INDEX

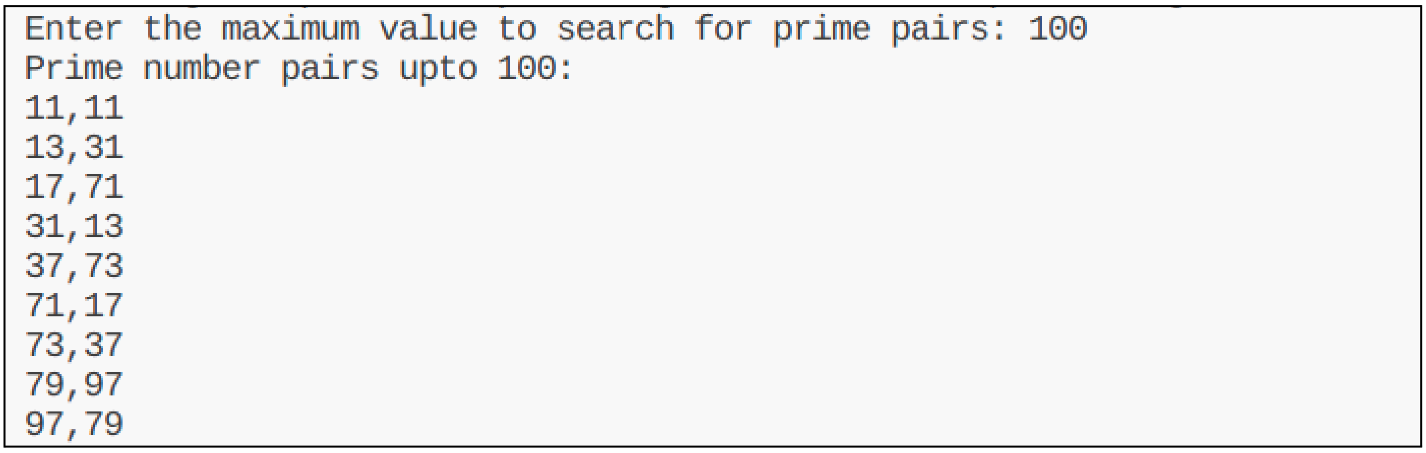
Q1) a) WAP to print prime number in pair like 13,31.

*/** CODE ***/*

```
import java.util.Scanner;

class PrimePairs {
  public boolean isPrime(int n) {
    // Corner case
    if (n <= 1)
      return false;

    // Check from 2 to n/2
    for (int i = 2; i < n/2; i++)
    {
      if (n % i == 0)
        return false;
    }
    return true;
  }

  public int reverse(int num) {
    int rev = 0;
    while(num >0) {
      rev = rev * 10 + num%10;
      num = num/10;
    }
    return rev;
  }

  public void printPrimePairs(int n) {
    for(int i=10;i<=n;i++) {
      if(isPrime(i)) {
        int rev = reverse(i);
        if(isPrime(rev)) {
          System.out.println(i+","+rev);
        }
      }
    }
  }
}
public class P1_a {
  public static void main(String[] args) {
   Scanner sc=new Scanner(System.in);
   System.out.print("Enter the maximum value to search for prime pairs: ");
   int n = sc.nextInt();
```

```java
        PrimePairs x = new PrimePairs();

        if(n<=10) {
          System.out.println("No Prime Pairs found!!!");
        }
        else {
          System.out.println("Prime number pairs upto "+n+":");
          x.printPrimePairs(n);
        }
        sc.close();
      }
    }
```

Output:



```
Enter the maximum value to search for prime pairs: 100
Prime number pairs upto 100:
11,11
13,31
17,71
31,13
37,73
71,17
73,37
79,97
97,79
```

Q1) b) Write a program to validate email address

- **@should be before .**
- **alphabet or digit should be before and after@**
- **alphabet should be before . and 3 alphabets after.**

```
/********************************************   CODE   ******************************************/

import java.util.Scanner;

class EmailValidator {
  public boolean validateEmail(String email) {
    int atIndex = email.indexOf('@');
    int dotIndex = email.lastIndexOf('.');

     // Check if '@' is before '.' and there is at least one character before and after '@'
    if(atIndex > 0 && dotIndex > atIndex + 1 && dotIndex < email.length() - 1) {
      // Check if there are exactly three characters after the last dot
      if(email.length() - dotIndex == 4) {
        return true;
      }
    }
    return false;
  }
}
public class P1_b {
  public static void main(String[] args) {
    EmailValidator x = new EmailValidator();
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter an email: ");
    String email = sc.nextLine();

    boolean isValid = x.validateEmail(email);
    System.out.println(email+ " is Valid: "+ isValid);
    sc.close();
  }
}
```

Output:

```
Enter an email: abcd@gmail.com
abcd@gmail.com is Valid: true
```

```
Enter an email: abcd@gmailcom
abcd@gmailcom is Valid: false
```

Q2) Implementing the concept of Wrapper Classes, Type Casting and auto-boxing and unboxing.

*/** CODE **/*

```
public class P2 {
  public static void main(String args[]) {

     int i = 30;
     float f = 50.0F;
     char c = 'a';
     boolean b = true;

     // Autoboxing: Converting primitives into objects
     Integer intobj = i;
     Float floatobj = f;
     Character charobj = c;
     Boolean boolobj = b;

     // Printing objects
     System.out.println("\n---Printing object values---");
     System.out.println("Integer object: " + intobj);
     System.out.println("Float object: " + floatobj);
     System.out.println("Character object: " + charobj);
     System.out.println("Boolean object: " + boolobj);

     // Unboxing: Converting Objects to Primitives
     int intvalue = intobj.intValue();   //converting Integer to int explicitly
     float floatvalue = floatobj.floatValue();
     char charvalue = charobj;   //unboxing, now compiler will write a.intValue() internally
     boolean boolvalue = boolobj;

     // Printing primitives
     System.out.println("\n---Printing primitive values---");
     System.out.println("int value: " + intvalue);
     System.out.println("float value: " + floatvalue);
     System.out.println("char value: " + charvalue);
     System.out.println("boolean value: " + boolvalue);

     // Type casting example
     double doubleValue = 42.56;
     int intValue = (int) doubleValue; // Explicit casting (narrowing)
     System.out.println("\nType Casting Example:");
     System.out.println("Original double: " + doubleValue);
     System.out.println("Casted int: " + intValue);
  }
}
```

Output:

```
---Printing object values---
Integer object: 30
Float object: 50.0
Character object: a
Boolean object: true

---Printing primitive values---
int value: 30
float value: 50.0
char value: a
boolean value: true

Type Casting Example:
Original double: 42.56
Casted int: 42
```

Q3) Developing java applications on the concept of Arrays-single dimension, multi-dimension, ragged array.

