

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP
TH LẬP TRÌNH MẠNG

Giảng viên: Th.S Nguyễn Văn Nguyên

Nhóm: 16.10B

Sinh viên thực hiện: Nguyễn Thái Học

Lớp: 16T1

Mã số sinh viên: 102160045

Đà Nẵng, 12/2019

MỤC LỤC

<i>DANH MỤC HÌNH ẢNH.....</i>	<i>3</i>
<i>I. BÀI THỰC HÀNH SỐ 1.....</i>	<i>4</i>
1. Bài tập 1.....	4
2. Bài tập 2.....	5
3. Bài tập 3.....	10
<i>II. BÀI THỰC HÀNH SỐ 2.....</i>	<i>14</i>
1. Bài tập 1.....	14
2. Bài tập 2.....	17
3. Bài tập 3.....	22
<i>III. BÀI THỰC HÀNH SỐ 3.....</i>	<i>27</i>

DANH MỤC HÌNH ẢNH

Hình 1. Kết quả chạy chương trình bài 1.1

Hình 2. Kết quả chạy chương trình bài 1.2

Hình 3. Kết quả chạy chương trình bài 1.3

Hình 4. Kết quả chạy chương trình bài 2.1

Hình 5. Kết quả chạy chương trình bài 2.2

Hình 6. Kết quả chạy chương trình bài 2.3

Hình 7. Giao diện đăng nhập

Hình 8. Giao diện đăng kí

Hình 9. Giao diện danh sách todo

Hình 10. Giao diện thêm todo

Hình 11. Giao diện chỉnh sửa todo

I. BÀI THỰC HÀNH SỐ 1

1. Bài tập 1

Xây dựng chương trình hội thoại Client/Server hoạt động theo giao thức TCP/IP

- Chương trình Server mở cổng và chờ nhận kết nối từ Client
- Client gửi một chuỗi ký tự đến Server. Server nhận và xử lý gửi trả về cho client các công việc:
 - Đổi chuỗi đã gửi thành chuỗi in hoa
 - Đổi chuỗi đã gửi thành chuỗi thường
 - Đếm số từ của chuỗi đã gửi

File Server.java

```
package com.hoc.thltn1.bai1;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) throws IOException {
        final ServerSocket serverSocket = new ServerSocket(5000);

        while (true) {
            final Socket socket = serverSocket.accept();

            new Thread(() -> {
                try {
                    final DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
                    final DataOutputStream dos = new
DataOutputStream(socket.getOutputStream())
                ) {
                    final String s = dataInputStream.readUTF();
                    dos.writeUTF(s.toUpperCase());
                    dos.writeUTF(s.toLowerCase());
                    dos.writeUTF(String.valueOf(s.split("\\W+").length));
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }).start();
        }
    }
}
```

File Client.java

```
package com.hoc.thltn1.bai1;

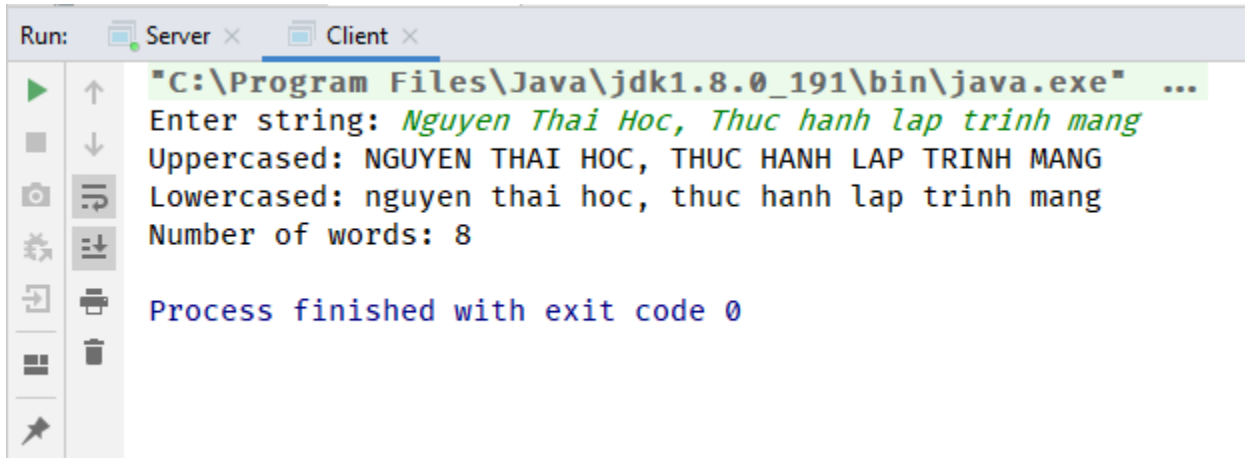
import java.io.*;
import java.net.Socket;

public class Client {
    public static void main(String[] args) throws IOException {
        final Socket socket = new Socket("localhost", 5000);

        try {
            final DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
            final DataOutputStream dataOutputStream = new
DataOutputStream(socket.getOutputStream());
            final BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in))
        ) {
            System.out.print("Enter string: ");
            final String s = bufferedReader.readLine();
            dataOutputStream.writeUTF(s);

            System.out.println("Uppercased: " + dataInputStream.readUTF());
            System.out.println("Lowercased: " + dataInputStream.readUTF());
            System.out.println("Number of words: " +
dataInputStream.readUTF());
        }
    }
}
```

