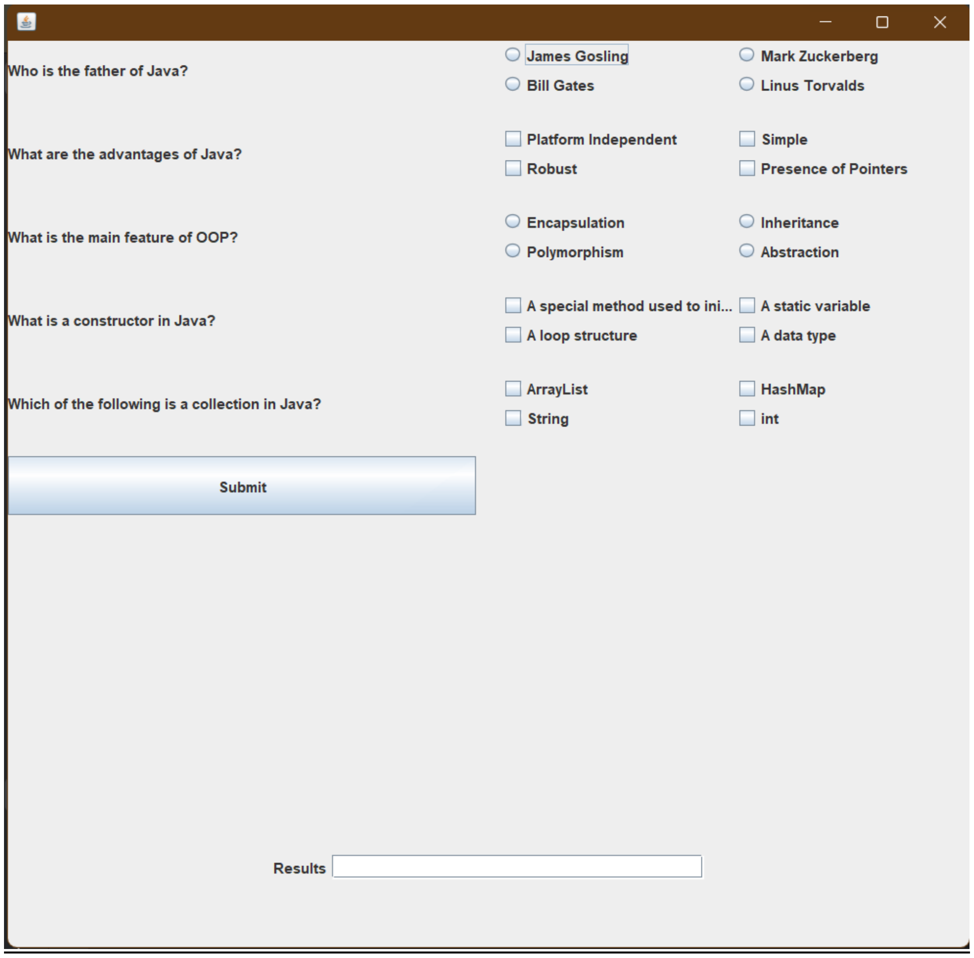
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class QuizApp_3568 extends JFrame implements ActionListener {
   JPanel qPanel, resPanel;
   JLabel ql1, ql2, ql3, ql4, ql5, 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;

      ql1 = 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]);

        ql2 = 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]);

        ql3 = 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]);

        ql4 = 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");
```

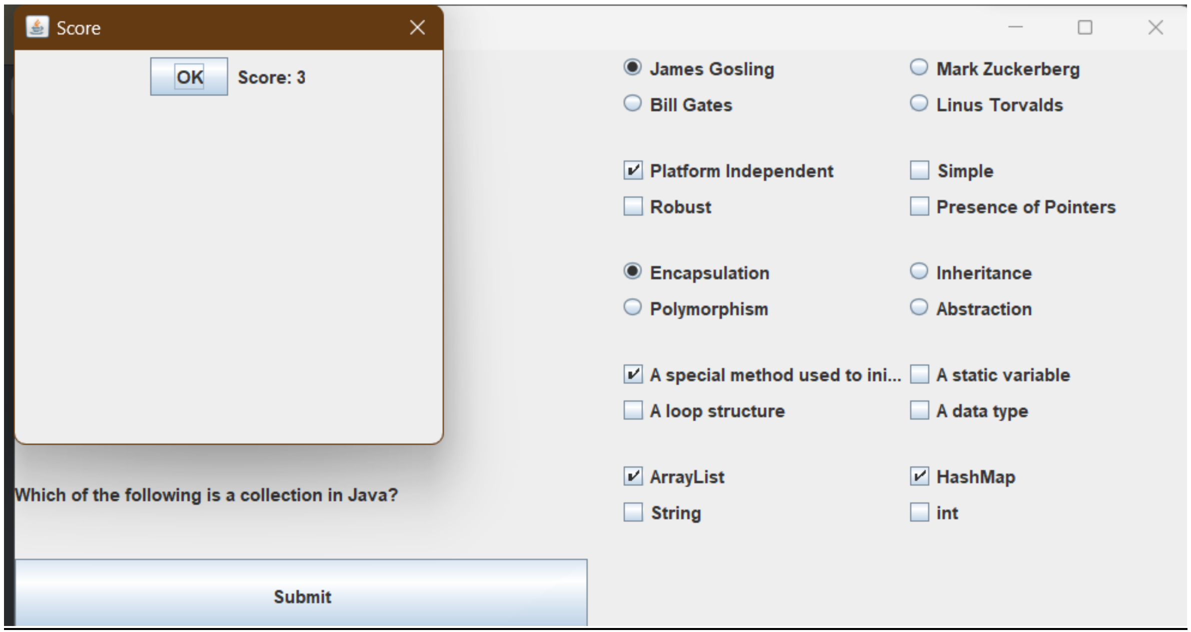
```
checkBoxes[1] = new JCheckBox("A static variable");
checkBoxes[2] = new JCheckBox("A loop structure");
checkBoxes[3] = new JCheckBox("A data type");
q4.add(checkBoxes[0]);
q4.add(checkBoxes[1]);
q4.add(checkBoxes[2]);
q4.add(checkBoxes[3]);

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:



```
Who is the father of Java?                 ○ James Gosling                  ○ Mark Zuckerberg
                                           ○ Bill Gates                     ○ Linus Torvalds

What are the advantages of Java?           ☐ Platform Independent           ☐ Simple
                                           ☐ Robust                         ☐ Presence of Pointers

What is the main feature of OOP?           ○ Encapsulation                  ○ Inheritance
                                           ○ Polymorphism                   ○ Abstraction

What is a constructor in Java?             ☐ A special method used to ini...☐ A static variable
                                           ☐ A loop structure               ☐ A data type

Which of the following is a collection in Java?  ☐ ArrayList                ☐ HashMap
                                           ☐ String                         ☐ int

[ Submit ]

Results [                    ]
```



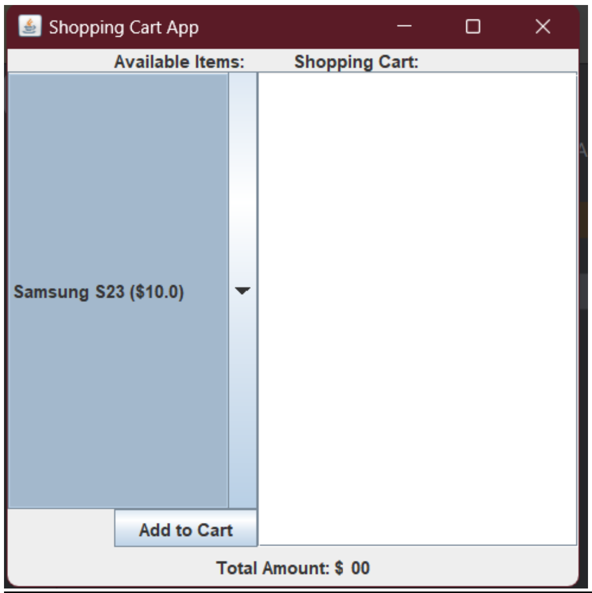
```
Score
[ OK ]  Score: 3

                                           ● 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

Which of the following is a collection in Java?  ☑ ArrayList                ☑ HashMap
                                           ☐ String                         ☐ int

[ Submit ]
```

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.

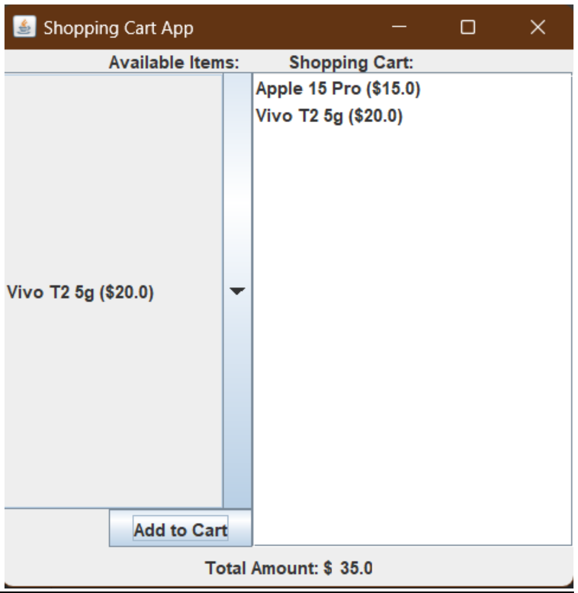
CODE:

```
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:



Total Amount: $ 00



Apple 15 Pro ($15.0)
Vivo T2 5g ($20.0)

Total Amount: $ 35.0

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

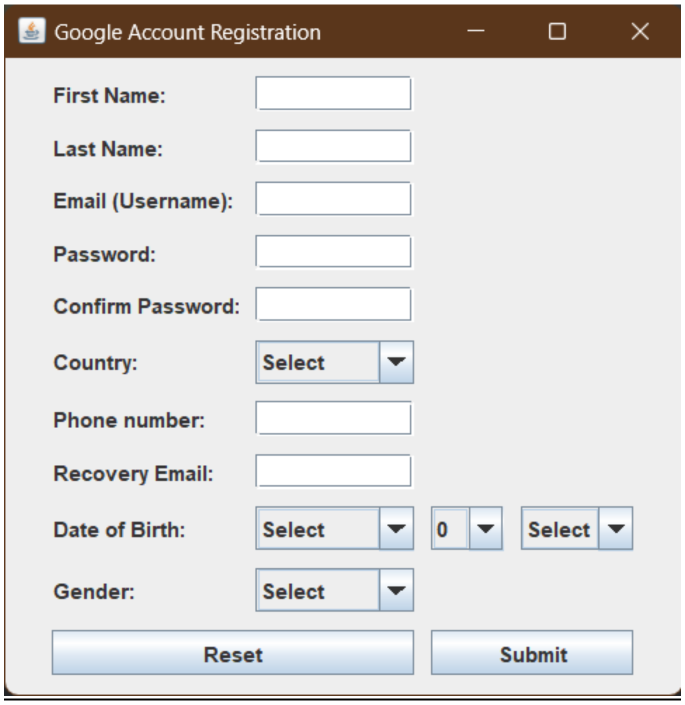
CODE:

```
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" });
```

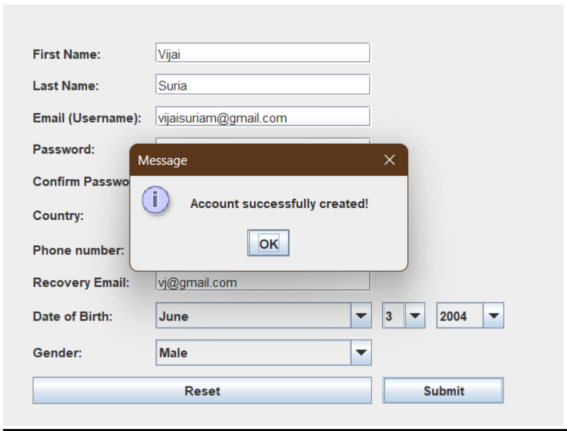
```
        JComboBox<String> dayComboBox = new JComboBox<>(new String[] { "Select"
});
        JComboBox<String> yearComboBox = new JComboBox<>(new String[] { "Select"
});
        // Populate dayComboBox with days 1 to 31 initially
        String[] daysInitial = new String[32];
        for (int i = 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;
        }
    }
}
```

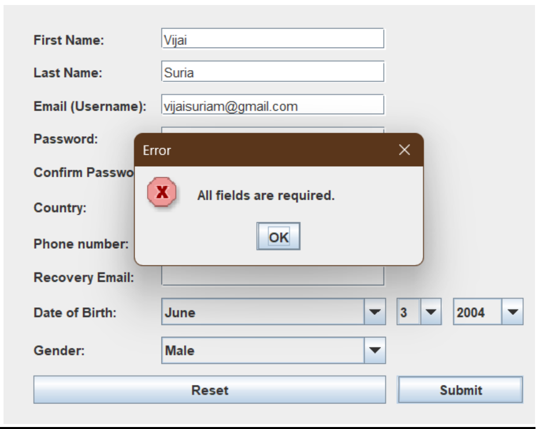
OUTPUT:


Google Account Registration

First Name:
Last Name:
Email (Username):
Password:
Confirm Password:
Country: Select
Phone number:
Recovery Email:
Date of Birth: Select 0 Select
Gender: Select


First Name: Vijai
Last Name: Suria
Email (Username): vijaisuriam@gmail.com
Password:
Confirm Passwo
Country:
Phone number:
Recovery Email: vj@gmail.com
Date of Birth: June 3 2004
Gender: Male

Message
Account successfully created!


First Name: Vijai
Last Name: Suria
Email (Username): vijaisuriam@gmail.com
Password:
Confirm Passwo
Country:
Phone number:
Recovery Email:
Date of Birth: June 3 2004
Gender: Male

Error
All fields are required.

RESULT:

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

Ex.No: 10 **DATE: 14-10-23**

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.

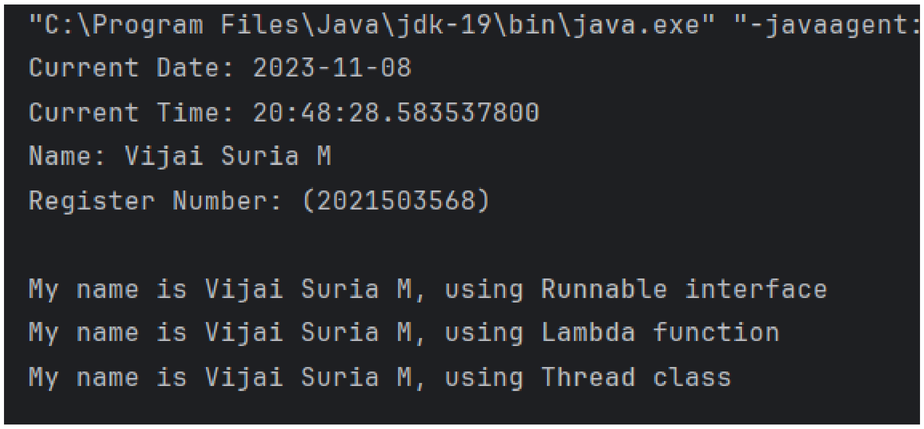
5. start() - A method from the Thread class used to start the execution of a thread. It initiates the execution of the run() method of the thread.
6. e.printStackTrace() - A method that prints the stack trace of an exception to the console. It's called on the IOException object (e) to display detailed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



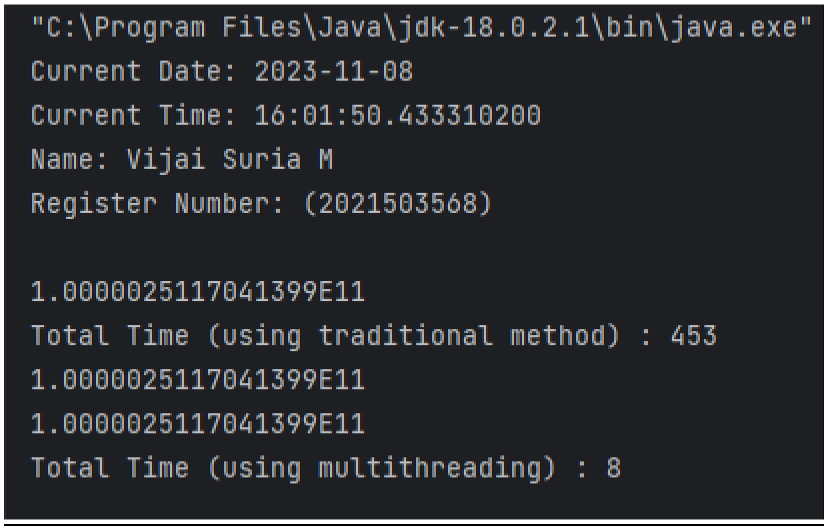
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:
Current Date: 2023-11-08
Current Time: 20:48:28.583537800
Name: Vijai Suria M
Register Number: (2021503568)

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.LocalTime;
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: " + LocalTime.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:



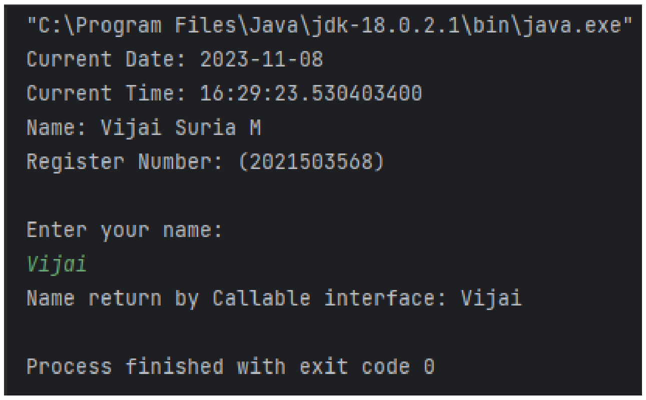
```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-08
Current Time: 16:01:50.433310200
Name: Vijai Suria M
Register Number: (2021503568)

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:



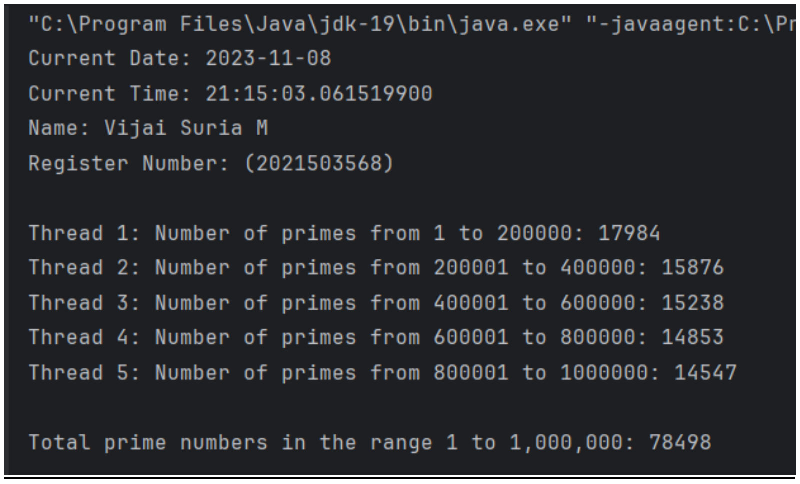
```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe"
Current Date: 2023-11-08
Current Time: 16:29:23.530403400
Name: Vijai Suria M
Register Number: (2021503568)

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;
ExecutorServiceexecutorService =
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 booleanisPrime(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:



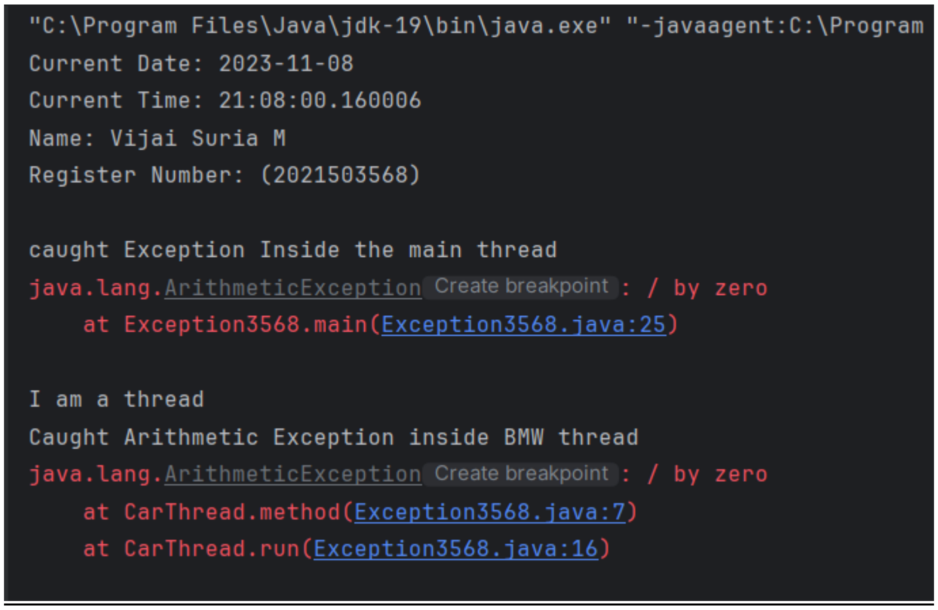
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Pr
Current Date: 2023-11-08
Current Time: 21:15:03.061519900
Name: Vijai Suria M
Register Number: (2021503568)

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:



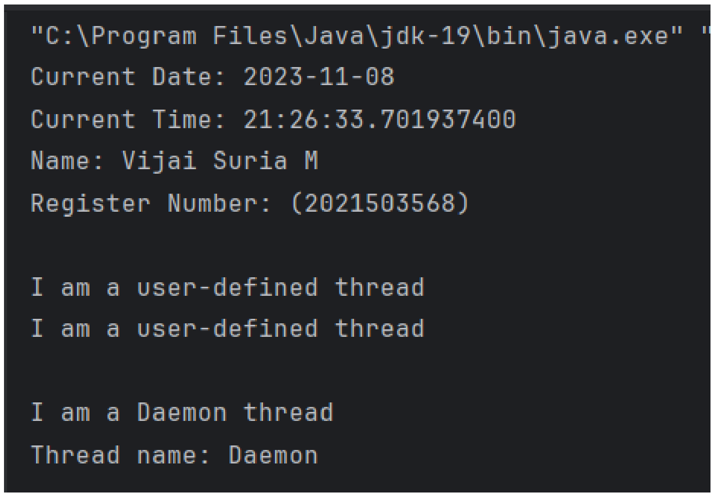
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program
Current Date: 2023-11-08
Current Time: 21:08:00.160006
Name: Vijai Suria M
Register Number: (2021503568)

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:



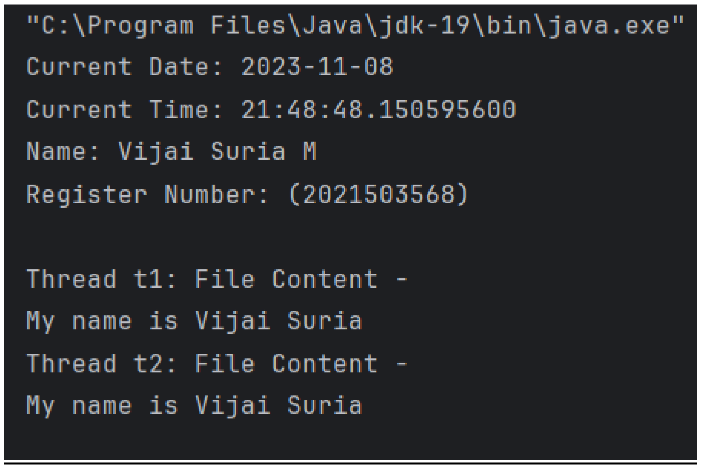
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "
Current Date: 2023-11-08
Current Time: 21:26:33.701937400
Name: Vijai Suria M
Register Number: (2021503568)

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:



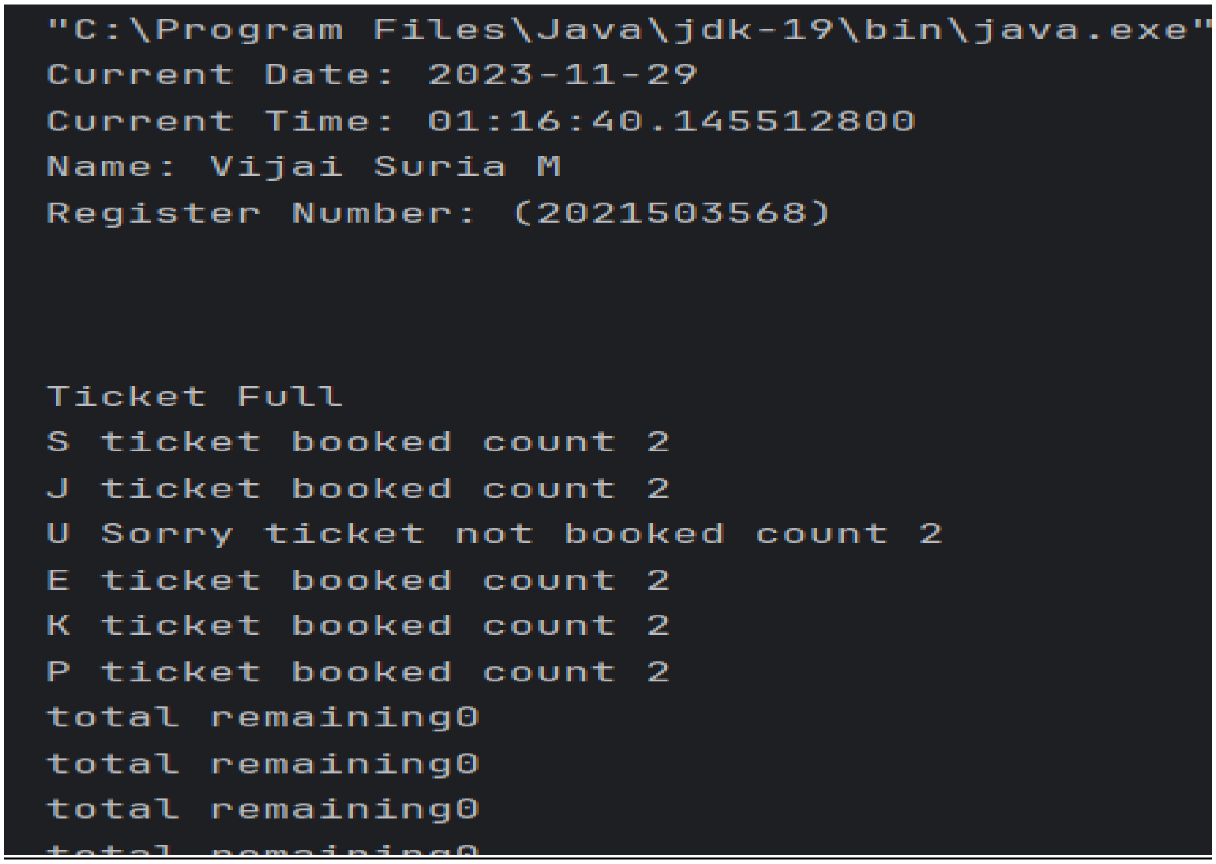
```
"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-08
Current Time: 21:48:48.150595600
Name: Vijai Suria M
Register Number: (2021503568)

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:



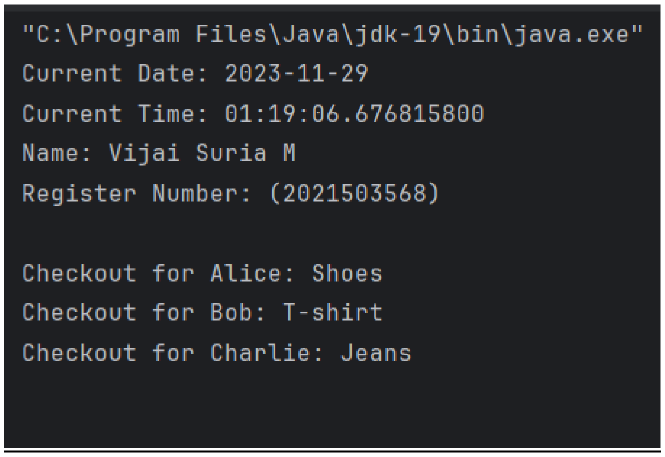
```
"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:16:40.145512800
Name: Vijai Suria M
Register Number: (2021503568)



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:



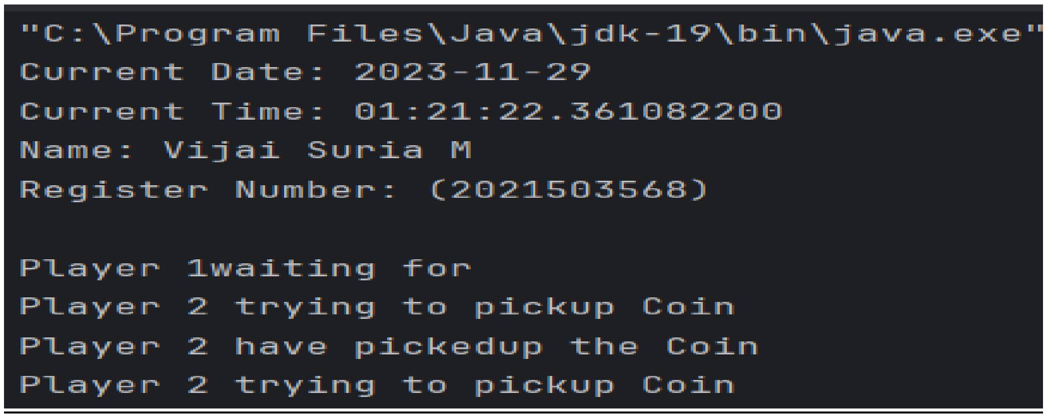
```
"C:\Program Files\Java\jdk-19\bin\java.exe"
Current Date: 2023-11-29
Current Time: 01:19:06.676815800
Name: Vijai Suria M
Register Number: (2021503568)

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
```

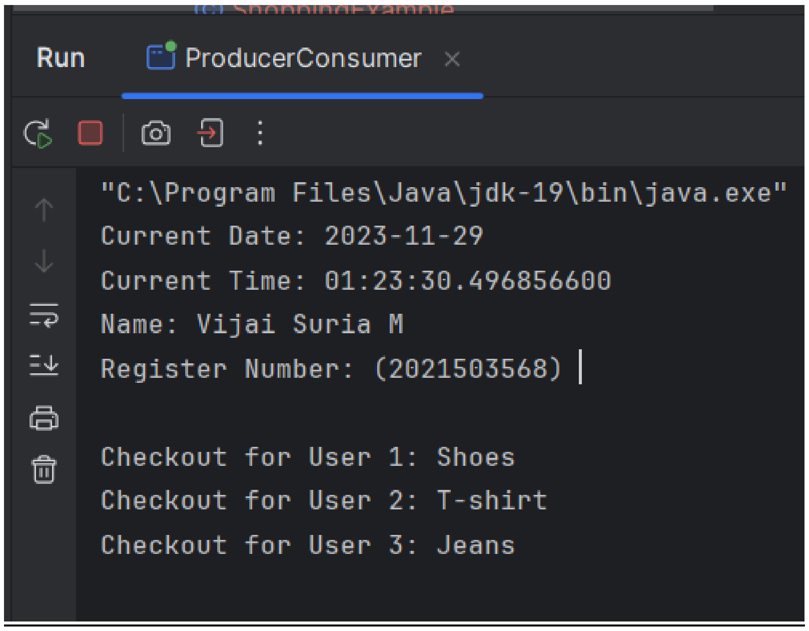
10.11 To illustrate producer consumer problem in java programming

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
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: " + LocalTime.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:



```
"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
```

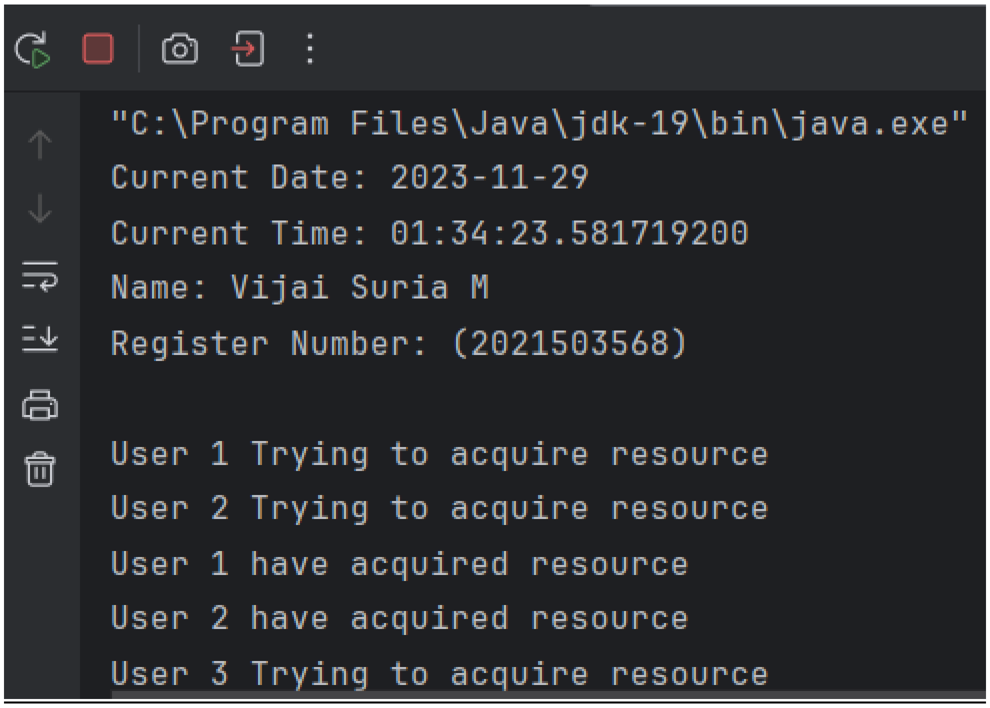
10.12 To make use of semaphore to control synchronization in java

CODE:

```
import java.time.LocalDate;
import java.time.LocalTime;
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: " + LocalTime.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:



```
"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.

Ex.No: 11 **DATE: 22-11-23**

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):

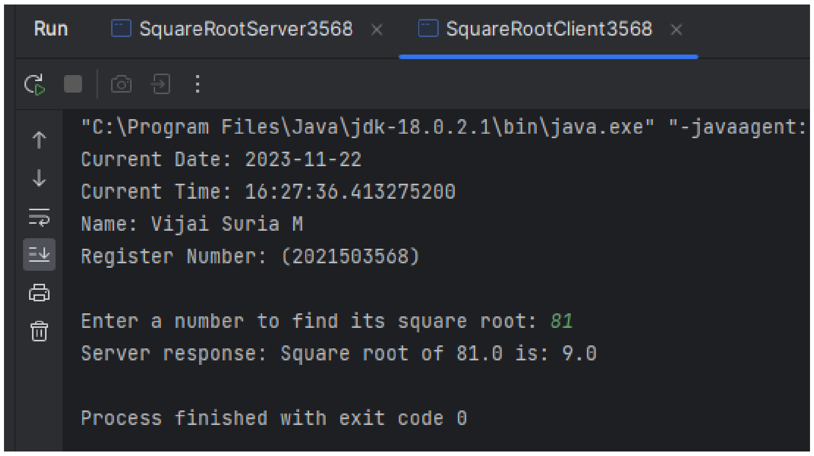
```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class SquareRootServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.now());
        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();
        }
    }
}
```

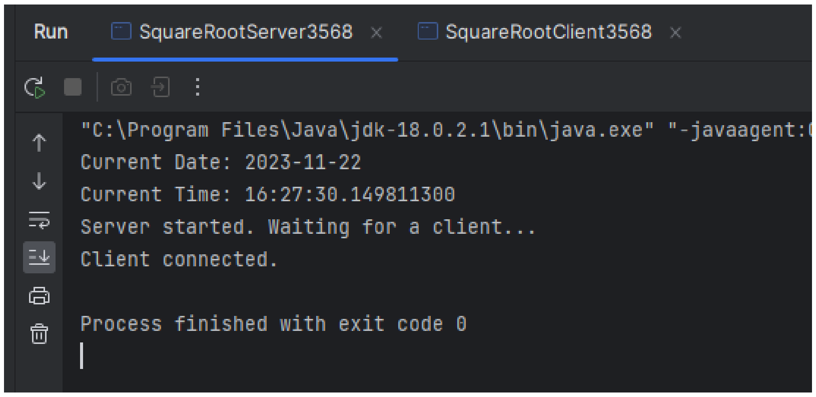
OUTPUT (finding square root):

TCP Client:



```
"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:



```
"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):

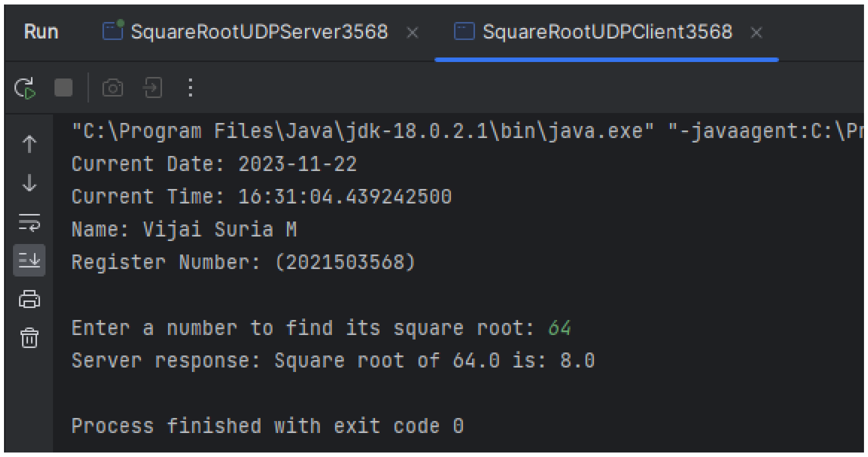
```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;

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 SquareRootUDPClient3568 {
   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();
      }
   }
}
```

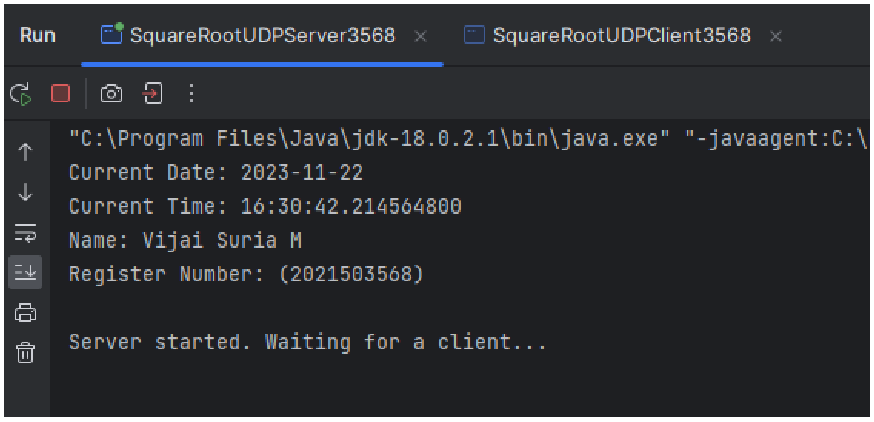
OUTPUT (finding square root):

UDP Client:



```
"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:



```
"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):

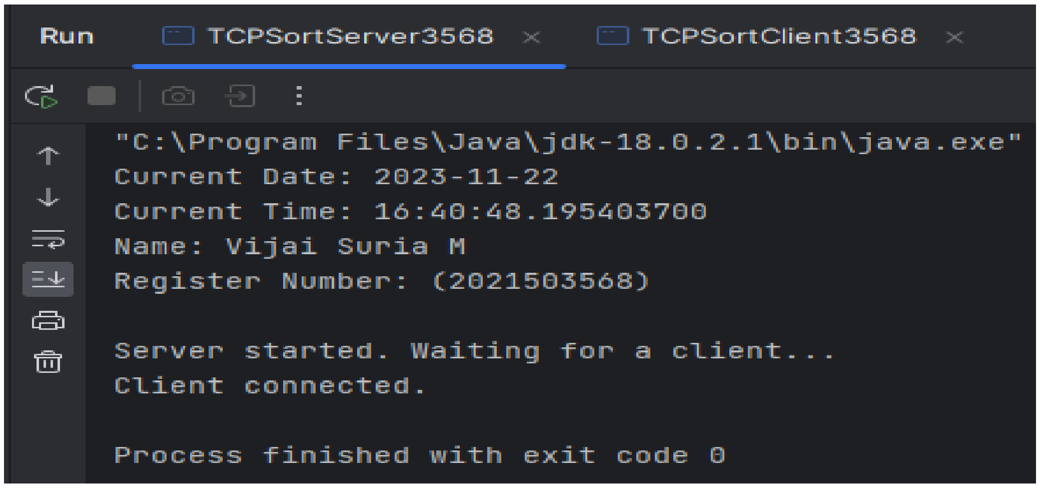
```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Arrays;
public class TCPSortServer3568 {
    public static void main(String[] args) {
        System.out.println("Current Date: " + LocalDate.now());
        System.out.println("Current Time: " + LocalTime.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();
        }
    }}
```

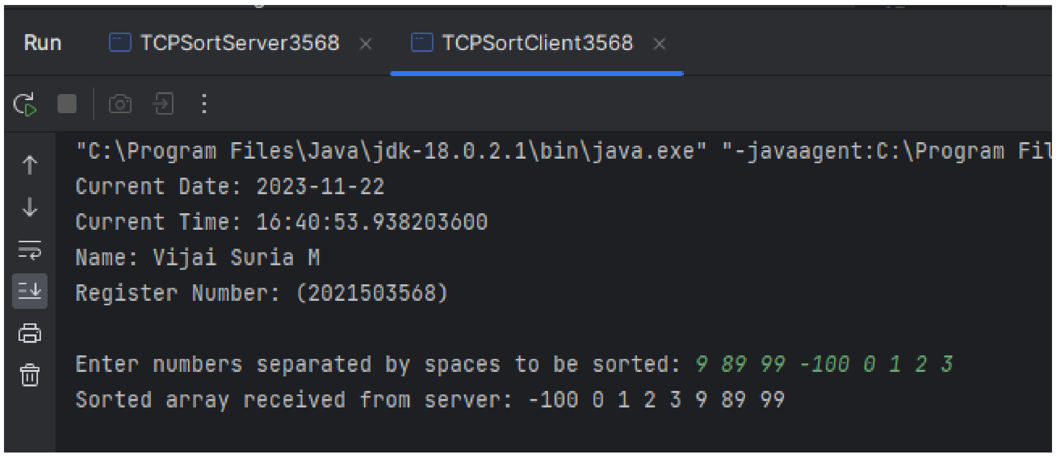
OUTPUT (**sorting array of inputs*****):***

TCP Server:



```
"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:



```
"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**):

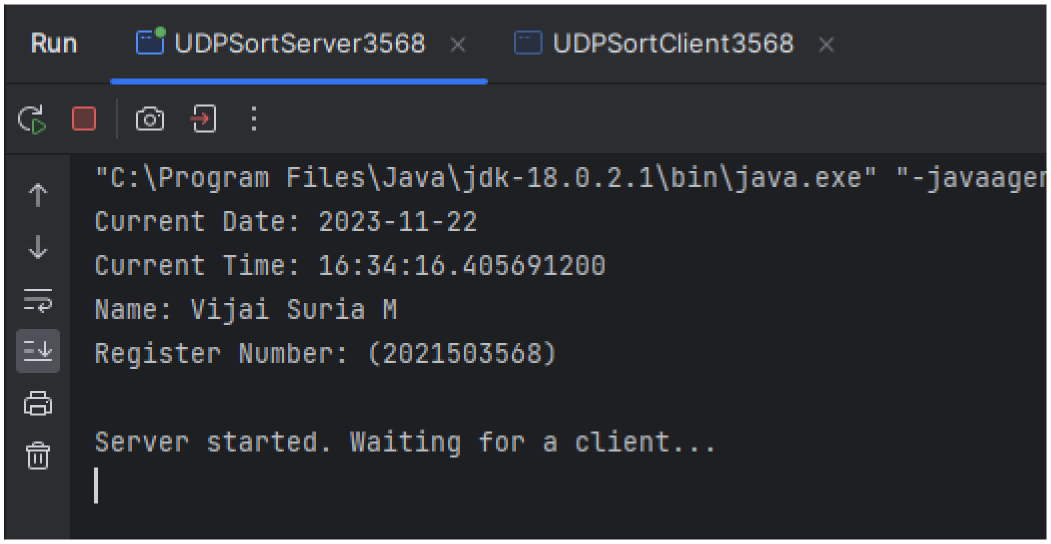
```
import java.io.*;
import java.net.*;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Arrays;
import java.time.LocalDate;
import java.time.LocalTime;

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****):***

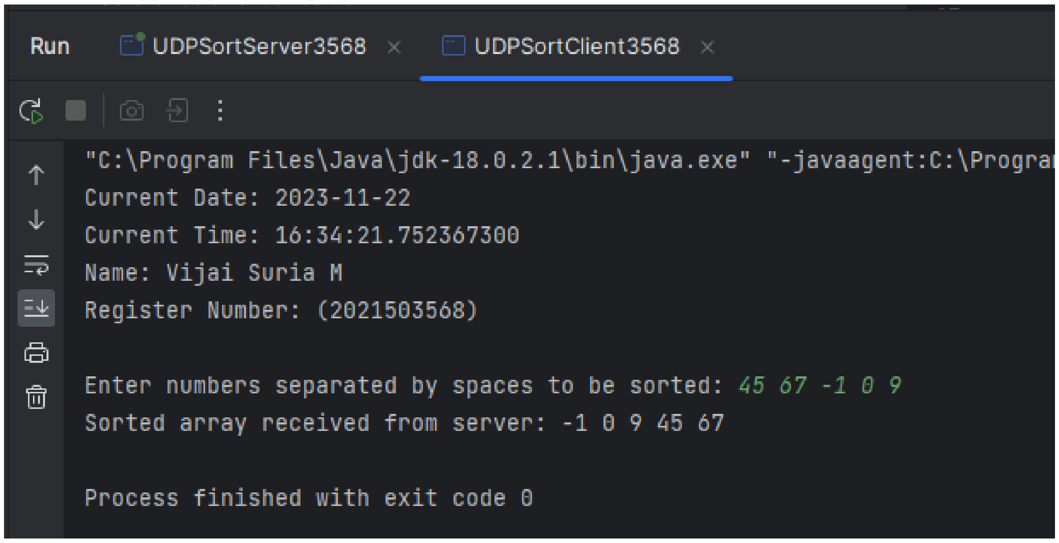
```
import java.io.*;
import java.net.*;
import java.util.Arrays;
import java.time.LocalDate;
import java.time.LocalTime;

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 (**<u>*sorting array of inputs*</u>**):
UDP Server:



```
"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:



```
"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

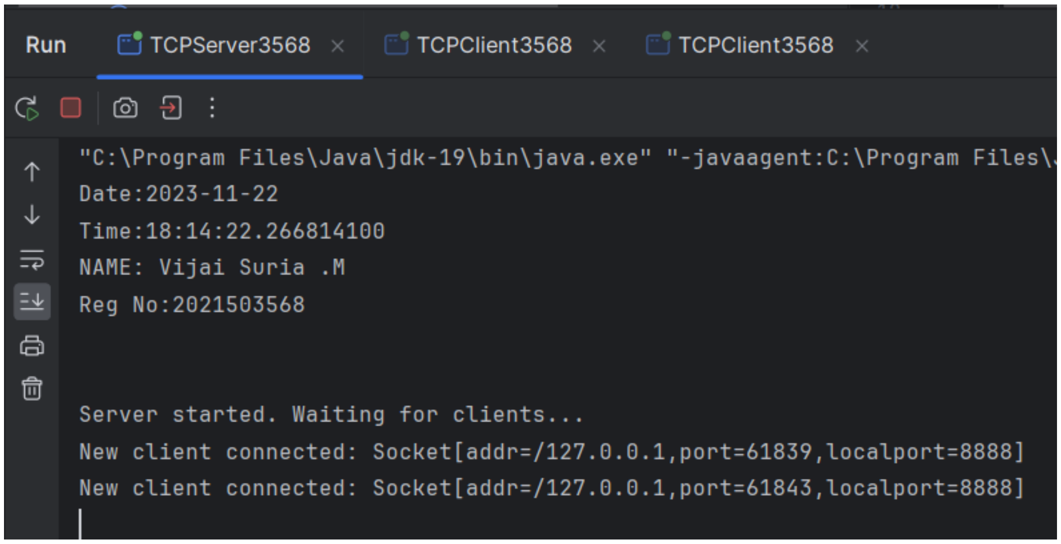
TCP Multi-client chat Server:

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

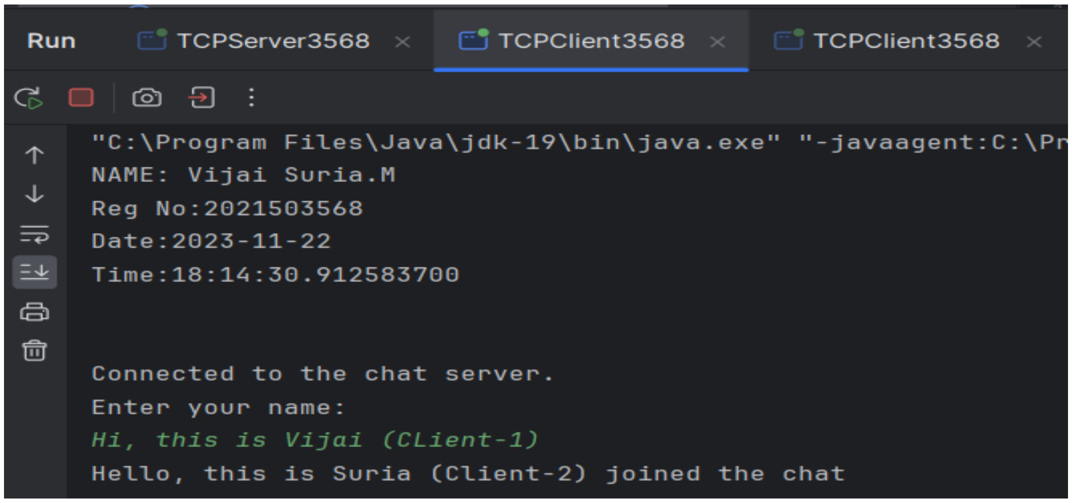
```
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalTime;
public class TCPClient3568 {
    private static final String SERVER_IP = "127.0.0.1"; // Change this to the server's IP
address
    private static final int PORT = 8888;
    public static 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:



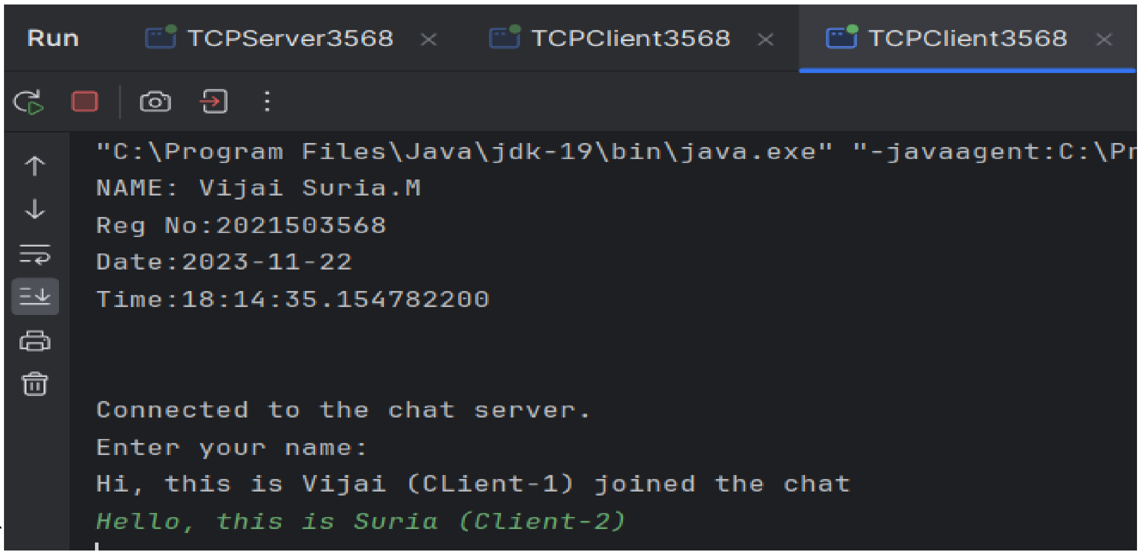
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\
Date:2023-11-22
Time:18:14:22.266814100
NAME: Vijai Suria .M
Reg No:2021503568


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



```
"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



```
"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);
```

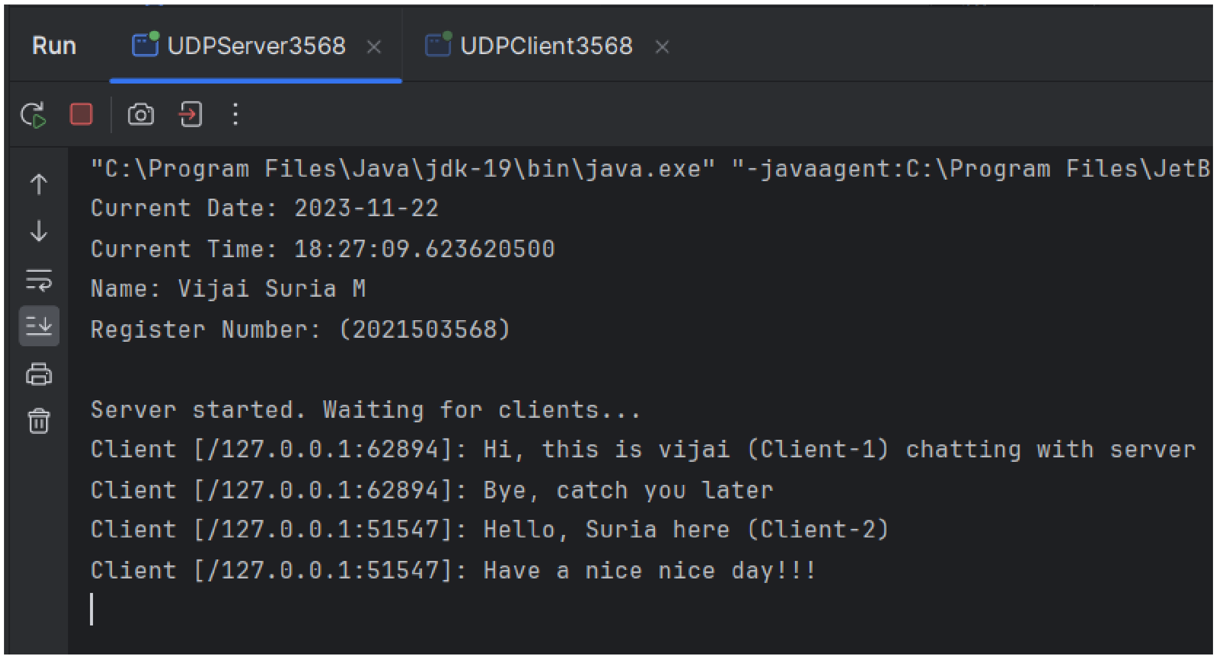
```
            // Sending received message back to the sender client
            sendData(serverSocket, clientAddress, clientPort,
receivedMessage.getBytes());
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
   private static void sendData(DatagramSocket socket,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

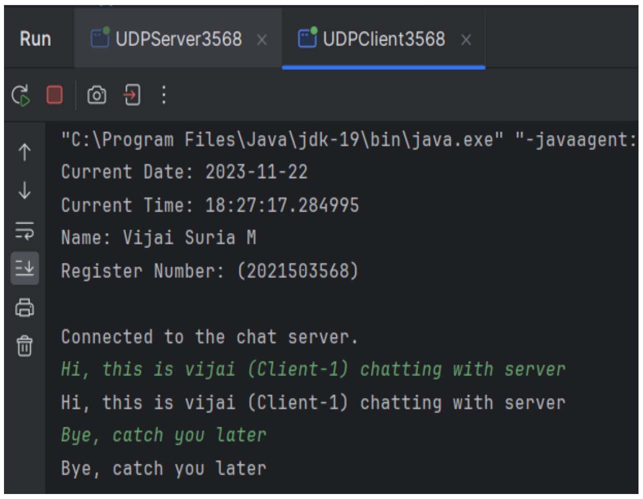
***OUTPUT** (Multi-client chat)*

UDP Multi-chat Server:

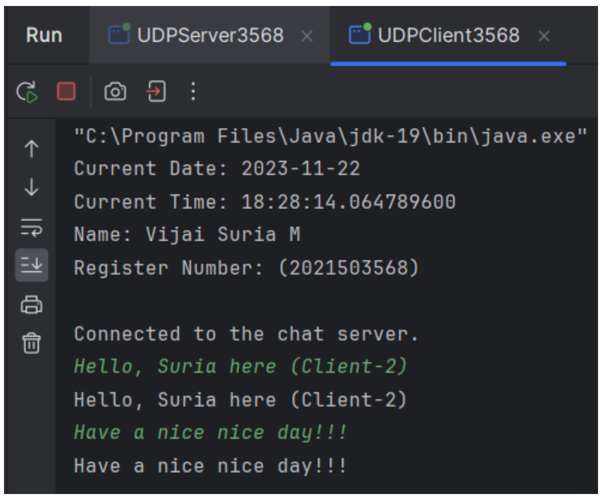


```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetB
Current Date: 2023-11-22
Current Time: 18:27:09.623620500
Name: Vijai Suria M
Register Number: (2021503568)

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:



```
"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
```



```
"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.

