Answer 1:

// Define a custom exception class for internal marks

class InternalMarksException extends Exception {

// Constructor with a message parameter

public InternalMarksException(String message) {

// Call the super class constructor with the message

super(message);

}

}

// Define a custom exception class for external marks

class ExternalMarksException extends Exception {

// Constructor with a message parameter

public ExternalMarksException(String message) {

// Call the super class constructor with the message

super(message);

}

}

// Define a class to represent a student

class Student {

// Declare instance variables for name, internal marks and external marks

private String name;

private int internalMarks;

private int externalMarks;

// Constructor with three parameters

public Student(String name, int internalMarks, int externalMarks) {

// Assign the parameters to the instance variables

this.name = name;

this.internalMarks = internalMarks;

this.externalMarks = externalMarks;

}

// Getter method for name

public String getName() {

return name;

}

// Getter method for internal marks

public int getInternalMarks() {

return internalMarks;

}

// Getter method for external marks

public int getExternalMarks() {

return externalMarks;

}

// A method to check the marks and throw exceptions if needed

public void checkMarks() throws InternalMarksException, ExternalMarksException {

// If the internal marks is greater than 30, throw an InternalMarksException

if (internalMarks > 30) {

throw new InternalMarksException("Internal marks exceeded");

}

// If the external marks is greater than 70, throw an ExternalMarksException

if (externalMarks > 70) {

throw new ExternalMarksException("External mark exceeded");

}

// Otherwise, print the name and marks of the student

System.out.println("Name: " + name);

System.out.println("Internal marks: " + internalMarks);

System.out.println("External marks: " + externalMarks);

}

}

// Define a main class to test the exceptions

public class Main {

public static void main(String[] args) {

// Create an array of four student objects with different marks

Student[] students = new Student[4];

students[0] = new Student("Alice", 25, 65);

students[1] = new Student("Bob", 35, 60);

students[2] = new Student("Charlie", 30, 75);

students[3] = new Student("David", 40, 80);

// Loop through the array and call the checkMarks method for each student

for (Student student : students) {

try {

student.checkMarks();

} catch (InternalMarksException e) {

// Catch and print the InternalMarksException message

System.out.println(e.getMessage());

} catch (ExternalMarksException e) {

// Catch and print the ExternalMarksException message

System.out.println(e.getMessage());

}

// Print a blank line after each student

System.out.println();

}

}

}

Answer 2:

// Define a class that implements Runnable and implements the logic for adding a part of the array

class AdderRunnable implements Runnable {

// Declare instance variables for the array, the start index, the end index and the sum

private int[] array;

private int start;

private int end;

private int sum;

// Constructor with three parameters

public AdderRunnable(int[] array, int start, int end) {

// Assign the parameters to the instance variables

this.array = array;

this.start = start;

this.end = end;

// Initialize the sum to zero

this.sum = 0;

}

// Getter method for sum

public int getSum() {

return sum;

}

// Override the run method to perform the addition

@Override

public void run() {

// Loop from the start index to the end index (inclusive) and add the array elements to the sum

for (int i = start; i <= end; i++) {

sum += array[i];

}

// Print a message with the thread name and the sum

System.out.println(Thread.currentThread().getName() + " computed sum: " + sum);

}

}

// Define a main class to test the program

public class Main {

public static void main(String[] args) throws InterruptedException {

// Create an array of nine numbers

int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9};

// Create three AdderRunnable objects with different parts of the array

AdderRunnable r1 = new AdderRunnable(array, 0, 2); // First three elements

AdderRunnable r2 = new AdderRunnable(array, 3, 5); // Middle three elements

AdderRunnable r3 = new AdderRunnable(array, 6, 8); // Last three elements

// Create three Thread objects with the AdderRunnable objects as the target

Thread t1 = new Thread(r1);

Thread t2 = new Thread(r2);

Thread t3 = new Thread(r3);

// Assign names to the threads

t1.setName("Thread-1");

t2.setName("Thread-2");

t3.setName("Thread-3");

// Start the threads

t1.start();

t2.start();

t3.start();

// Wait for the threads to finish using join method

t1.join();

t2.join();

t3.join();