Triển khai và kết quả



```
Run: Server x Client x
"C:\Program Files\Java\jdk1.8.0_191\bin\java.exe" ...
Enter string: Nguyen Thai Hoc, Thuc hanh lap trinh mang
Uppercased: NGUYEN THAI HOC, THUC HANH LAP TRINH MANG
Lowercased: nguyen thai hoc, thuc hanh lap trinh mang
Number of words: 8

Process finished with exit code 0
```

Hình 1. Kết quả chạy chương trình bài 1.1

2. Bài tập 2

Xây dựng chương trình hội thoại Client/Server hoạt động theo giao thức TCP/IP

- Chương trình Client cho phép nhập vào từ bàn phím một chuỗi biểu diễn một phép tính gồm các toán tử +, -, (,). Ví dụ: $5+13-(12-4*6)-((3+4)-5)$
- Chương trình Server thực hiện tính toán và trả kết quả về cho Client

File Server.java

```
package com.hoc.thltn1.bai2;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    public static void main(String[] args) throws IOException {
        final ServerSocket serverSocket = new ServerSocket(5000);

        while (true) {
            final Socket socket = serverSocket.accept();

            new Thread(() -> {
                try {
                    final DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
                    final DataOutputStream dos = new
DataOutputStream(socket.getOutputStream())
                ) {
                    final String s = dataInputStream.readUTF();

                    try {
                        final double result = Eval.execute(s);
                        dos.writeUTF(String.valueOf(result));
                    } catch (Exception e) {
                        dos.writeUTF(e.toString());
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }).start();
        }
    }
}
```

File Eval.java

```
package com.hoc.thltn1.bai2;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Stack;
import java.util.function.BiFunction;

public class Eval {
    public static void main(String[] args) {
        System.out.println(Eval.execute("5+13-(12-4*6) -((3+4)-5)"));
        System.out.println(Eval.execute("1+2+3+4"));
        System.out.println(Eval.execute("1 % 2"));
    }

    private static final Map<String, BiFunction<Double, Double, Double>>
OPERATORS_FUNCTION;

    static {
        OPERATORS_FUNCTION = new HashMap<>();
        OPERATORS_FUNCTION.put("+", Double::sum);
        OPERATORS_FUNCTION.put("-", (x, y) -> x - y);
        OPERATORS_FUNCTION.put("*", (x, y) -> x * y);
        OPERATORS_FUNCTION.put("/", (x, y) -> x / y);
        OPERATORS_FUNCTION.put("%", (x, y) -> x % y);
        OPERATORS_FUNCTION.put("^", Math::pow);
    }

    /*@NonNull*/
    private static String refine(/*@NonNull*/ String infix) {
        final StringBuilder infixBuilder = new StringBuilder(infix);
        final int parenthesesDifferent = countChar(infixBuilder, '(')
            - countChar(infixBuilder, ')');
        if (parenthesesDifferent > 0) {
            infixBuilder.append(
                String.valueOf(new char[parenthesesDifferent])
                    .replace('\0', ')')
            );
        } else if (parenthesesDifferent < 0 || !balancedParentheses(infix))
        {
            return "";
        }

        return infixBuilder.toString()
            .replaceAll("\\s+", "")
            .replaceAll("([*/%^])-(\\d+(\\.\\d+)?)?", "$1(0-$2)")
            .replaceAll("([*/%^])-(\\d+(\\.\\d+)?)?", "$1(-1)*(")
            .replaceAll("\\\\)\\\\(", ")*(")
            .replaceAll("\\\\d\\\\(", "$1*(")
            .replaceAll("[+\\-*/%^()]", " $0 ")
            .replaceAll("\\d+(\\.\\d+)?", "$0 ")
            .trim();
    }

    private static boolean balancedParentheses(/*@NonNull*/ String s) {
        final Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {