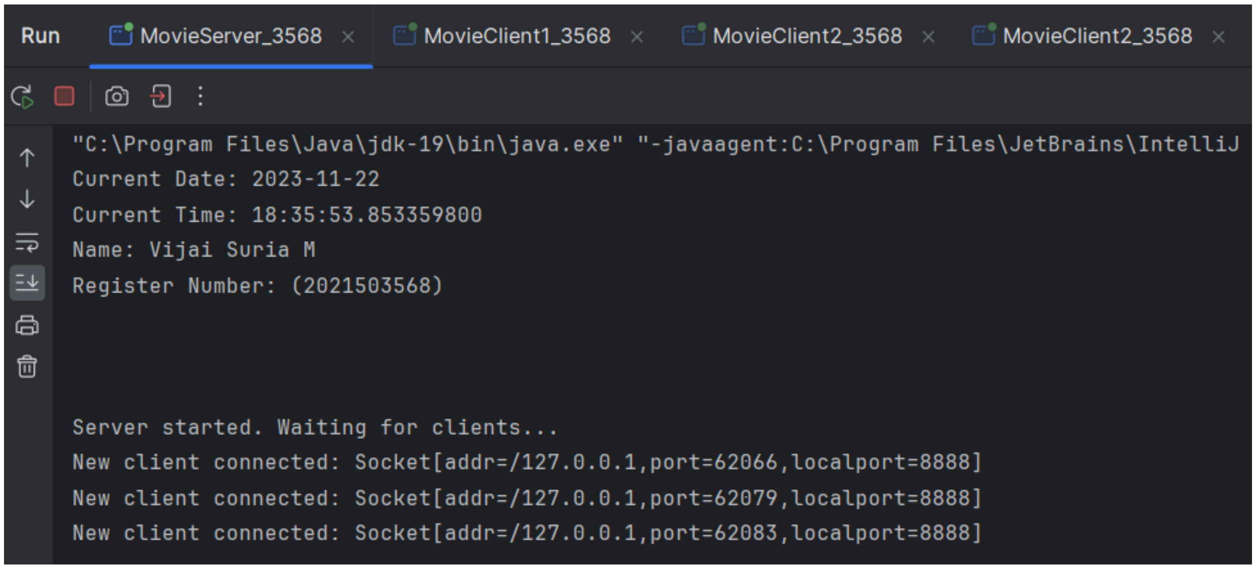
SERVER CODE (for movie ticket booking):

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.time.LocalDate;
import java.time.LocalTime;
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: " + 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 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.");
                }
            }
        }
    }
}
```

<u>CLIENT CODE (</u> *<u>for movie ticket booking</u>* ***<u>):</u>***

```
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();
        }
    }
}
```

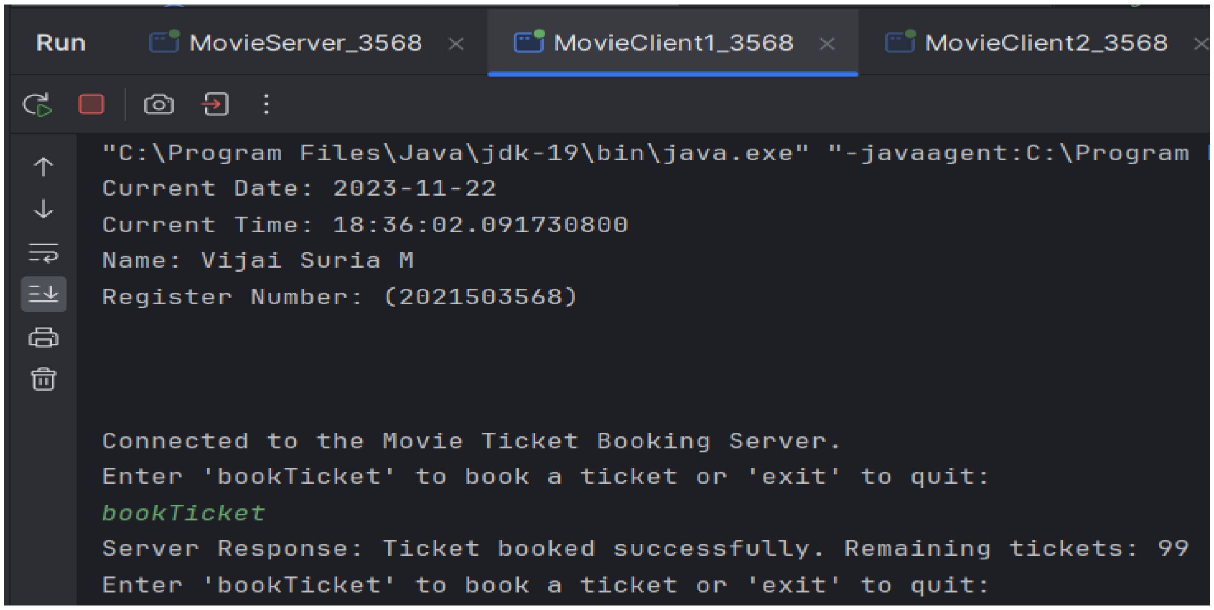
<u>Server (*for movie ticket booking*):</u>



Run | MovieServer_3568 | MovieClient1_3568 | MovieClient2_3568 | MovieClient2_3568

```
"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]
```

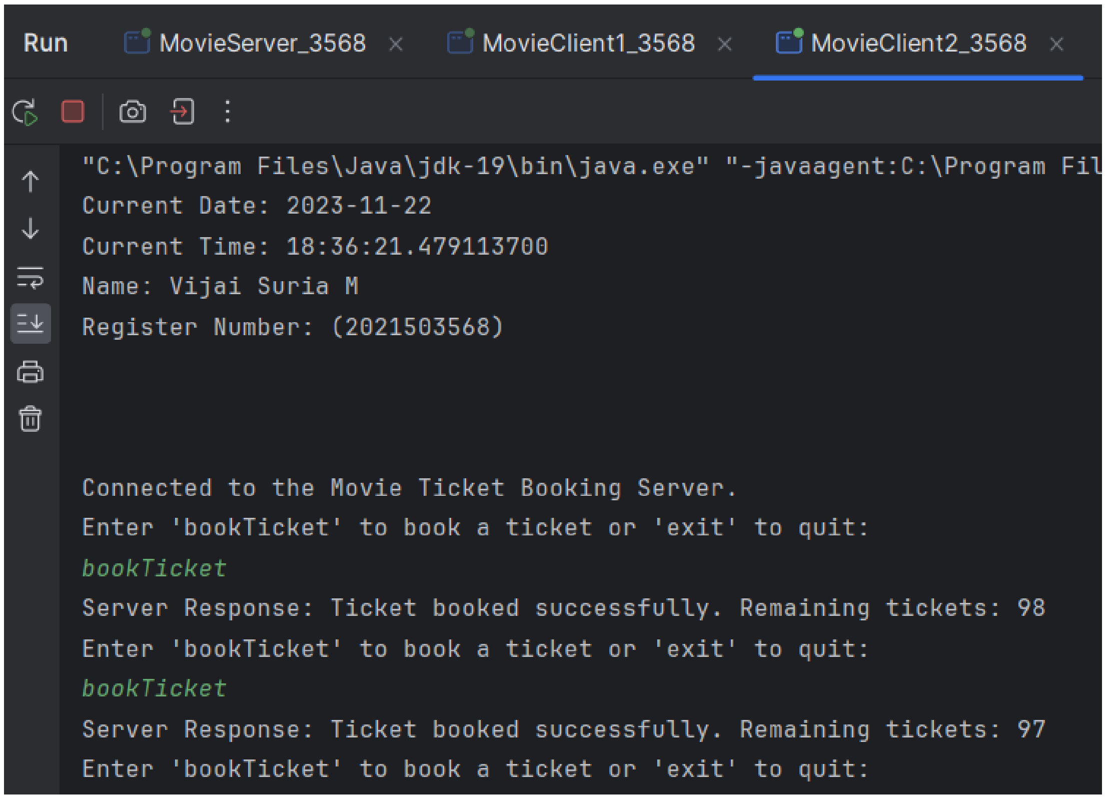
<u>Clients (*for movie ticket booking*):-</u>



Run | MovieServer_3568 | MovieClient1_3568 | MovieClient2_3568

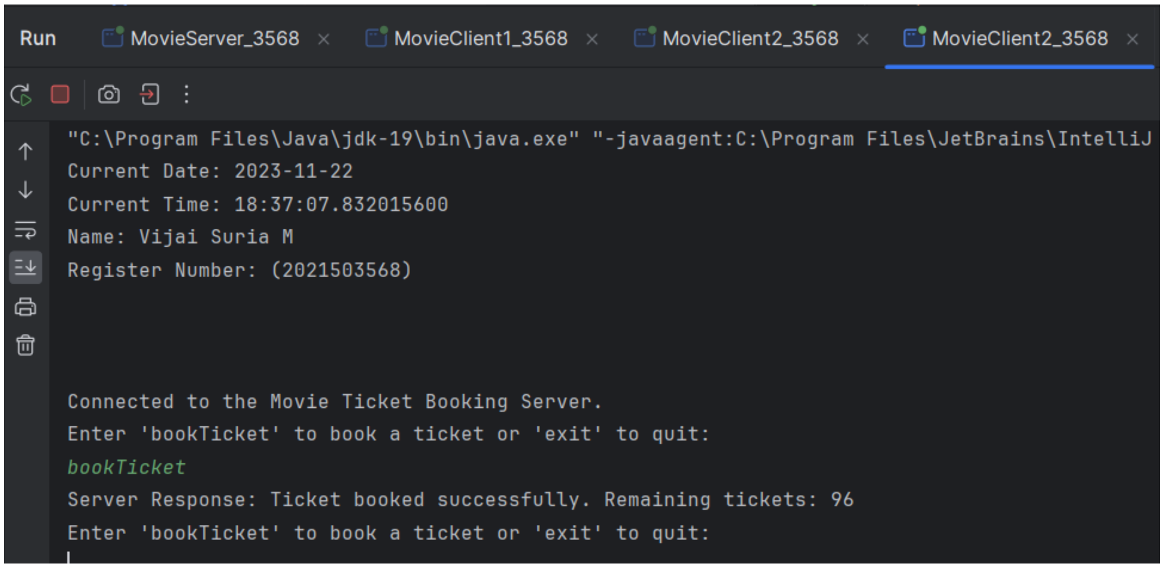
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program
Current Date: 2023-11-22
Current Time: 18:36:02.091730800
Name: Vijai Suria M
Register Number: (2021503568)



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):-