// Compute the total sum by adding the sums of each thread

int totalSum = r1.getSum() + r2.getSum() + r3.getSum();

// Print a message with the total sum

System.out.println("Total sum: " + totalSum);

}

}

Answer 3:

// Import the necessary classes

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

// Define a class that extends JFrame and implements ActionListener

public class EchoServer extends JFrame implements ActionListener {

// Declare instance variables for the GUI components

private JTextArea textArea;

private JButton startButton;

private JButton stopButton;

private JLabel statusLabel;

// Declare instance variables for the server socket and the thread

private ServerSocket serverSocket;

private Thread serverThread;

// Declare a constant for the port number

private static final int PORT = 1234;

// Constructor with no parameters

public EchoServer() {

// Call the super class constructor with the title

super("Echo Server");

// Initialize the GUI components

textArea = new JTextArea(10, 30);

textArea.setEditable(false);

textArea.setLineWrap(true);

textArea.setWrapStyleWord(true);

startButton = new JButton("Start");

startButton.addActionListener(this);

stopButton = new JButton("Stop");

stopButton.addActionListener(this);

stopButton.setEnabled(false);

statusLabel = new JLabel("Not running");

// Add the GUI components to the frame using a border layout

add(new JScrollPane(textArea), BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.add(startButton);

panel.add(stopButton);

panel.add(statusLabel);

add(panel, BorderLayout.SOUTH);

// Set the frame properties

pack();

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

// Override the actionPerformed method to handle button clicks

@Override

public void actionPerformed(ActionEvent e) {

// If the start button is clicked, start the server thread

if (e.getSource() == startButton) {

startServer();

}

// If the stop button is clicked, stop the server thread

else if (e.getSource() == stopButton) {

stopServer();

}

}

// A method to start the server thread

public void startServer() {

try {

// Create a server socket with the port number

serverSocket = new ServerSocket(PORT);

// Create a thread with a runnable object that calls the runServer method

serverThread = new Thread(() -> runServer());

// Start the thread

serverThread.start();

// Update the GUI components

textArea.append("Server started on port " + PORT + "\n");

startButton.setEnabled(false);

stopButton.setEnabled(true);

statusLabel.setText("Running");

} catch (IOException e) {

// Handle any IO exception by showing an error message

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// A method to stop the server thread

public void stopServer() {

try {

// Close the server socket

serverSocket.close();

// Interrupt the thread

serverThread.interrupt();

// Update the GUI components

textArea.append("Server stopped\n");

startButton.setEnabled(true);

stopButton.setEnabled(false);

statusLabel.setText("Not running");

} catch (IOException e) {

// Handle any IO exception by showing an error message

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// A method to run the server logic in a loop

public void runServer() {

try {

while (true) {

// Accept a client connection and create a socket

Socket socket = serverSocket.accept();

// Create a thread with a runnable object that calls the handleClient method with the socket as a parameter

Thread clientThread = new Thread(() -> handleClient(socket));

// Start the thread

clientThread.start();

// Append a message to the text area with the client address and port number

textArea.append("Connected to client " + socket.getInetAddress() + ":" + socket.getPort() + "\n");

}

} catch (IOException e) {

// Handle any IO exception by showing an error message

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// A method to handle a client connection in a separate thread

public void handleClient(Socket socket) {

try {

// Create a buffered reader and a print writer for the socket input and output streams

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

while (true) {

// Read a line from the client and check if it is null or "bye"

String line = in.readLine();

if (line == null || line.equalsIgnoreCase("bye")) {

// If so, break the loop

break;

}

// Otherwise, echo the line back to the client and append it to the text area

out.println(line);

textArea.append("Echoed to client " + socket.getInetAddress() + ":" + socket.getPort() + ": " + line + "\n");

}

// Close the socket and the streams

socket.close();

in.close();

out.close();

// Append a message to the text area with the client address and port number

textArea.append("Disconnected from client " + socket.getInetAddress() + ":" + socket.getPort() + "\n");

} catch (IOException e) {

// Handle any IO exception by showing an error message

JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

// A main method to create an instance of the EchoServer class

public static void main(String[] args) {

new EchoServer();

}

}