```

```

        if (c == '(') {
            stack.push(c);
        } else if (c == ')') {
            if (stack.isEmpty()) {
                return false;
            }
            stack.pop();
        }
    }
    return stack.isEmpty();
}

private static int countChar(/*@NonNull*/ CharSequence s, char ch) {
    return Math.toIntExact(s.chars().filter(c -> c == ch).count());
}

public static double execute(/*@NonNull*/ String infix) {
    final String postfix = infixToPostfix(infix);
    //System.out.println("Postfix: " + postfix);
    return evaluation(postfix);
}

private static double evaluation(/*@NonNull*/ String postfix) {
    final Stack<Double> stack = new Stack<>();
    for (String s : postfix.trim().split("\\s+")) {
        if ("+-*/%^".contains(s)) {
            final Double y = stack.pop();
            final Double x = stack.pop();

            final BiFunction<Double, Double, Double> function =
OPERATORS_FUNCTION.get(s);
            if (function == null) {
                throw new IllegalStateException("Unknown operator '" + s +
"'");
            } else {
                stack.push(function.apply(x, y));
            }
        } else {
            stack.push(Double.parseDouble(s));
        }
    }
    return stack.peek();
}

/*@NonNull*/
private static String infixToPostfix(/*@NonNull*/ String infix) {
    final StringBuilder postfix = new StringBuilder();
    final Stack<String> stack = new Stack<>();

    final String refined = refine(infix);
    // System.out.println("Refined: " + refined);
    for (String elem : refined.split("\\s+")) {
        if ("+-*/%^".contains(elem)) {
            while (!stack.isEmpty())

```



```
        && priorityOf(elem) <= priorityOf(stack.peek())) {
            postfix.append(stack.pop())
                .append(' ');
        }
        stack.push(elem);
    } else if ("(".equals(elem)) {
        stack.push(elem);
    } else if (")".equals(elem)) {
        while (!"(".equals(stack.peek())) {
            postfix.append(stack.pop())
                .append(' ');
        }
        stack.pop();
    } else {
        postfix.append(elem).append(' ');
    }
}

while (!stack.isEmpty()) {
    postfix.append(stack.pop()).append(' ');
}

return postfix.toString();
}

private static int priorityOf(/*@NonNull*/ String operator) {
    if ("^".equals(operator)) return 3;
    if ("*/".contains(operator)) return 2;
    if ("+-".contains(operator)) return 1;
    if ("()".contains(operator)) return 0;
    throw new IllegalStateException("Operator '" + operator + "' not
implement");
}
}
```

File Client.java

```
package com.hoc.thltn1.bai2;

import java.io.*;
import java.net.Socket;

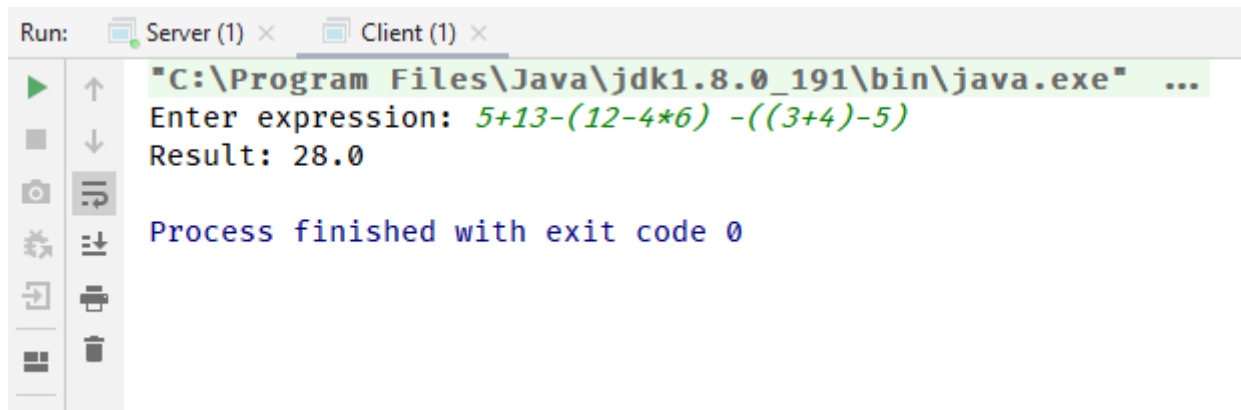
public class Client {
    public static void main(String[] args) throws IOException {
        final Socket socket = new Socket("localhost", 5000);

        try {
            final DataInputStream dataInputStream = new
DataInputStream(socket.getInputStream());
            final DataOutputStream dos = new
DataOutputStream(socket.getOutputStream());
            final BufferedReader bufferedReader = new BufferedReader(new
```

```
InputStreamReader(System.in))
    {
        System.out.print("Enter expression: ");
        final String expression = bufferedReader.readLine();
        dos.writeUTF(expression);

        System.out.println("Result: " + dataInputStream.readUTF());
    }
}
```