```
"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):-



```
"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.

Ex.No: 12 **DATE: 14-10-23**

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

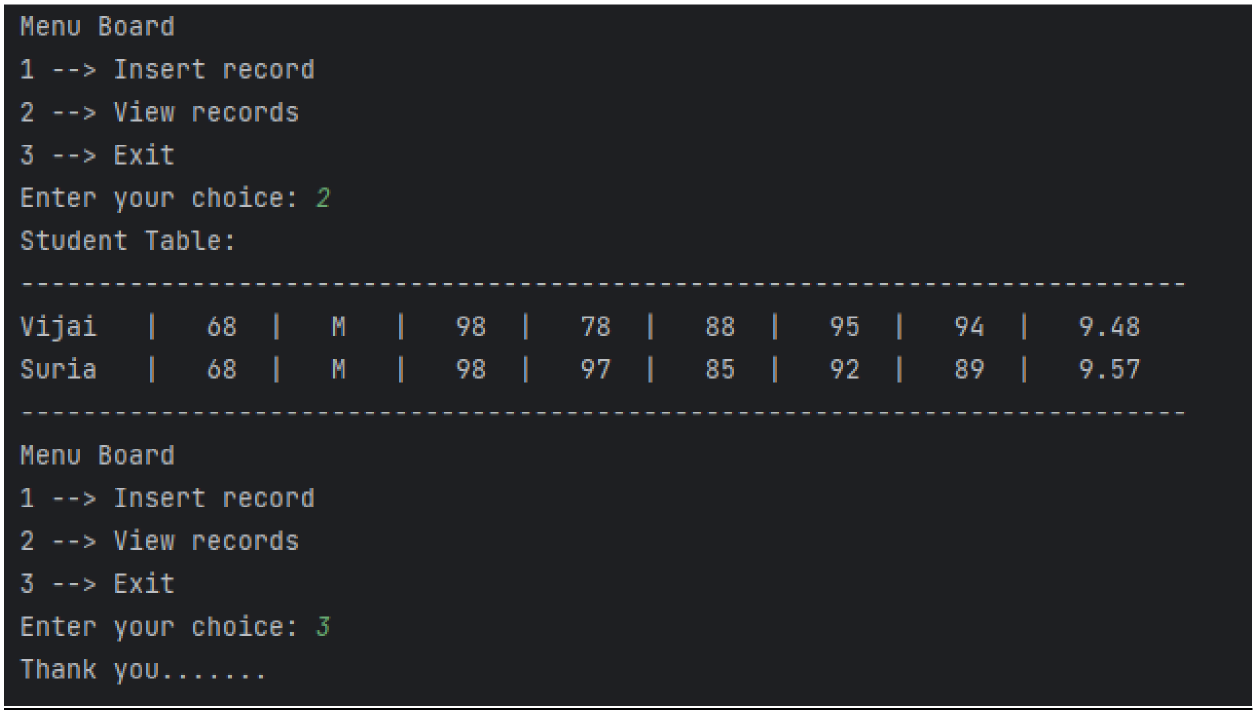
    - Handles server-client communication over sockets, reading input, and writing output.
12. ExecutorService, Executors
    - Manages a thread pool for handling multiple client connections concurrently.
13. Connection, DriverManager, PreparedStatement, SQLException
    - Manages database connections, executes 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):

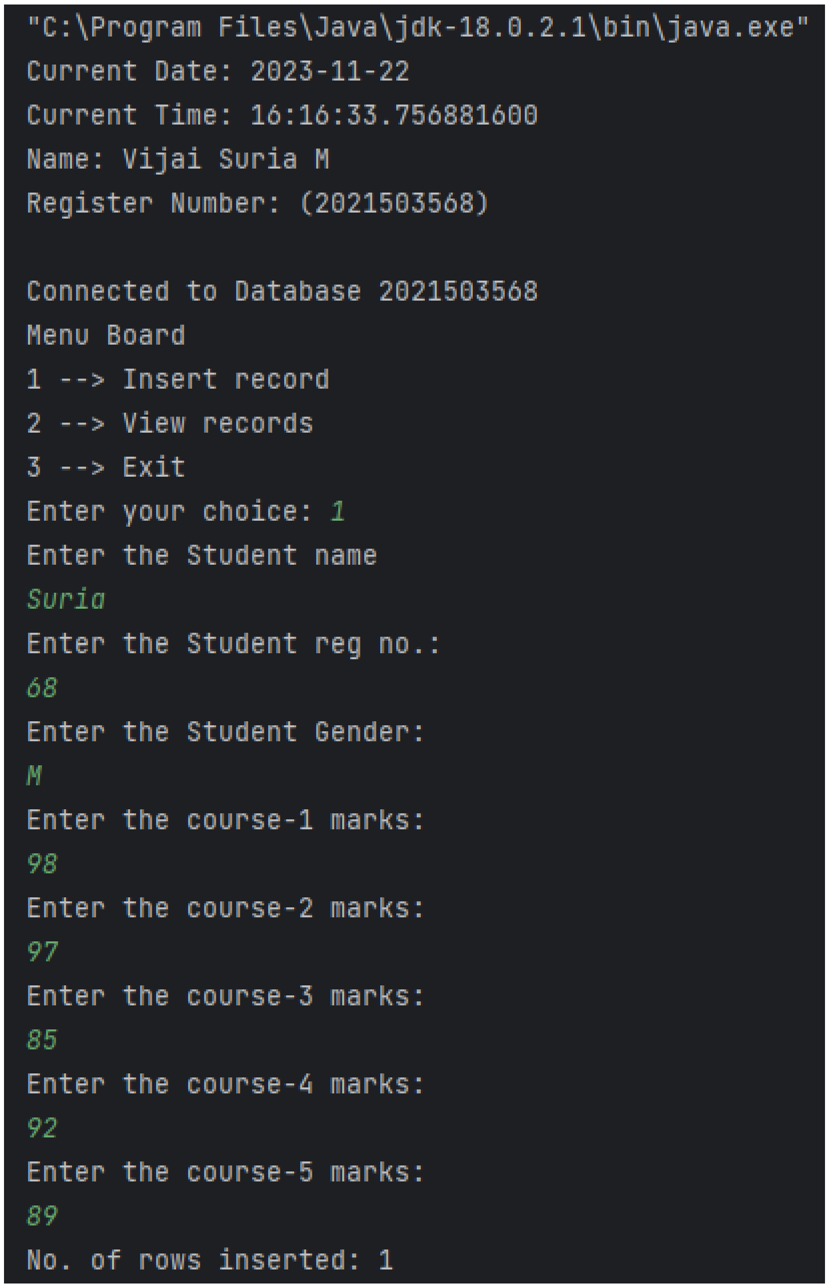
CODE:

```
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.

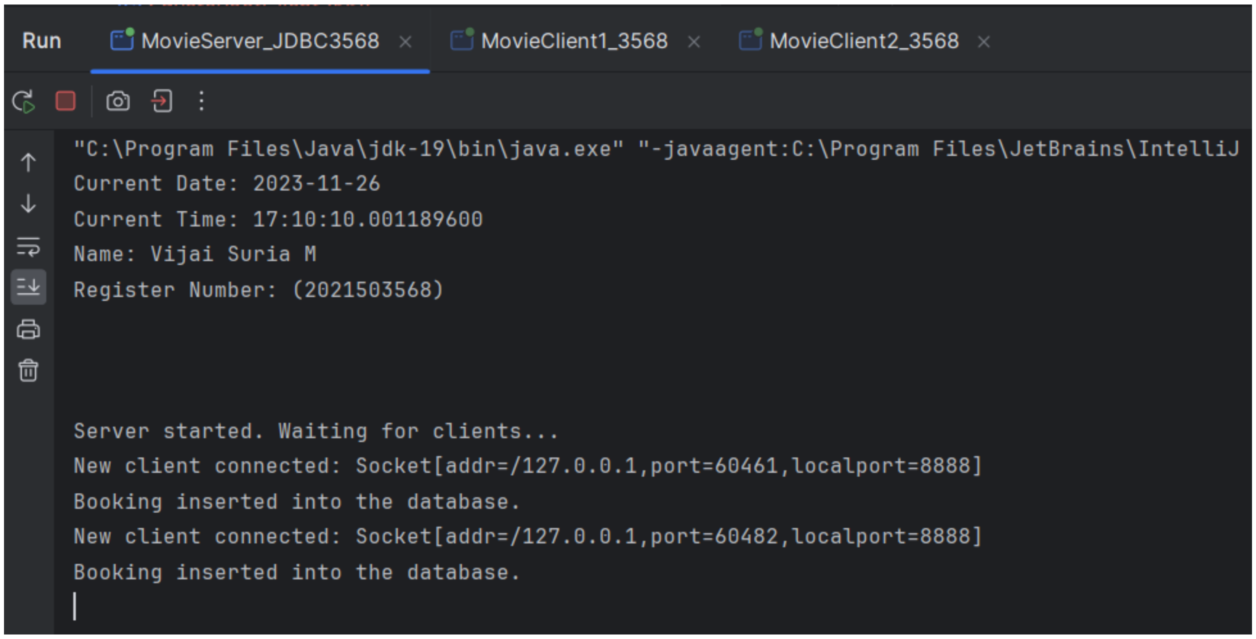
CODE:

```
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.");
            }
        }
    }
}
}
```

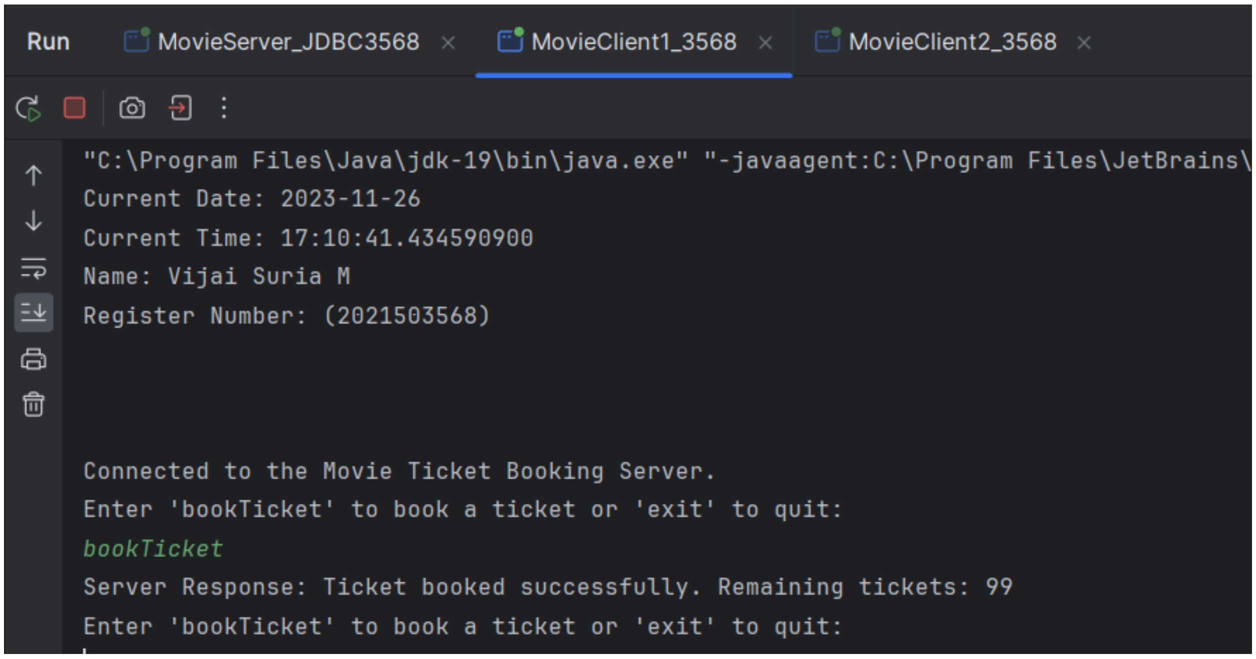
OUTPUT:

Movie Database Server: -



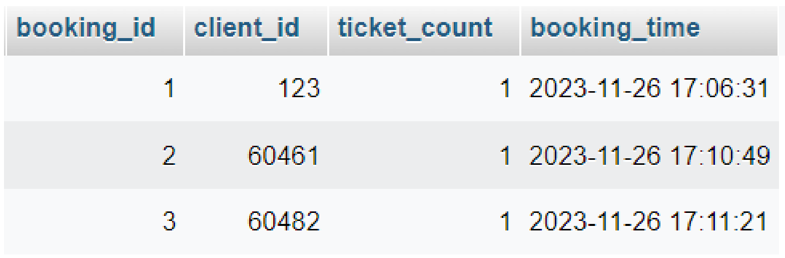
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Current Date: 2023-11-26
Current Time: 17:10:10.001189600
Name: Vijai Suria M
Register Number: (2021503568)



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: -



```
"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.

Ex.No: 13 **DATE: 14-10-23**

COLLECTIONS AND GENERICS

Aim: To explore about collections and genrics 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.

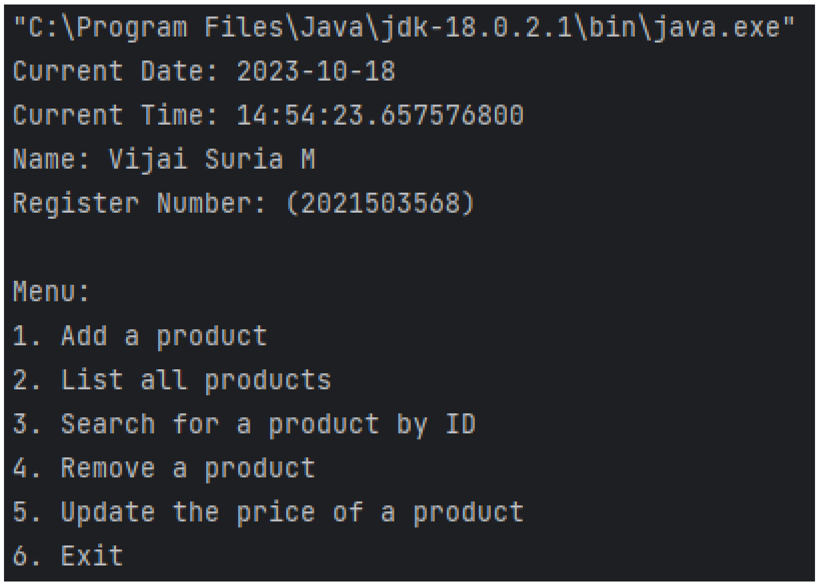
13.1) Write a Java program to implement a Product class with id, name , and price attributes. Use an ArrayList to store Product objects. Prompt the user to add products (id, name, price) and provide a menu to list all products, search by ID, remove a product, and update its price.

CODE:

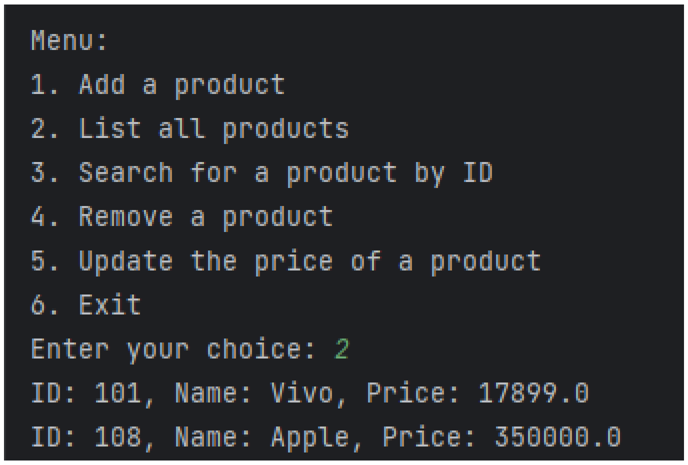
```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

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
```



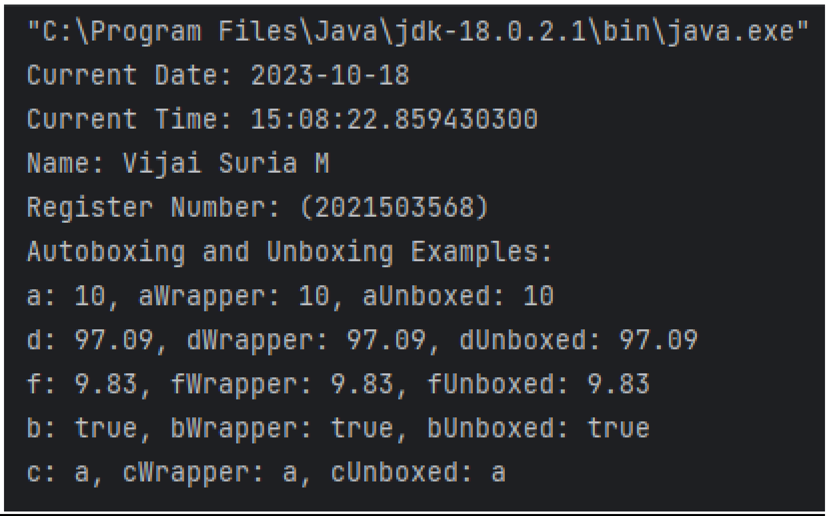
```
Menu:
1. Add a product
2. List all products
3. Search for a product by ID
4. Remove a product
5. Update the price of a product
6. Exit
Enter your choice: 2
ID: 101, Name: Vivo, Price: 17899.0
ID: 108, Name: Apple, Price: 350000.0
```

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

CODE:

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

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)
booleanbUnboxed = 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
```

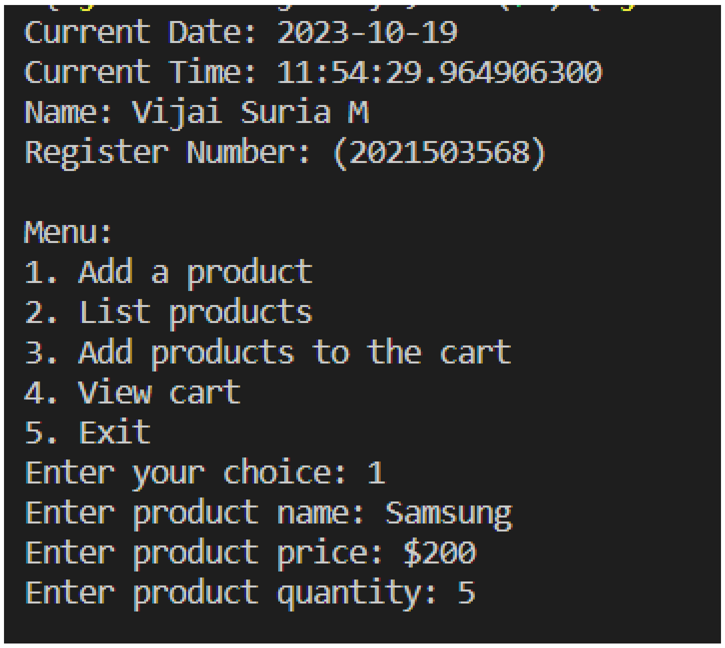
13.3) Implement a Java program featuring Product, AmazonStore, and Main classes to manage an Amazon store, handling customer orders with dedicated functionalities and serving as the program's entry point.