*/*** CODE **/*

```java
class ArraysExample {
  // Function to demonstrate single-dimensional array
  void singleDimensionalArrayExample() {
    System.out.println("Single-Dimensional Array Example:");

    // Declaration and initialization of a single-dimensional array
    int[] numbers = {1, 2, 3, 4, 5};

    // Accessing and displaying elements of the array
    System.out.print("Array Elements: ");
    for (int i = 0; i < numbers.length; i++) {
      System.out.print(numbers[i] + " ");
    }
    System.out.println();
  }
  // Function to demonstrate multi-dimensional array
  void multiDimensionalArrayExample() {
    System.out.println("\nMulti-Dimensional Array Example:");
    // Declaration and initialization of a 2D array
    int[][] matrix = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    // Accessing and displaying elements of the 2D array
    for (int i = 0; i < matrix.length; i++) {
      for (int j = 0; j < matrix[i].length; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}
// Function to demonstrate ragged array
void raggedArrayExample() {
    // Declaration and initialization of a ragged array
    int[][] raggedMatrix = {
            {1, 2, 3},
            {4, 5},
            {6, 7, 8, 9}
    };
    // Accessing and displaying elements of the ragged array
    System.out.println("\nRagged Array Elements:");
    for (int i = 0; i < raggedMatrix.length; i++) {
        for (int j = 0; j < raggedMatrix[i].length; j++) {
            System.out.print(raggedMatrix[i][j] + " ");
        }
        System.out.println();
    }
}
public static void main(String[] args) {
    ArraysExample x = new ArraysExample();
    x.singleDimensionalArrayExample();
    x.multiDimensionalArrayExample();
    x.raggedArrayExample();
}
}
```

Output:

```
Single-Dimensional Array Example:
Array Elements: 1 2 3 4 5

Multi-Dimensional Array Example:
1 2 3
4 5 6
7 8 9

Ragged Array Elements:
1 2 3
4 5
6 7 8 9
```

Q4) Developing java applications working on complex array arithmetic using Comparable and Comparator interfaces.

*/*** CODE **/*

```
import java.util.Arrays;
import java.util.Comparator;

// Complex number class
class Complex implements Comparable<Complex> {
  private double real;
  private double imaginary;

  public Complex(double real, double imaginary) {
     this.real = real;
     this.imaginary = imaginary;
  }

  // Getter methods for real and imaginary parts
  public double getReal() {
     return real;
  }

  public double getImaginary() {
     return imaginary;
  }

  // Method to calculate the magnitude of a complex number
  public double magnitude() {
     return Math.sqrt(real * real + imaginary * imaginary);
  }

  // Implementing Comparable interface based on magnitude
  @Override
  public int compareTo(Complex other) {
     return Double.compare(this.magnitude(), other.magnitude());
  }

  // Override toString for better display of complex numbers
  @Override
  public String toString() {
     return "(" + real + " + " + imaginary + "i)";
  }
}
```

```java
class ComplexExample {
  public static void main(String[] args) {
    // Creating an array of complex numbers
    Complex[] complexArray = {
        new Complex(3, 4),
        new Complex(1, -2),
        new Complex(-2, 6),
        new Complex(5, 0),
        new Complex(-1, -1)
    };

    // Sorting the array using Comparable (based on magnitude)
    Arrays.sort(complexArray);

    // Displaying sorted array
    System.out.println("Sorted Array (Comparable):");
    for (Complex complex : complexArray) {
      System.out.println(complex);
    }

    // Sorting the array using Comparator (based on real part)
    Arrays.sort(complexArray, Comparator.comparingDouble(Complex::getReal));

    // Displaying sorted array
    System.out.println("\nSorted Array (Comparator - Real Part):");
    for (Complex complex : complexArray) {
      System.out.println(complex);
    }
  }
}
```

Output:

```
Sorted Array (Comparable):
(-1.0 + -1.0i)
(1.0 + -2.0i)
(3.0 + 4.0i)
(5.0 + 0.0i)
(-2.0 + 6.0i)

Sorted Array (Comparator - Real Part):
(-2.0 + 6.0i)
(-1.0 + -1.0i)
(1.0 + -2.0i)
(3.0 + 4.0i)
(5.0 + 0.0i)
```

Q5) Implementing the concept of inheritance in Java and various types of inheritance available.

*/** CODE **/*

```
// Base class (parent class)
class Animal {
  void eat() {
     System.out.println("Animal is eating");
  }
}
// Single Inheritance: Dog is a subclass of Animal
class Dog extends Animal {
  void bark() {
     System.out.println("Dog is barking");
  }
}
// Multilevel Inheritance: Poodle is a subclass of Dog
class Poodle extends Dog {
  void groom() {
     System.out.println("Poodle is being groomed");
  }
}
// Multiple Inheritance: Bird is a subclass of Animal and implements the Flyable
// and Swimmable interfaces
interface Flyable {
  void fly();
}
interface Swimmable {
  void swim();
}
class Bird extends Animal implements Flyable, Swimmable {
  @Override
  void eat() {
     System.out.println("Bird is eating");
  }
  @Override
  public void fly() {
     System.out.println("Bird is flying");
  }
  @Override
  public void swim() {
     System.out.println("Bird is swimming");
  }
}
```

```
// Hierarchical Inheritance: Cat and Lion are subclasses of Animal
class Cat extends Animal {
  void meow() {
      System.out.println("Cat is meowing");
  }
}

class Lion extends Animal {
  void roar() {
      System.out.println("Lion is roaring");
  }
}

// Hybrid Inheritance: Parrot is a subclass of Bird and implements the Talkable
// interface
interface Talkable {
  void talk();
}

class Parrot extends Bird implements Talkable {
  @Override
  public void talk() {
      System.out.println("Parrot is talking");
  }
}

public class P5 {
  public static void main(String[] args) {
      // Single Inheritance
      Dog myDog = new Dog();
      myDog.eat();
      myDog.bark();

      // Multilevel Inheritance
      Poodle myPoodle = new Poodle();
      myPoodle.eat();
      myPoodle.bark();
      myPoodle.groom();

      // Multiple Inheritance
      Bird myBird = new Bird();
      myBird.eat();
      myBird.fly();
      myBird.swim();
```

```
        // Hierarchical Inheritance
        Cat myCat = new Cat();
        myCat.eat();
        myCat.meow();

        Lion myLion = new Lion();
        myLion.eat();
        myLion.roar();

        // Hybrid Inheritance
        Parrot myParrot = new Parrot();
        myParrot.eat();
        myParrot.fly();
        myParrot.talk();
    }
}
```

Output:

```
Animal is eating
Dog is barking
Animal is eating
Dog is barking
Poodle is being groomed
Bird is eating
Bird is flying
Bird is swimming
Animal is eating
Cat is meowing
Animal is eating
Lion is roaring
Bird is eating
Bird is flying
Parrot is talking
```

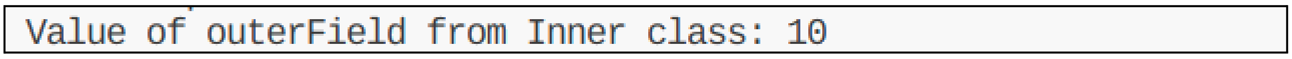
Q6) Constructing java programs to see the working of Inner Classes and Static Inner Classes in java.