Triển khai và kết quả



Hình 2. Kết quả chạy chương trình bài 1.2

3. Bài tập 3

Xây dựng chương trình hội thoại chat room Client/Server hoạt động theo giao thức TCP/IP

- Chương trình Server mở cổng chờ nhận kết nối từ Client.
- Chương trình Client kết nối và thực hiện trao đổi với chương trình Server.

File Server.java

```
package com.hoc.thltm1.bai3;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Collections;
import java.util.LinkedHashSet;
```

```
import java.util.Scanner;
import java.util.Set;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Server {
    private static final Set<PrintWriter> writers =
        Collections.synchronizedSet(new LinkedHashSet<>());

    public static void main(String[] args) throws IOException {
        final ServerSocket serverSocket = new ServerSocket(5000);
        final ExecutorService threadPool =
            Executors.newFixedThreadPool(100);
        System.out.println(serverSocket);

        while (true) {
            final Socket socket = serverSocket.accept();
            threadPool.submit(new Handler(socket));
        }
    }

    private static class Handler implements Runnable {
        private final Socket socket;

        public Handler(Socket socket) {
            this.socket = socket;
        }

        @Override
        public void run() {
            System.out.println("" + socket.getInetAddress().getHostName() + "
connected");

            Scanner scanner = null;
            PrintWriter writer = null;

            try {
                scanner = new Scanner(socket.getInputStream());
                writer = new PrintWriter(socket.getOutputStream(), true);
                final PrintWriter finalWriter = writer;

                writers.add(writer);
                writers.forEach(e -> {
                    if (e == finalWriter) {
                        e.println(">>> You joined chat room. Type /quit to quit!");
                    } else {
                        e.println(">>> " + socket.getInetAddress().getHostAddress()
+ " joined chat room");
                    }
                });

                while (true) {
                    final String s = scanner.nextLine();
                    System.out.println(s);
                }
            }
        }
    }
}
```

```

        if (s.equalsIgnoreCase("/quit")) {
            return;
        }

        writers.forEach(e -> {
            if (e == finalWriter) {
                e.println("[MESSAGE] You: " + s);
            } else {
                e.println("[MESSAGE] " +
socket.getInetAddress().getHostAddress() + ": " + s);
            }
        });

    }

} catch (IOException e) {
    e.printStackTrace();
} finally {
    writers.remove(writer);
    writers.forEach(e -> e.println("<<< " +
socket.getInetAddress().getHostAddress() + " left chat room!"));

    if (scanner != null) {
        scanner.close();
    }
    if (writer != null) {
        writer.close();
    }
    try {
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
}
}
}

```

File Client.java

```

package com.hoc.thltn1.bai3;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