CODE:

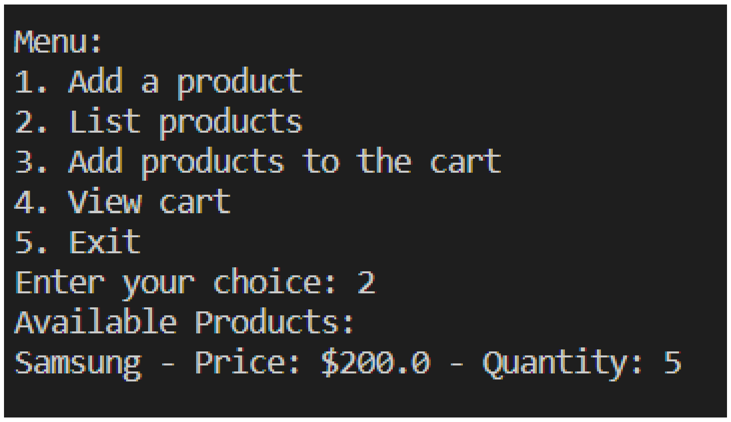
```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.*;
class ProductAlreadyExistsException extends Exception {
   public ProductAlreadyExistsException(String message) {
      super(message);
   }
}
class ProductNotFoundException extends Exception {
   public ProductNotFoundException(String 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)");1

      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.

Ex.No: 14 **DATE: 27-10-23**

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 vistors 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

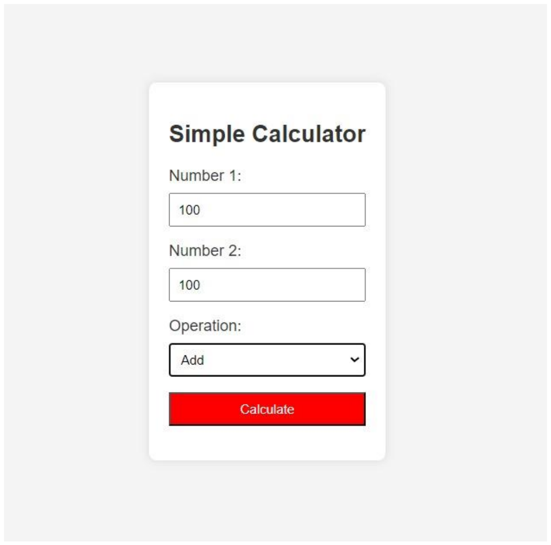
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Simple Calculator</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         background-color: #f4f4f4;
         margin: 0;
         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;
      }
```

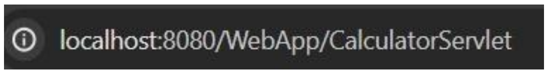
```
        input, select {
            width: 100%;
            padding: 8px;
            margin-bottom: 16px;
            box-sizing: border-box;
        }

        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>
```

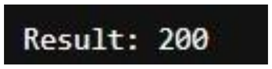
CalculatorServlet.java

```
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/CalculatorServlet


```
Result: 200
```

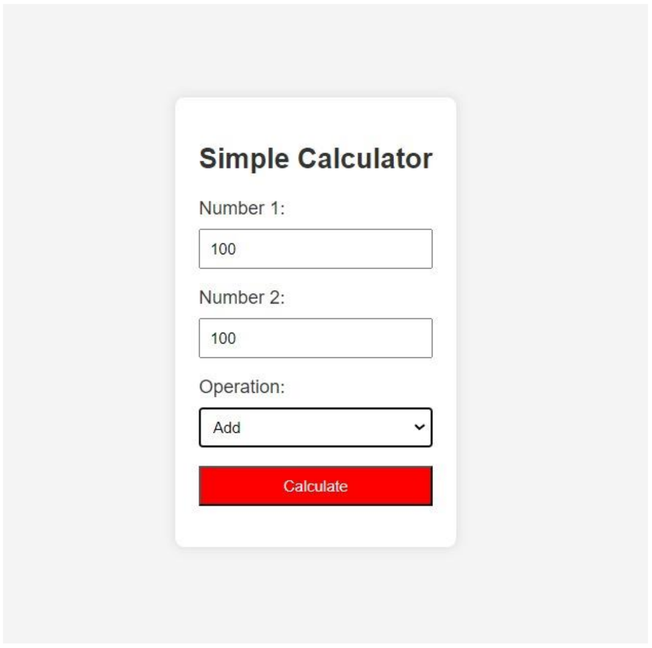
JSP:

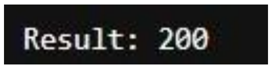
index.jsp

```
<%@ 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:





```
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
```

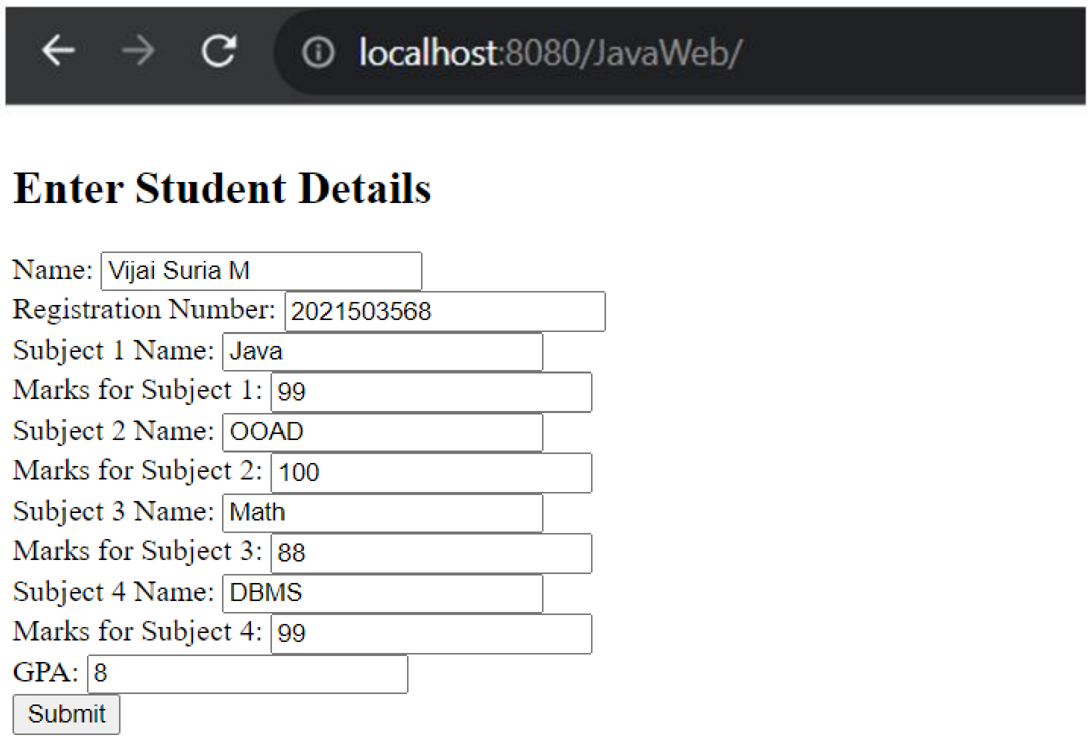
```
IOException, ServletException {
        final String JDBC_URL = "jdbc:oracle:thin:@localhost:XE";
        final String DB_USERNAME = "system";
        final String DB_PASSWORD = "kiran";
        // Retrieving parameters from the form
        String name = request.getParameter("name");
        int 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>
```

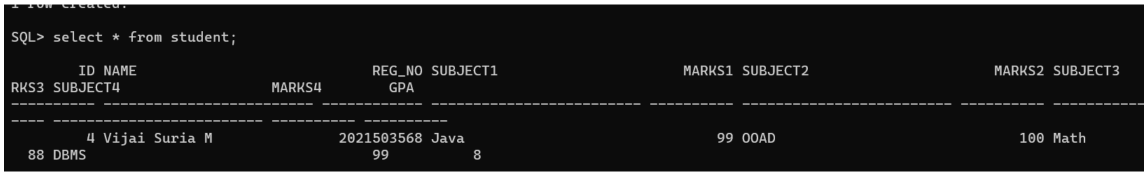
OUTPUT:



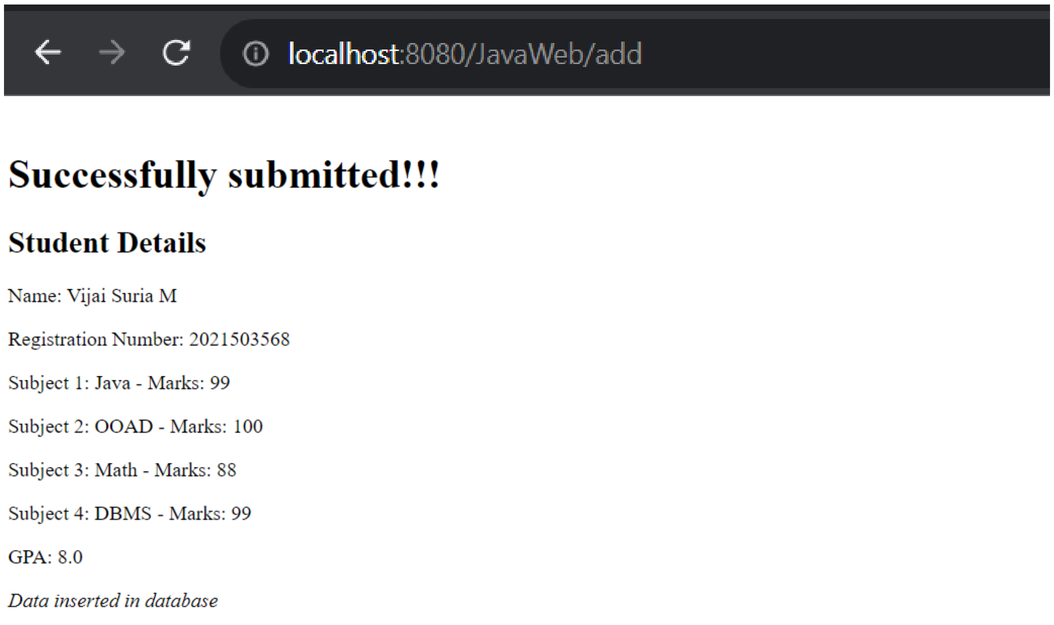
localhost:8080/JavaWeb/

Enter Student Details

Name:	Vijai Suria M
Registration Number:	2021503568
Subject 1 Name:	Java
Marks for Subject 1:	99
Subject 2 Name:	OOAD
Marks for Subject 2:	100
Subject 3 Name:	Math
Marks for Subject 3:	88
Subject 4 Name:	DBMS
Marks for Subject 4:	99
GPA:	8



```
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
```



localhost:8080/JavaWeb/add

Successfully submitted!!!