*/*** CODE **/*

InnerClassesExample.java:

```java
// Outer class
class Outer {
  private int outerField = 10;

  // Inner class
  class Inner {
    void display() {
      System.out.println("Value of outerField from Inner class: " + outerField);
    }
  }

  // Method to use the inner class
  void useInner() {
    Inner innerObj = new Inner();
    innerObj.display();
  }
}

class InnerClassesExample {
  public static void main(String[] args) {
    // Creating an instance of the outer class
    Outer outerObj = new Outer();

    // Using the outer class method, which in turn uses the inner class
    outerObj.useInner();
  }
}
```

Output:



```
Value of outerField from Inner class: 10
```

StaticInnerClassesExample.java:

```java
// Outer class
class OuterStatic {
  private static int outerStaticField = 20;

  // Static inner class
  static class StaticInner {
     void display() {
        System.out.println("Value of outerStaticField from StaticInner class: " + outerStaticField);
     }
  }

  // Static method to use the static inner class
  static void useStaticInner() {
     StaticInner staticInnerObj = new StaticInner();
     staticInnerObj.display();
  }
}

class StaticInnerClassesExample {
  public static void main(String[] args) {
     // Using the static inner class directly
     OuterStatic.StaticInner staticInnerObj = new OuterStatic.StaticInner();
     staticInnerObj.display();

     // Using the static inner class through the outer class method
     OuterStatic.useStaticInner();
  }
}
```

Output:

```
Value of outerStaticField from StaticInner class: 20
Value of outerStaticField from StaticInner class: 20
```

Q7) Constructing java programs to see the working of Exception Handling in java.

*/** CODE **/*

```java
import java.util.Scanner;

// Custom exception for divide by zero
class CustomDivideByZeroException extends Exception {
  public CustomDivideByZeroException(String message) {
     super(message);
  }
}

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

     try {
        System.out.print("Enter a numerator: ");
        int numerator = scanner.nextInt();

        System.out.print("Enter a denominator: ");
        int denominator = scanner.nextInt();

        // Division operation that may cause an exception
        int result = divide(numerator, denominator);

        // Displaying the result
        System.out.println("Result of division: " + result);
     } catch (CustomDivideByZeroException e) {
        // Catch block for handling CustomDivideByZeroException
        System.out.println("Error: " + e.getMessage());
     } catch (Exception e) {
        // Catch block for handling other exceptions
        System.out.println("An unexpected error occurred: " + e.getMessage());
     } finally {
        // Finally block to close resources or perform cleanup (always executed)
        System.out.println("Finally block executed");
        scanner.close();
     }
  }

  // Method to perform division with throws clause
  private static int divide(int numerator, int denominator) throws CustomDivideByZeroException {
     if (denominator == 0) {
        // Throw the custom exception when attempting to divide by zero
        throw new CustomDivideByZeroException("Division by zero is not allowed");
      }
      return numerator / denominator;
    }
}
```

Output:

```
Enter a numerator: 20
Enter a denominator: 0
Error: Division by zero is not allowed
Finally block executed
```

```
Enter a numerator: 20
Enter a denominator: 10
Result of division: 2
Finally block executed
```

Q8) Creating threads using Thread Class, Runnable Interface and Anonymous Implementations.

*/*** CODE ***/*

AnonymousThreadExample.java

```
class AnonymousThreadExample {
  public static void main(String[] args) {
    // Creating and starting a thread using anonymous implementation of Runnable
    Thread myThread = new Thread(new Runnable() {
      @Override
      public void run() {
        for (int i = 1; i <= 5; i++) {
          System.out.println("Thread using Anonymous Implementation - Count: " + i);
        }
      }
    });
    myThread.start();

    // Main thread's execution
    for (int i = 1; i <= 5; i++) {
      System.out.println("Main Thread - Count: " + i);
    }
  }
}
```

Output:

```
Thread using Anonymous Implementation - Count: 1
Main Thread - Count: 1
Thread using Anonymous Implementation - Count: 2
Main Thread - Count: 2
Thread using Anonymous Implementation - Count: 3
Main Thread - Count: 3
Thread using Anonymous Implementation - Count: 4
Main Thread - Count: 4
Thread using Anonymous Implementation - Count: 5
Main Thread - Count: 5
```

RunnableInterfaceExample.java

```java
// Runnable interface implementation
class MyRunnable implements Runnable {
  @Override
  public void run() {
    for (int i = 1; i <= 5; i++) {
      System.out.println("Thread using Runnable interface - Count: " + i);
    }
  }
}

class RunnableInterfaceExample {
  public static void main(String[] args) {
    // Creating a Runnable object
    MyRunnable myRunnable = new MyRunnable();

    // Creating and starting a thread using the Runnable interface
    Thread myThread = new Thread(myRunnable);
    myThread.start();

    // Main thread's execution
    for (int i = 1; i <= 5; i++) {
      System.out.println("Main Thread - Count: " + i);
    }
  }
}
```

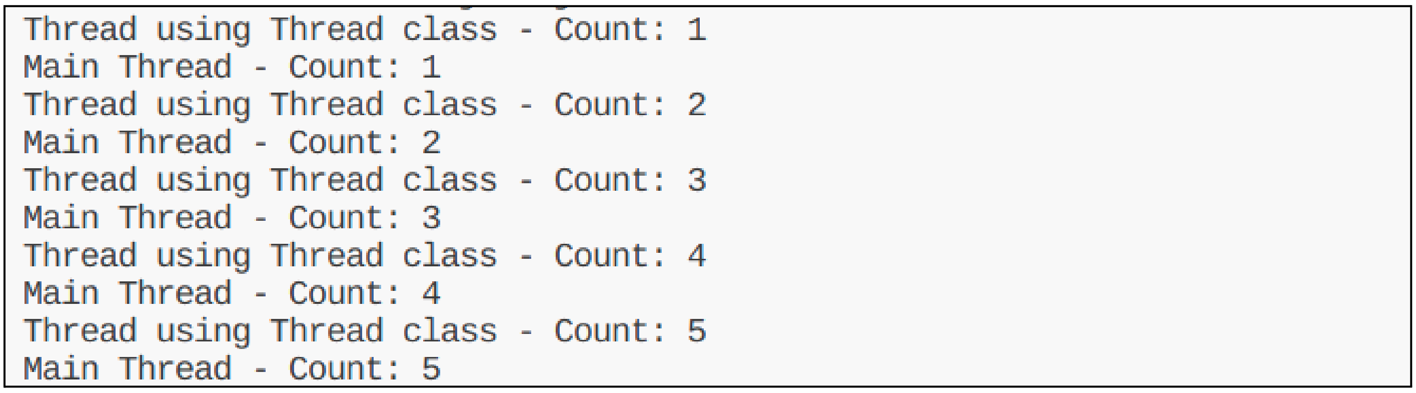
Output:

```
Main Thread - Count: 1
Thread using Runnable interface - Count: 1
Main Thread - Count: 2
Thread using Runnable interface - Count: 2
Main Thread - Count: 3
Thread using Runnable interface - Count: 3
Main Thread - Count: 4
Thread using Runnable interface - Count: 4
Main Thread - Count: 5
Thread using Runnable interface - Count: 5
```

Thread.java:

```java
// Thread class implementation
class MyThread extends Thread {
  @Override
  public void run() {
    for (int i = 1; i <= 5; i++) {
      System.out.println("Thread using Thread class - Count: " + i);
    }
  }
}

class ThreadClassExample {
  public static void main(String[] args) {
    // Creating and starting a thread using the Thread class
    MyThread myThread = new MyThread();
    myThread.start();

    // Main thread's execution
    for (int i = 1; i <= 5; i++) {
      System.out.println("Main Thread - Count: " + i);
    }
  }
}
```

Output:



```
Thread using Thread class - Count: 1
Main Thread - Count: 1
Thread using Thread class - Count: 2
Main Thread - Count: 2
Thread using Thread class - Count: 3
Main Thread - Count: 3
Thread using Thread class - Count: 4
Main Thread - Count: 4
Thread using Thread class - Count: 5
Main Thread - Count: 5
```

Q9) Familiarizing the concept of block, method and volatile synchronization in Threads and File Handling.