```

```
public class Client {
    private static class MainPanel extends JPanel {

        private JTextArea textArea;
        private JTextField textField;

        private final Socket socket;

        MainPanel() throws IOException {
            socket = new Socket("localhost", 5000);
            setupUI(new PrintWriter(socket.getOutputStream(), true));
            listen(new Scanner(socket.getInputStream()));
        }

        private void setupUI(PrintWriter out) {
            setPreferredSize(new Dimension(500, 400));
            setBackground(Color.WHITE);

            setLayout(null);
            textArea = new JTextArea();
            final JScrollPane scrollPane = new JScrollPane(textArea);
            scrollPane.setBounds(0, 0, 500, 350);
            this.add(scrollPane);

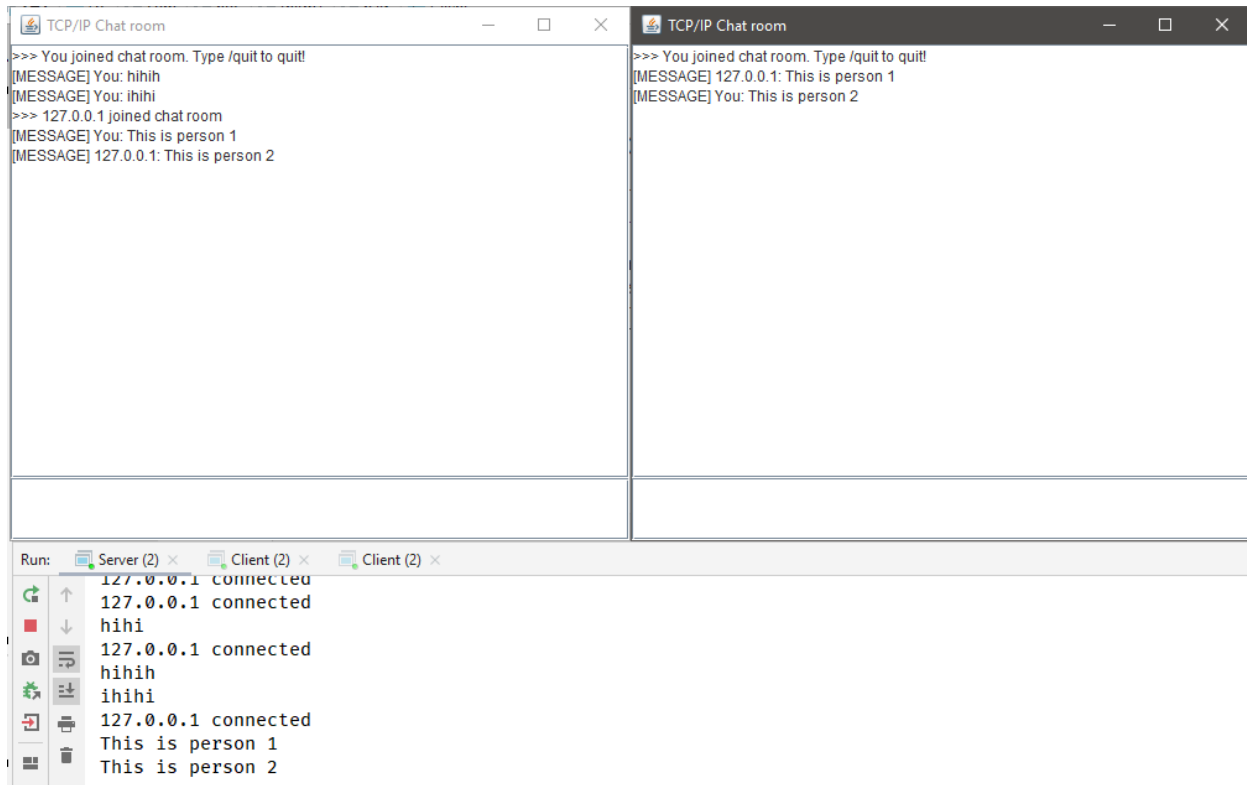
            textField = new JTextField("", 50);
            textField.setBounds(0, 350, 500, 50);
            this.add(textField);

            textField.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    out.println(textField.getText());
                    textField.setText("");
                }
            });
        }

        private void listen(Scanner in) {
            new Thread(() -> {
                while (in.hasNextLine()) {
                    String text = in.nextLine();
                    try {
                        System.out.println(text);
                        SwingUtilities.invokeLater(() -> textArea.append(text +
"\n"));
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }).start();
        }
    }
}
```

```
public static void main(String[] args) throws IOException {
    SwingUtilities.invokeLater(() -> {
        final JFrame frame = new JFrame("TCP/IP Chat room");
        try {
            frame.add(new MainPanel());
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setVisible(true);
    });
}
```

Triển khai và kết quả



Hình 3. Kết quả chạy chương trình bài 1.3

II. BÀI THỰC HÀNH SỐ 2

1. Bài tập 1

Xây dựng chương trình hội thoại Client/Server hoạt động theo giao thức UDP

- Chương trình Server mở cổng và chờ nhận kết nối từ Client.
- Client gửi một chuỗi ký tự đến Server. Server nhận và xử lý gửi trả về cho client các công việc:
 - Đổi chuỗi đã gửi thành chuỗi in hoa
 - Đổi chuỗi đã gửi thành chuỗi thường
 - Đếm số từ của chuỗi đã gửi