Student Details

Name: Vijai Suria M

Registration Number: 2021503568

Subject 1: Java - Marks: 99

Subject 2: OOAD - Marks: 100

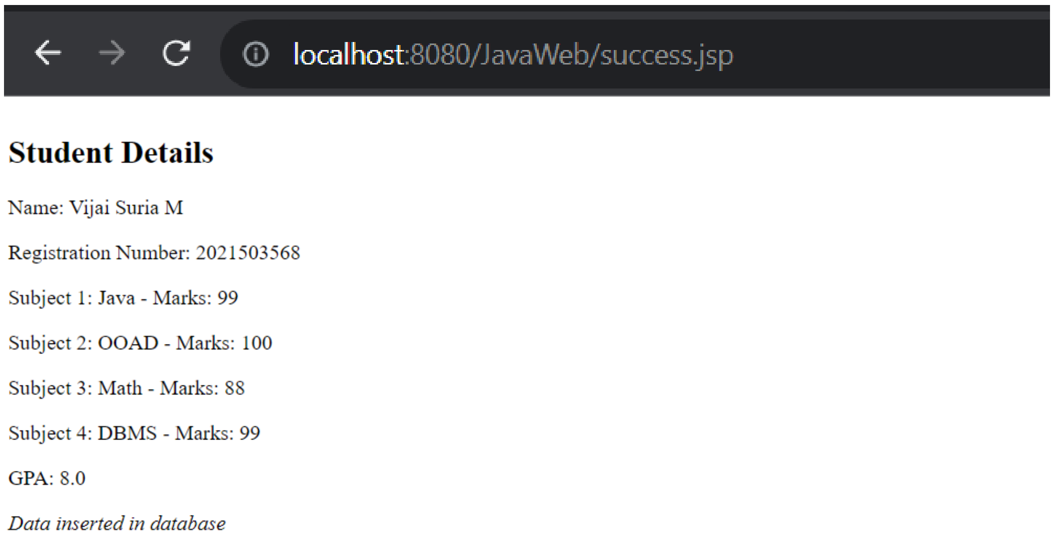
Subject 3: Math - Marks: 88

Subject 4: DBMS - Marks: 99

GPA: 8.0

Data inserted in database

OUTPUT: JSP



localhost:8080/JavaWeb/success.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">
```

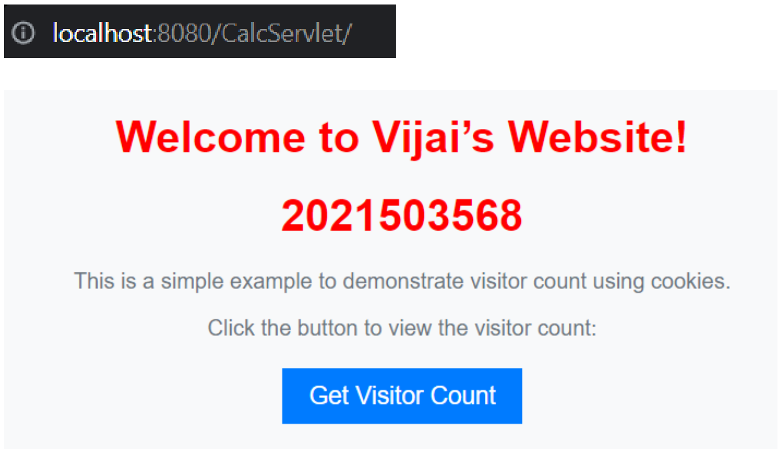
```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Visitor Count Example</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f8f9fa;
            margin: 20px;
            padding: 20px;
            text-align: 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:



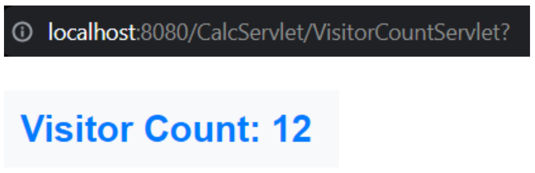
localhost:8080/CalcServlet/

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:



localhost:8080/CalcServlet/VisitorCountServlet?

Visitor Count: 12

JSP:

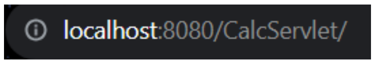
index.jsp

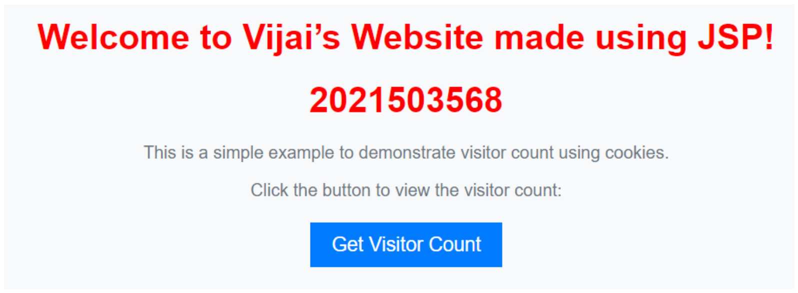
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Visitor Count Example</title>
  <style>
    body {
      font-family: '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:

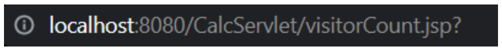

localhost:8080/CalcServlet/

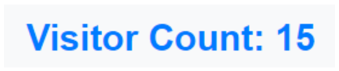

Welcome to Vijai’s Website made using JSP!

2021503568

This is a simple example to demonstrate visitor count using cookies.

Click the button to view the visitor count:


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;
      }
```

```
        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>
```

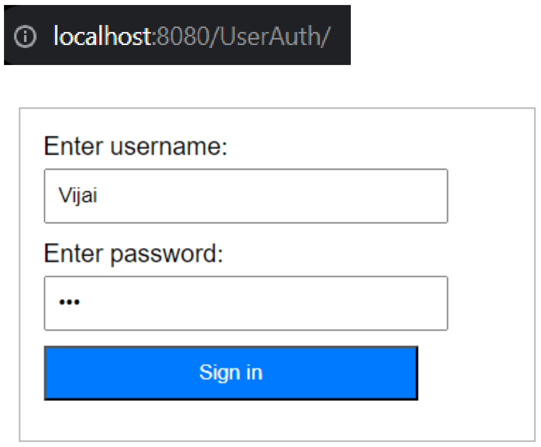
Session.java

```
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")) {
                res.sendRedirect("session");
            } else
            {
                out.println("<font color='red'><b>You have entered incorrect password</b></font>");
                RequestDispatcher rd = req.getRequestDispatcher("index.html");
                rd.include(req, res);
            }
        }catch (Exception e) { System.out.println(e);
        }
    }
}
```

HTTPSession.java

```
package webapp;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet("/session")
public class HTTPSession extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) {
        try {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            HttpSession session = req.getSession();
            String name = (String) session.getAttribute("Name");
            out.println("<!DOCTYPE html>");
            out.println("<html lang=\"en\">");
            out.println("<head>");
            out.println("<meta charset=\"UTF-8\">");
            out.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">");
            out.println("<title>Session Page</title>");
            out.println("<style>");
            out.println("body { font-family: 'Arial', sans-serif; background-color: #f8f9fa; text-align:
center; margin: 20px; }");
            out.println("h2 { color: #007bff; }");
            out.println("</style>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h2>Hello " + name + "</h2>");
            out.println("</body>");
            out.println("</html>");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}:
```

OUTPUT:



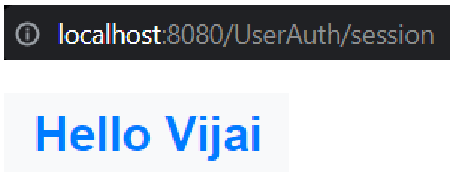
localhost:8080/UserAuth/

Enter username:

Vijai

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>
```

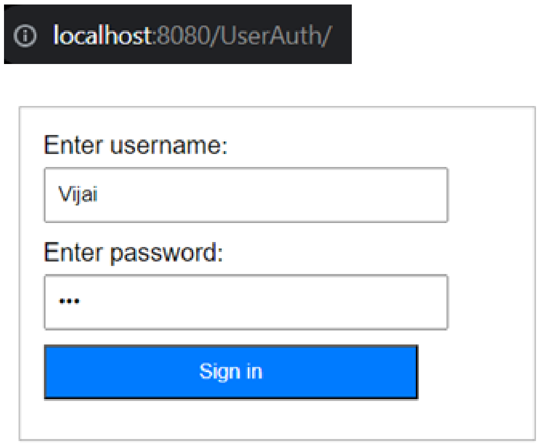
authenticate.jsp

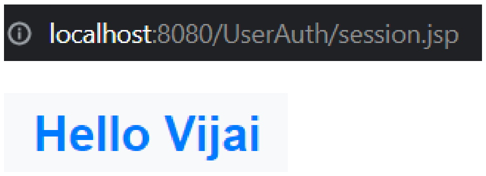
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%
    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;
        }
        h2 {
            color: #007bff;
        }
    </style>
</head>
<body>
    <h2>Hello <%= session.getAttribute("Name") %></h2>
</body>
</html>
```

OUTPUT:





14.5) Write a Servlet and JSP code to redirect to welcome page based on user credentials

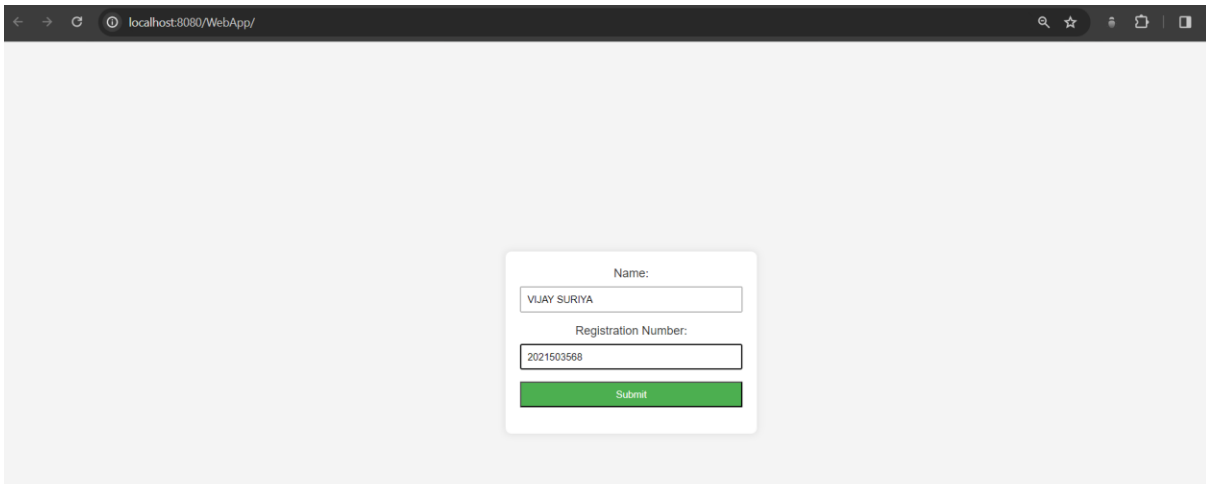
CODE: (SERVLET)

```
</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:

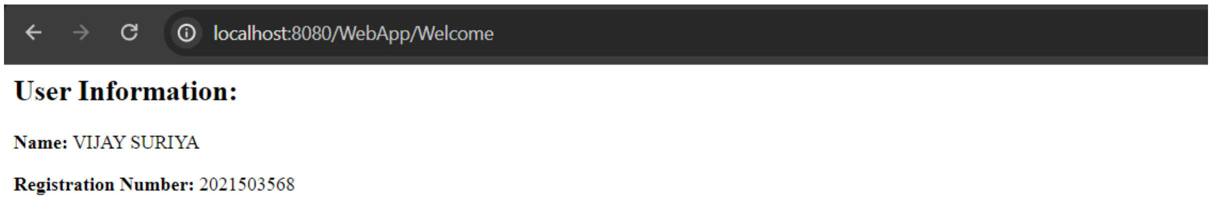
```
import java.io.*;

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:



VIJAY SURIYA

2021503568



User Information:

Name: VIJAY SURIYA

Registration Number: 2021503568

JSP:

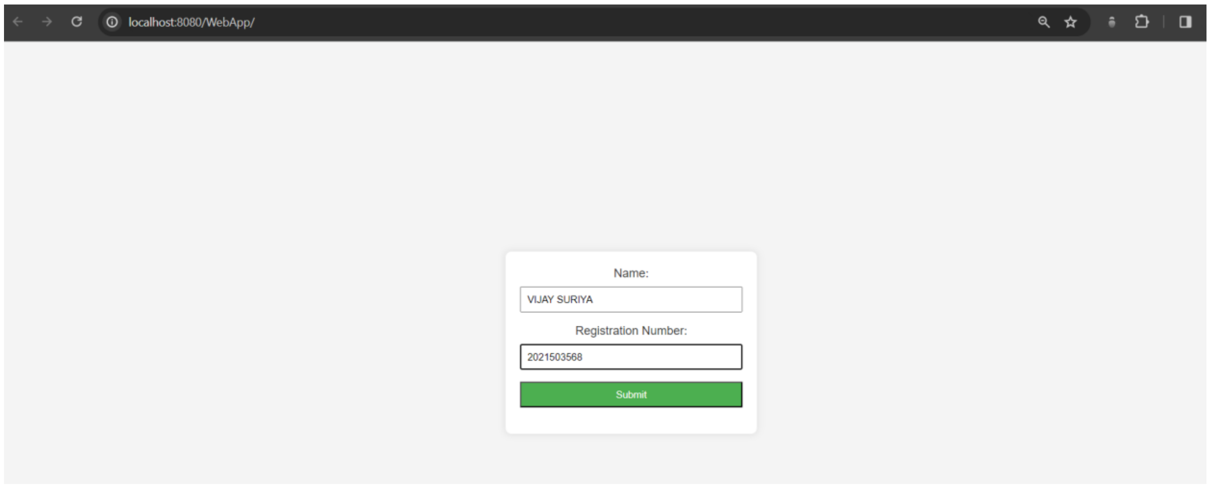
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Information Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            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

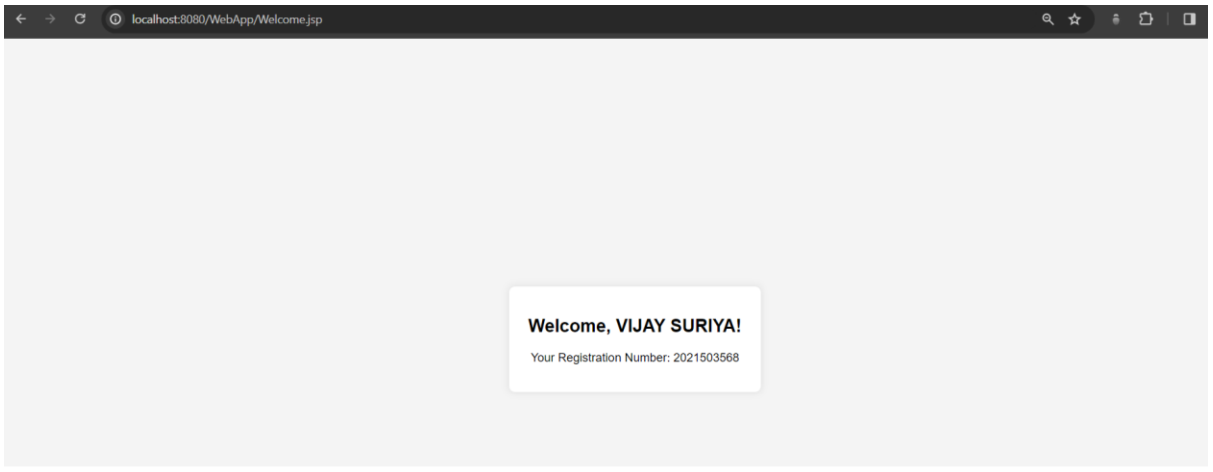
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Welcome Page</title>
   <style>
      body {
         font-family: 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:


localhost:8080/WebApp/

Name: VIJAY SURIYA

Registration Number: 2021503568


localhost:8080/WebApp/Welcome.jsp

Welcome, VIJAY SURIYA!

Your Registration Number: 2021503568

RESULT:

Thus, the Servlets and JSP in java in implemented and executed successfully.