*/** CODE ***/*

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

class SharedResource {
  // Using volatile for synchronization
  private volatile int counter = 0;

  public void increment() {
    counter++;
  }

  public int getCounter() {
    return counter;
  }
}

class IncrementerThread extends Thread {
  private SharedResource sharedResource;

  public IncrementerThread(SharedResource sharedResource) {
    this.sharedResource = sharedResource;
  }

  @Override
  public void run() {
    for (int i = 0; i < 1000; i++) {
      sharedResource.increment();
    }
  }
}

class Main {
  public static void main(String[] args) {
    SharedResource sharedResource = new SharedResource();

    IncrementerThread thread1 = new IncrementerThread(sharedResource);
    IncrementerThread thread2 = new IncrementerThread(sharedResource);

    thread1.start();
    thread2.start();
```

```
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Counter value: " + sharedResource.getCounter());

        // Write counter value to a file
        writeToFile("counter.txt", String.valueOf(sharedResource.getCounter()));
    }

    private static void writeToFile(String fileName, String content) {
        try (FileWriter writer = new FileWriter(new File(fileName))) {
            writer.write(content);
            System.out.println("Counter value written to " + fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Counter value: 1962
Counter value written to counter.txt
```

Q10) Connecting machines over the intranet using the concept of TCP and UDP sockets.

/** *CODE* **/

P10_TCP_client.java:

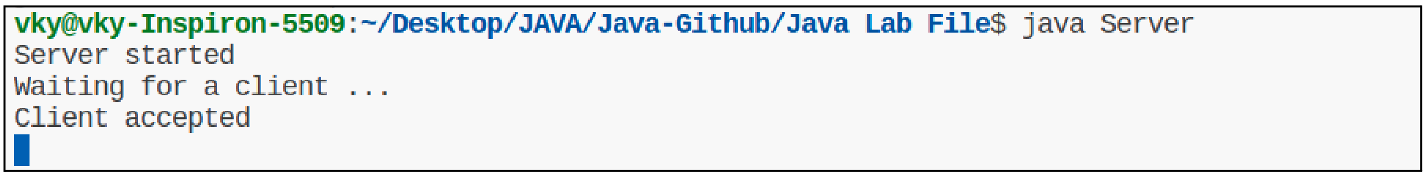
```
import java.io.*;
import java.net.*;

class Client {
    // initialize socket and input output streams
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;

    // constructor to put ip address and port
    public Client(String address, int port)
    {
        // establish a connection
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
            // takes input from terminal
            input = new DataInputStream(System.in);
            // sends output to the socket
            out = new DataOutputStream(
              socket.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
        // string to read message from input
        String line = "";
        // keep reading until "Over" is input
        while (!line.equals("Over")) {
            try {
              line = input.readLine();
              out.writeUTF(line);
            }
            catch (IOException i) {
              System.out.println(i);
            }
        }
        // close the connection
        try {
            input.close();
            out.close();
            socket.close();
        }
        catch (IOException i) {
            System.out.println(i);
        }
    }
    public static void main(String args[])
    {
            Client client = new Client("127.0.0.1", 5000);
    }
}
```

P10_TCP_server.java:

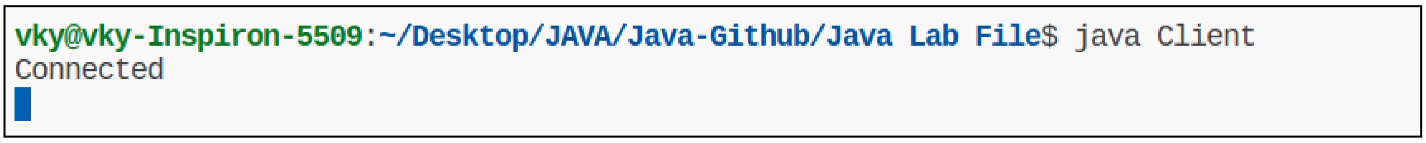
```
import java.net.*;
import java.io.*;

class Server
{
  //initialize socket and input stream
  private Socket      socket = null;
  private ServerSocket server = null;
  private DataInputStream in   = null;
  // constructor with port
```

```
    public Server(int port)
    {
        // starts server and waits for a connection
        try
        {
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client ...");
            socket = server.accept();
            System.out.println("Client accepted");
            // takes input from the client socket
            in = new DataInputStream(
                new BufferedInputStream(socket.getInputStream()));
            String line = "";
            // reads message from client until "Over" is sent
            while (!line.equals("Over"))
            {
                try
                {
                    line = in.readUTF();
                    System.out.println(line);
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");
            // close connection
            socket.close();
            in.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }
    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

Output:



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$ java Server
Server started
Waiting for a client ...
Client accepted
```



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$ java Client
Connected
```

P10_UDP_client.java:

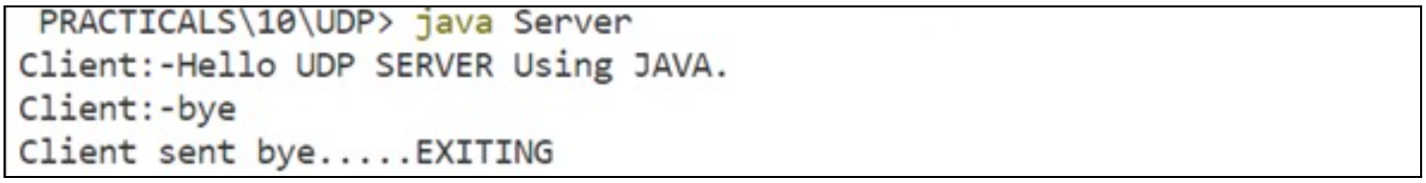
```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

class Client {
  public static void main(String args[]) throws IOException {
     Scanner sc = new Scanner(System.in);
     // Step 1:Create the socket object for
     // carrying the data.
     DatagramSocket ds = new DatagramSocket();

     InetAddress ip = InetAddress.getLocalHost();
     byte buf[] = null;
     // loop while user not enters "bye"
     while (true) {
        String inp = sc.nextLine();
        // convert the String input into the byte array.
        buf = inp.getBytes();
        // Step 2 : Create the datagramPacket for sending
        // the data.
        DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);
        // Step 3 : invoke the send call to actually send
        // the data.
        ds.send(DpSend);



        // break the loop if user enters "bye"

        if (inp.equals("bye"))
           break;
     }
     sc.close();
     ds.close();
  }
}
```

P10_UDP_server.java:

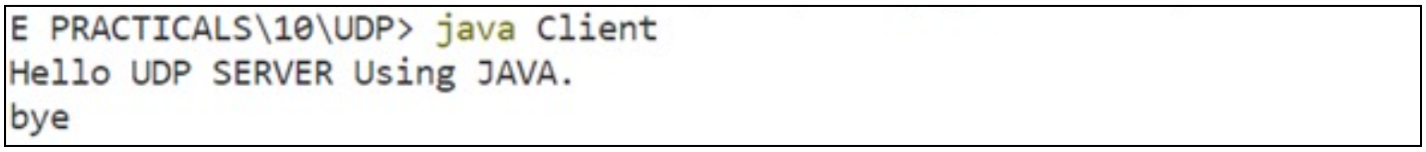
```java
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

class Server {
  public static void main(String[] args) throws IOException {
     // Step 1 : Create a socket to listen at port 1234
     DatagramSocket ds = new DatagramSocket(1234);
     byte[] receive = new byte[65535];
     DatagramPacket DpReceive = null;
     while (true) {
        // Step 2 : create a DatgramPacket to receive the data.
```

```
            DpReceive = new DatagramPacket(receive, receive.length);
            // Step 3 : revieve the data in byte buffer.
            ds.receive(DpReceive);
            System.out.println("Client:-" + data(receive));
            // Exit the server if the client sends "bye"
            if (data(receive).toString().equals("bye")) {
                System.out.println("Client sent bye.....EXITING");
                break;
            }

            // Clear the buffer after every message.
            receive = new byte[65535];
        }
        ds.close();
    }

    // A utility method to convert the byte array
    // data into a string representation.
    public static StringBuilder data(byte[] a) {
        if (a == null)
            return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0) {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }
}
```

Output:



```
 PRACTICALS\10\UDP> java Server
Client:-Hello UDP SERVER Using JAVA.
Client:-bye
Client sent bye.....EXITING
```



```
E PRACTICALS\10\UDP> java Client
Hello UDP SERVER Using JAVA.
bye
```

Q11) Creating Java Applications for implementing File Handling for reading/writing data from persistent storage and vice-versa.

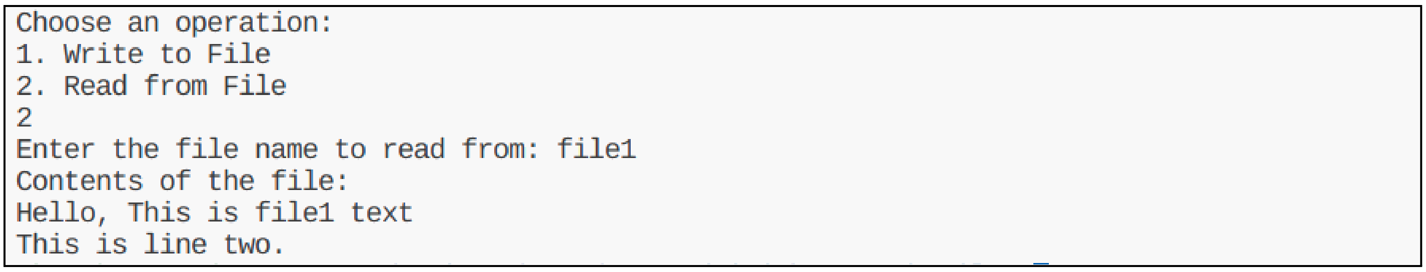
*/** CODE **/*

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

    System.out.println("Choose an operation:");
    System.out.println("1. Write to File");
    System.out.println("2. Read from File");
    int choice = scanner.nextInt();

    switch (choice) {
      case 1:
        writeToTextFile();
        break;
      case 2:
        readFromTextFile();
        break;
      default:
        System.out.println("Invalid choice. Exiting.");
    }

    scanner.close();
  }

  private static void writeToTextFile() {
    try {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the file name to write to: ");
      String fileName = scanner.nextLine();

      FileWriter fileWriter = new FileWriter(fileName);

      System.out.println("Enter text to write to the file (type 'exit' to finish):");
      while (true) {
        String line = scanner.nextLine();
```

```java
            if (line.equals("exit")) {
                break;
            }
            fileWriter.write(line + "\n");
        }

        fileWriter.close();
        scanner.close();
        System.out.println("Data written to the file successfully.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private static void readFromTextFile() {
    try {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the file name to read from: ");
        String fileName = scanner.nextLine();

        FileReader fileReader = new FileReader(fileName);
        BufferedReader bufferedReader = new BufferedReader(fileReader);

        System.out.println("Contents of the file:");
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            System.out.println(line);
        }

        bufferedReader.close();
        scanner.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
```

Output:

```
Choose an operation:
1. Write to File
2. Read from File
1
Enter the file name to write to: file1
Enter text to write to the file (type 'exit' to finish):
Hello, This is file1 text
This is line two.
exit
Data written to the file successfully.
```



```
Choose an operation:
1. Write to File
2. Read from File
2
Enter the file name to read from: file1
Contents of the file:
Hello, This is file1 text
This is line two.
```

Q12) Exploring the Collections Framework and various collection types in Java.

```
/*****************************************  CODE  ****************************************/
import java.util.*;
class CollectionsFrameworkExample {
  public static void main(String[] args) {
     // List Example
     List<String> arrayList = new ArrayList<>();
     arrayList.add("Item 1");
     arrayList.add("Item 2");
     arrayList.add("Item 3");

     List<String> linkedList = new LinkedList<>();
     linkedList.add("Node 1");
     linkedList.add("Node 2");
     linkedList.add("Node 3");

     // Set Example
     Set<String> hashSet = new HashSet<>();
     hashSet.add("Element 1");
     hashSet.add("Element 2");
     hashSet.add("Element 3");

     Set<String> treeSet = new TreeSet<>();
     treeSet.add("B");
     treeSet.add("A");
     treeSet.add("C");

     // Queue Example
     Queue<String> queue = new LinkedList<>();
     queue.offer("Task 1");
     queue.offer("Task 2");
     queue.offer("Task 3");

     Queue<String> priorityQueue = new PriorityQueue<>();
     priorityQueue.offer("Priority 3");
     priorityQueue.offer("Priority 1");
     priorityQueue.offer("Priority 2");


     // Map Example
     Map<Integer, String> hashMap = new HashMap<>();
     hashMap.put(1, "Value 1");
```

```
        hashMap.put(2, "Value 2");
        hashMap.put(3, "Value 3");

        Map<String, Integer> treeMap = new TreeMap<>();
        treeMap.put("Three", 3);
        treeMap.put("One", 1);
        treeMap.put("Two", 2);

        // Print Results
        System.out.println("ArrayList: " + arrayList);
        System.out.println("LinkedList: " + linkedList);
        System.out.println("HashSet: " + hashSet);
        System.out.println("TreeSet: " + treeSet);
        System.out.println("Queue: " + queue);
        System.out.println("PriorityQueue: " + priorityQueue);
        System.out.println("HashMap: " + hashMap);
        System.out.println("TreeMap: " + treeMap);
    }
}
```

Output:

```
ArrayList: [Item 1, Item 2, Item 3]
LinkedList: [Node 1, Node 2, Node 3]
HashSet: [Element 1, Element 3, Element 2]
TreeSet: [A, B, C]
Queue: [Task 1, Task 2, Task 3]
PriorityQueue: [Priority 1, Priority 3, Priority 2]
HashMap: {1=Value 1, 2=Value 2, 3=Value 3}
TreeMap: {One=1, Three=3, Two=2}
```

Q13) Implementing NetBeans IDE for GUI Development in Java by means of AWT and Swings Framework.

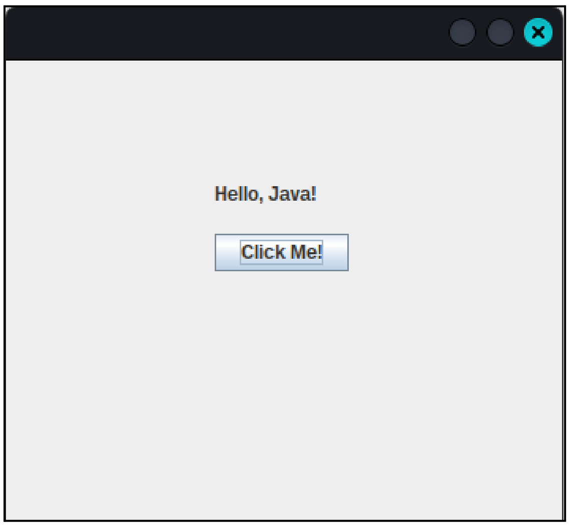
*/** CODE ***/*

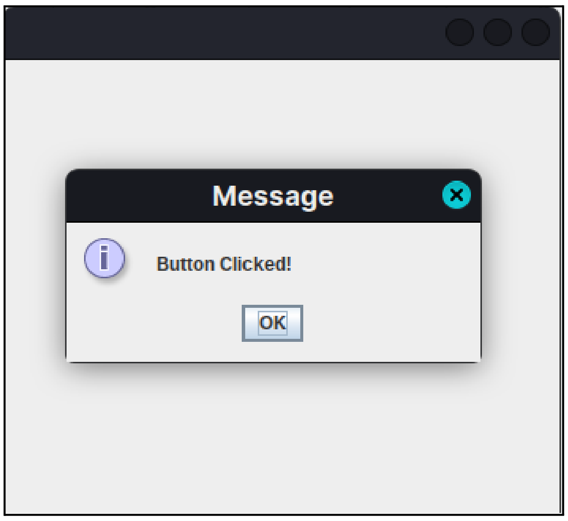
```
import javax.swing.*;

class MainGUI extends javax.swing.JFrame {

  public MainGUI() {
    initComponents();
  }

  private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
```

```
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Hello, NetBeans!");

        jButton1.setText("Click Me!");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(132, 132, 132)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap(135, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(78, 78, 78)
                .addComponent(jLabel1)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap(163, Short.MAX_VALUE))
        );

        pack();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(this, "Button Clicked!");
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
        public void run() {
            new MainGUI().setVisible(true);
        }
    });
  }

  private javax.swing.JButton jButton1;
  private javax.swing.JLabel jLabel1;
}
```

Output:





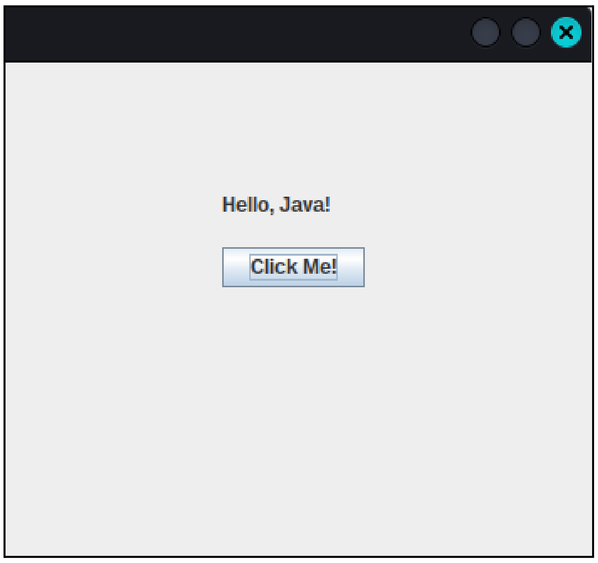
Q14) Introducing event handling model in the same to make intuitive and responsive GUI with the help of NetBeans/ Eclipse IDE.

*/*** CODE ***/*

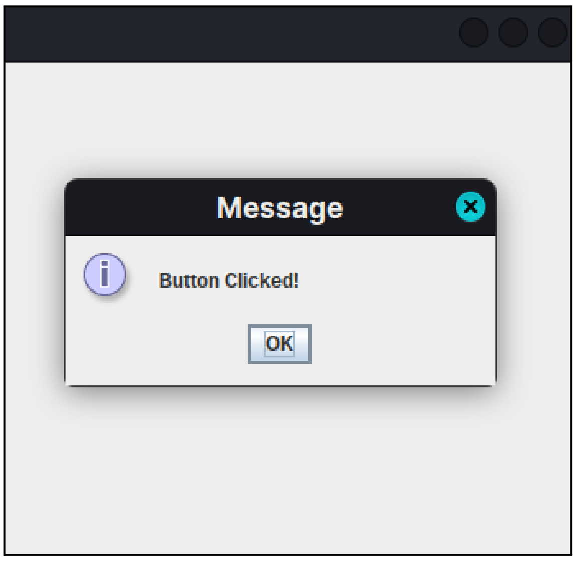
```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

class MainGUI extends javax.swing.JFrame {

  private javax.swing.JButton jButton1;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JTextField jTextField1;

  public MainGUI() {
     initComponents();
  }
  private void initComponents() {
     jLabel1 = new javax.swing.JLabel();
     jTextField1 = new javax.swing.JTextField();
```

```
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Hello, NetBeans!");

        jTextField1.setText("Enter your name");

        jButton1.setText("Click Me!");
        jButton1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(132, 132, 132)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jTextField1)
                    .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap(135, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(78, 78, 78)
                .addComponent(jLabel1)
                .addGap(18, 18, 18)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap(130, Short.MAX_VALUE))
        );
        pack();
    }