File Server.java

```
package com.hoc.thltn2.bai1;

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

public class Server {
    public static void main(String[] args) throws IOException {
        final DatagramSocket socket = new DatagramSocket(5000);

        while (true) {
            final byte[] buffer = new byte[1024];
            final DatagramPacket incoming = new DatagramPacket(buffer,
buffer.length);
            socket.receive(incoming);
            final String s = new String(incoming.getData(), 0,
incoming.getLength());

            final InetAddress address = incoming.getAddress();
            final int port = incoming.getPort();

            send(socket, s.toUpperCase(), address, port);
            send(socket, s.toLowerCase(), address, port);
            send(socket, String.valueOf(s.split("\\W+").length), address,
port);
        }

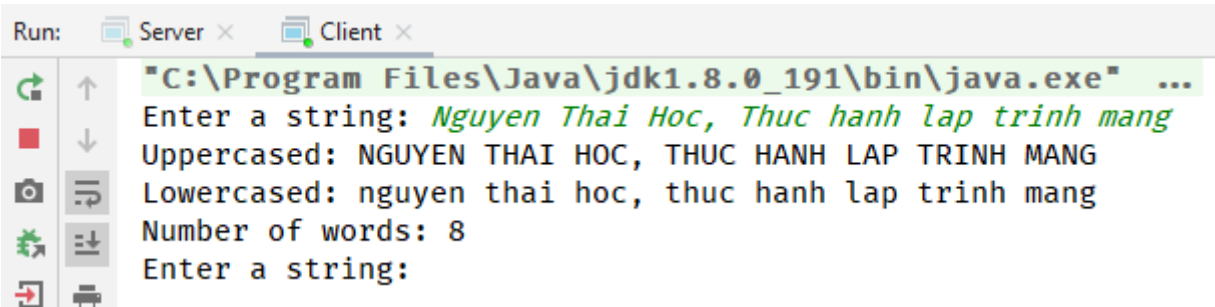
        private static void send(DatagramSocket socket, String s, InetAddress
address, int port) throws IOException {
            final DatagramPacket outSending = new DatagramPacket(
s.getBytes(),
0,
s.length(),
address,
port
);
            socket.send(outSending);
        }
    }
}
```

```
}  
}
```

File Client.java

```
package com.hoc.thltn2.bai1;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
  
public class Client {  
    public static void main(String[] args) throws IOException {  
        final DatagramSocket socket = new DatagramSocket();  
  
        while (true) {  
            final BufferedReader reader = new BufferedReader(new  
InputStreamReader(System.in));  
            System.out.print("Enter a string: ");  
            final String s = reader.readLine();  
  
            final byte[] bytes = s.getBytes();  
            final DatagramPacket sender = new DatagramPacket(  
                bytes,  
                bytes.length,  
                InetAddress.getByName("localhost"),  
                5000  
            );  
            socket.send(sender);  
  
            receive(socket, "Uppercased: ");  
            receive(socket, "Lowercased: ");  
            receive(socket, "Number of words: ");  
        }  
    }  
  
    private static void receive(DatagramSocket socket, String tag) throws  
IOException {  
        final byte[] buffer = new byte[1024];  
        final DatagramPacket receiver = new DatagramPacket(buffer,  
buffer.length);  
        socket.receive(receiver);  
        System.out.println(tag + new String(receiver.getData(), 0,  
receiver.getLength()));  
    }  
}
```

Triển khai và kết quả



Hình 4. Kết quả chạy chương trình bài 2.1

2. Bài tập 2

Xây dựng chương trình hội thoại Client/Server hoạt động theo giao thức UDP

- Chương trình Client cho phép nhập vào từ bàn phím một chuỗi biểu diễn một phép tính gồm các toán tử +, -, (,). Ví dụ: $5+13-(12-4*6) - ((3+4)-5)$
- Chương trình Server thực hiện tính toán và trả kết quả về cho Client.

File Server.java

```

package com.hoc.thltn2.bai2;

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

public class Server {
    public static void main(String[] args) throws IOException {
        final DatagramSocket socket = new DatagramSocket(5000);

        while (true) {
            final byte[] buffer = new byte[1024];
            final DatagramPacket incoming = new DatagramPacket(buffer,
            buffer.length);
            socket.receive(incoming);
            final String s = new String(incoming.getData(), 0,
            incoming.getLength());

            final InetAddress address = incoming.getAddress();
            final int port = incoming.getPort();

            try {
                final String result = String.valueOf(Eval.execute(s));

                final DatagramPacket outSending = new DatagramPacket(
  
```

```

        result.getBytes(),
        0,
        result.length(),
        address,
        port
    );
    socket.send(outSending);
} catch (Exception e) {
    final String error = e.toString();
    final DatagramPacket outSending = new DatagramPacket(
        error.getBytes(),
        0,
        error.length(),
        address,
        port
    );
    socket.send(outSending);
}
}
}
}

```

File Eval.java

```

package com.hoc.thltm2.bai2;

import java.util.HashMap;
import java.util.Map;
import java.util.Stack;
import java.util.function.BiFunction;

public class Eval {
    private static final Map<String, BiFunction<Double, Double, Double>>
    OPERATORS_FUNCTION;

    static {
        OPERATORS_FUNCTION = new HashMap<>();
        OPERATORS_FUNCTION.put("+", Double::sum);
        OPERATORS_FUNCTION.put("-", (x, y) -> x - y);
        OPERATORS_FUNCTION.put("*", (x, y) -> x * y);
        OPERATORS_FUNCTION.put("/", (x, y) -> x / y);
        OPERATORS_FUNCTION.put("%", (x, y) -> x % y);
        OPERATORS_FUNCTION.put("^", Math::pow);
    }

    public static void main(String[] args) {
        System.out.println(Eval.execute("5+13-(12-4*6) -((3+4)-5)"));
        System.out.println(Eval.execute("1+2+3+4"));
        System.out.println(Eval.execute("1 % 2"));
    }

    /*@NonNull*/

```

```

private static String refine(/*@NonNull*/ String infix) {
    final StringBuilder infixBuilder = new StringBuilder(infix);
    final int parenthesesDifferent = countChar(infixBuilder, '(')
        - countChar(infixBuilder, ')');
    if (parenthesesDifferent > 0) {
        infixBuilder.append(
            String.valueOf(new char[parenthesesDifferent])
                .replace('\0', ' '));
    } else if (parenthesesDifferent < 0 || !balancedParentheses(infix))
    {
        return "";
    }

    return infixBuilder.toString()
        .replaceAll("\\s+", " ")
        .replaceAll("([*/%^])-(\\d+(\\.\\d+)?)?", "$1(0-$2)")
        .replaceAll("([*/%^])-\\(", "$1(-1)*(")
        .replaceAll("\\)\\(", ")*(")
        .replaceAll("(\\d)\\(", "$1*(")
        .replaceAll("[+\\-*/%^()]", " $0 ")
        .replaceAll("\\d+(\\.\\d+)?", "$0 ")
        .trim();
}

private static boolean balancedParentheses(/*@NonNull*/ String s) {
    final Stack<Character> stack = new Stack<>();
    for (char c : s.toCharArray()) {
        if (c == '(') {
            stack.push(c);
        } else if (c == ')') {
            if (stack.isEmpty()) {
                return false;
            }
            stack.pop();
        }
    }
    return stack.isEmpty();
}

private static int countChar(/*@NonNull*/ CharSequence s, char ch) {
    return Math.toIntExact(s.chars().filter(c -> c == ch).count());
}

public static double execute(/*@NonNull*/ String infix) {
    final String postfix = infixToPostfix(infix);
    //System.out.println("Postfix: " + postfix);
    return evaluation(postfix);
}

private static double evaluation(/*@NonNull*/ String postfix) {
    final Stack<Double> stack = new Stack<>();
    for (String s : postfix.trim().split("\\s+")) {
        if ("+-*/%^".contains(s)) {