    private void jButton1ActionPerformed(ActionEvent evt) {
        String name = jTextField1.getText();
        JOptionPane.showMessageDialog(this, "Hello, " + name + "!");
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainGUI().setVisible(true);
            }
        });
    }
}
```

Output:



Hello, Java!



Message

Button Clicked!

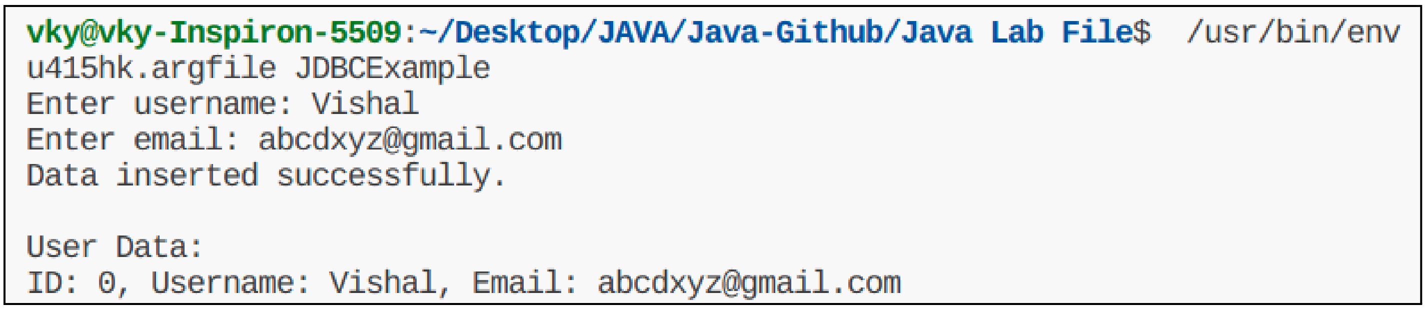
Q15) Connecting Java applications to underlying databases using JDBC API.

*/***   CODE   ***/*

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

class JDBCExample {

    static final String JDBC_URL = "jdbc:mysql://localhost:3306/jdbcdemo";
    static final String USER = "vky";
    static final String PASSWORD = "root123";

    public static void main(String[] args) {
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
```

```
        // Step 1: Register JDBC driver
        Class.forName("com.mysql.cj.jdbc.Driver");

        // Step 2: Open a connection
        connection = DriverManager.getConnection(JDBC_URL, USER, PASSWORD);

        // Step 3: Check if the table exists, create it if not
        if (!tableExists(connection, "users")) {
          createTable(connection);
        }

        // Step 4: Get user input
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter username: ");
        String username = scanner.nextLine();
        System.out.print("Enter email: ");
        String email = scanner.nextLine();

        // Step 5: Execute INSERT query
        statement = connection.createStatement();
        String insertQuery = "INSERT INTO users (username, email) VALUES ('" + username + "', '" + email +
"')";
        int rowsAffected = statement.executeUpdate(insertQuery);

        if (rowsAffected > 0) {
          System.out.println("Data inserted successfully.");
        } else {
          System.out.println("Failed to insert data.");
        }

        // Step 6: Display all data from the table
        displayUserData(connection);
        scanner.close();

      } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
      } finally {
        // Step 7: Close the resources
        try {
          if (resultSet != null)
            resultSet.close();
          if (statement != null)
            statement.close();
          if (connection != null)
            connection.close();
```

```java
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    private static boolean tableExists(Connection connection, String tableName) throws SQLException {
        DatabaseMetaData meta = connection.getMetaData();
        ResultSet resultSet = meta.getTables(null, null, tableName, new String[] { "TABLE" });
        return resultSet.next();
    }

    private static void createTable(Connection connection) throws SQLException {
        Statement statement = connection.createStatement();
        String createTableQuery = "CREATE TABLE users (id INT PRIMARY KEY AUTO_INCREMENT, "
                + "username VARCHAR(50), email VARCHAR(50))";
        statement.executeUpdate(createTableQuery);
        statement.close();
    }

    private static void displayUserData(Connection connection) throws SQLException {
        Statement statement = connection.createStatement();
        String sqlQuery = "SELECT id, username, email FROM users";
        ResultSet resultSet = statement.executeQuery(sqlQuery);

        // Display the results
        System.out.println("\nUser Data:");
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String username = resultSet.getString("username");
            String email = resultSet.getString("email");

            System.out.println("ID: " + id + ", Username: " + username + ", Email: " + email);
        }
        resultSet.close();
        statement.close();
    }
}
```
Output:



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$  /usr/bin/env
u415hk.argfile JDBCExample
Enter username: Vishal
Enter email: abcdxyz@gmail.com
Data inserted successfully.

User Data:
ID: 0, Username: Vishal, Email: abcdxyz@gmail.com
```

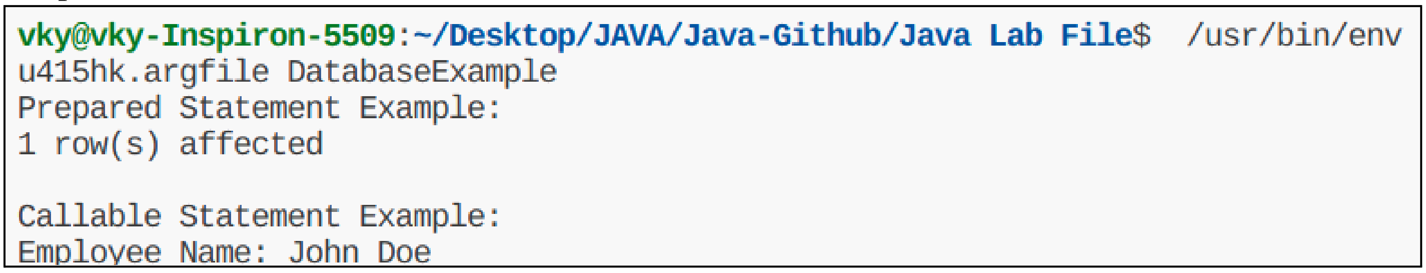
Q16) Exploring Prepared Statement and Callable Statement Interfaces for database connectivity.

*/**   CODE   **/*

```
import java.sql.*;

class DatabaseExample {
  // JDBC URL, username, and password of MySQL server
  private static final String JDBC_URL = "jdbc:mysql://localhost:3306/jdbcdemo";
  private static final String USER = "vky";
  private static final String PASSWORD = "root123";

  public static void main(String[] args) {
    try {
      // Load the JDBC driver
      Class.forName("com.mysql.cj.jdbc.Driver");

      // Establish a connection
      Connection connection = DriverManager.getConnection(JDBC_URL, USER, PASSWORD);

      // Example using PreparedStatement
      preparedStatementExample(connection);

      // Example using CallableStatement
      callableStatementExample(connection);

      // Close the connection
      connection.close();
    } catch (ClassNotFoundException | SQLException e) {
      e.printStackTrace();
    }
  }

  private static void preparedStatementExample(Connection connection) throws SQLException {
    System.out.println("Prepared Statement Example:");

    // Create a prepared statement
    String insertQuery = "INSERT INTO employee (id, name, salary) VALUES (?, ?, ?)";
    PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

    // Set values for the prepared statement
    preparedStatement.setInt(1, 101);
    preparedStatement.setString(2, "John Doe");
    preparedStatement.setDouble(3, 50000.00);

    // Execute the prepared statement
```

```java
        int rowsAffected = preparedStatement.executeUpdate();
        System.out.println(rowsAffected + " row(s) affected");

        // Close the prepared statement
        preparedStatement.close();
    }
    private static void callableStatementExample(Connection connection) throws SQLException {
        System.out.println("\nCallable Statement Example:");

        // Create a callable statement for calling a stored procedure
        String storedProcedureCall = "{CALL getEmployeeDetails(?, ?)}";
        CallableStatement callableStatement = connection.prepareCall(storedProcedureCall);

        // Set input parameter for the stored procedure
        callableStatement.setInt(1, 101);

        // Register output parameter for the stored procedure
        callableStatement.registerOutParameter(2, Types.VARCHAR);

        // Execute the stored procedure
        callableStatement.execute();

        // Retrieve the output parameter value
        String employeeName = callableStatement.getString(2);
        System.out.println("Employee Name: " + employeeName);

        // Close the callable statement
        callableStatement.close();
    }
}
```

Output:



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$  /usr/bin/env
u415hk.argfile DatabaseExample
Prepared Statement Example:
1 row(s) affected

Callable Statement Example:
Employee Name: John Doe
```

Q17) Constructing RMI client server applications to connect two remote machines for method access.

*/** CODE ***/*

Hello.java:

```
import java.rmi.*;

import java.rmi.server.*;

public class Hello extends UnicastRemoteObject implements HelloInterface {

  private String message;

  public Hello(String msg) throws RemoteException {

     message = msg;

  }

  public String say() throws RemoteException {

     return message;

  }

}
```

HelloClient.java:

```
import java.rmi.*;

public class HelloClient {

  public static void main(String[] argv) {

     try {

        HelloInterface hello = (HelloInterface) Naming.lookup("//localhost/Hello");

        System.out.println(hello.say());

     } catch (Exception e) {

        System.out.println("HelloClient exception: " + e);

     }

  }

}
```

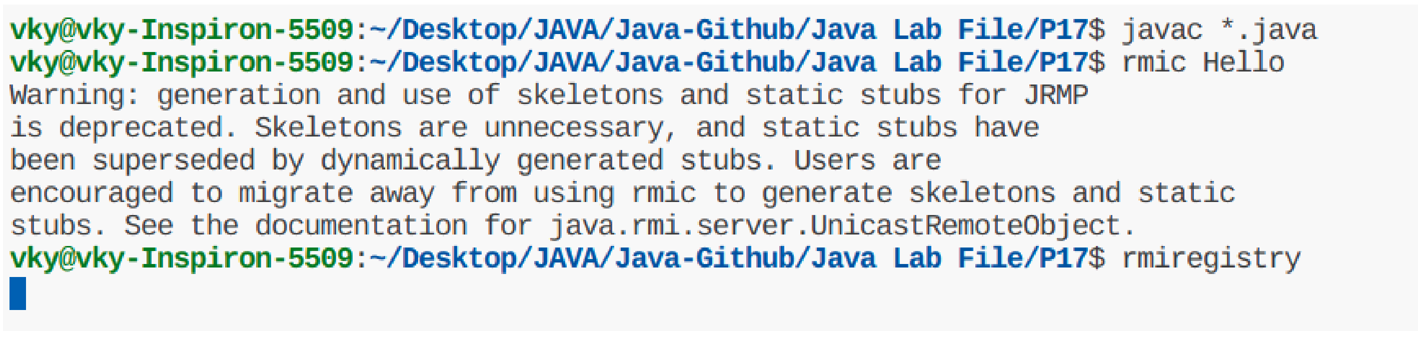
HelloInterface.java:

```
import java.rmi.*;

public interface HelloInterface extends Remote {

  public String say() throws RemoteException;

}
```

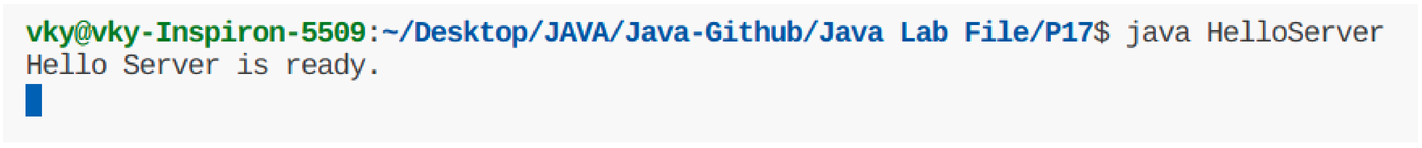
HelloServer.java:

```java
import java.rmi.*;

public class HelloServer {

  public static void main(String[] argv) {

     try {

        Hello robj = new Hello("Hello, world!");

        Naming.rebind("Hello", robj);

        System.out.println("Hello Server is ready.");

     } catch (Exception e) {

        System.out.println("Hello Server failed: " + e);

     }

  }

}
```

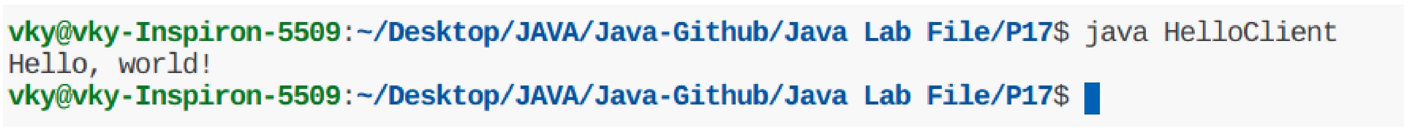
Output:



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$ javac *.java
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$ rmic Hello
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$ rmiregistry
```



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$ java HelloServer
Hello Server is ready.
```



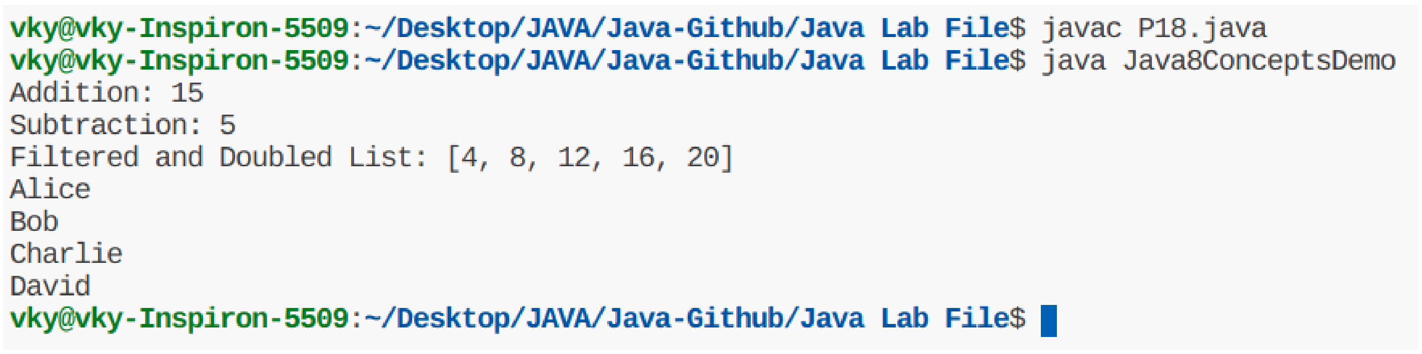
```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$ java HelloClient
Hello, world!
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File/P17$
```

Q18) Implementing Java 8 concepts.

*/** CODE **/*

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

// Define a functional interface for a simple operation
@FunctionalInterface
interface Operation {
  int perform(int a, int b);
}

class Java8ConceptsDemo {

  public static void main(String[] args) {
     // Example 1: Lambda expressions
     Operation addition = (a, b) -> a + b;
     Operation subtraction = (a, b) -> a - b;

     System.out.println("Addition: " + operate(10, 5, addition));
     System.out.println("Subtraction: " + operate(10, 5, subtraction));

     // Example 2: Stream API
     List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

     // Use stream to filter even numbers, double them, and collect the result in a new list
     List<Integer> result = numbers.stream()
          .filter(n -> n % 2 == 0)
          .map(n -> n * 2)
          .collect(Collectors.toList());

     System.out.println("Filtered and Doubled List: " + result);

     // Example 3: Method references
     List<String> names = Arrays.asList("Alice", "Bob", "Charlie", "David");

     // Use method reference to print each name
     names.forEach(System.out::println);
  }
  // Method that takes an operation as a parameter and performs it on two numbers
  private static int operate(int a, int b, Operation operation) {
     return operation.perform(a, b);
  }
}
```

Output:



```
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$ javac P18.java
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$ java Java8ConceptsDemo
Addition: 15
Subtraction: 5
Filtered and Doubled List: [4, 8, 12, 16, 20]
Alice
Bob
Charlie
David
vky@vky-Inspiron-5509:~/Desktop/JAVA/Java-Github/Java Lab File$
```