```

```

        final Double y = stack.pop();
        final Double x = stack.pop();

        final BiFunction<Double, Double, Double> function =
OPERATORS_FUNCTION.get(s);
        if (function == null) {
            throw new IllegalStateException("Unknown operator '" + s +
""");
        } else {
            stack.push(function.apply(x, y));
        }
        } else {
            stack.push(Double.parseDouble(s));
        }
    }
    return stack.peek();
}

/*@NonNull*/
private static String infixToPostfix(/*@NonNull*/ String infix) {
    final StringBuilder postfix = new StringBuilder();
    final Stack<String> stack = new Stack<>();

    final String refined = refine(infix);
    // System.out.println("Refined: " + refined);
    for (String elem : refined.split("\\s+")) {
        if ("+-*/%^".contains(elem)) {
            while (!stack.isEmpty()
                && priorityOf(elem) <= priorityOf(stack.peek())) {
                postfix.append(stack.pop())
                    .append(' ');
            }
            stack.push(elem);
        } else if ("(".equals(elem)) {
            stack.push(elem);
        } else if (")".equals(elem)) {
            while (!"(".equals(stack.peek())) {
                postfix.append(stack.pop())
                    .append(' ');
            }
            stack.pop();
        } else {
            postfix.append(elem).append(' ');
        }
    }

    while (!stack.isEmpty()) {
        postfix.append(stack.pop()).append(' ');
    }

    return postfix.toString();
}

private static int priorityOf(/*@NonNull*/ String operator) {

```

```
        if ("^".equals(operator)) return 3;
        if ("*/%".contains(operator)) return 2;
        if ("+-".contains(operator)) return 1;
        if ("(").contains(operator)) return 0;
        throw new IllegalStateException("Operator '" + operator + "' not
implement");
    }
}
```

File Client.java

```
package com.hoc.thltn2.bai2;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class Client {
    public static void main(String[] args) throws IOException {
        final DatagramSocket socket = new DatagramSocket();

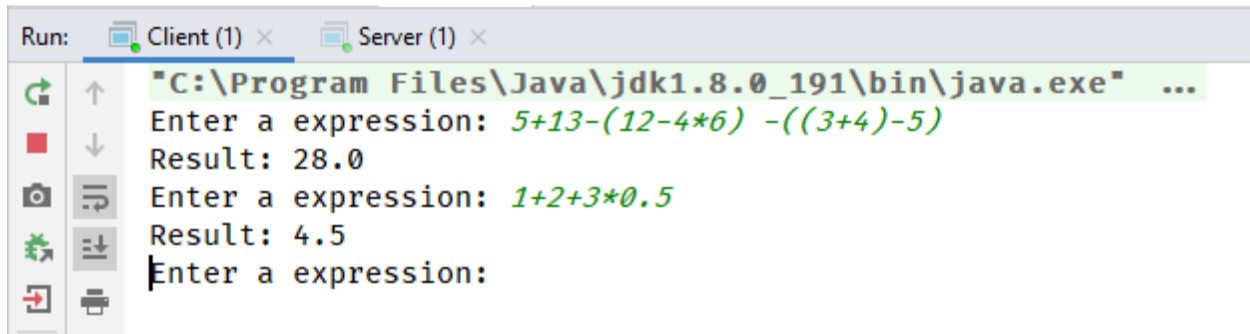
        while (true) {
            final BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Enter a expression: ");
            final String s = reader.readLine();

            final byte[] bytes = s.getBytes();
            final DatagramPacket sender = new DatagramPacket(bytes,
bytes.length,
                InetAddress.getByName("localhost"), 5000);
            socket.send(sender);

            final byte[] buffer = new byte[1024];
            final DatagramPacket receiver = new DatagramPacket(buffer,
buffer.length);
            socket.receive(receiver);
            final String result = new String(receiver.getData(), 0,
receiver.getLength());

            System.out.println("Result: " + result);
        }
    }
}
```

Triển khai và kết quả



Hình 5. Kết quả chạy chương trình bài 2.2

3. Bài tập 3

Xây dựng chương trình hội thoại chat room Client/Server hoạt động theo giao thức UDP

- Chương trình Server mở cổng chờ nhận kết nối từ Client.
- Chương trình Client kết nối và thực hiện trao đổi với chương trình Server.

File Server.java

```
package com.hoc.thltn2.bai3;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Collections;
import java.util.LinkedHashSet;
import java.util.Objects;
import java.util.Set;

public class Server {
    private static final class InetAddressAndPort {
        final InetAddress address;
        final int port;

        InetAddressAndPort(InetAddress address, int port) {
            this.address = address;
            this.port = port;
        }
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        InetAddressAndPort that = (InetAddressAndPort) o;
```

```

        return port == that.port &&
            Objects.equals(address, that.address);
    }

    @Override
    public int hashCode() {
        return Objects.hash(address, port);
    }

    @Override
    public String toString() {
        return "InetAddressAndPort{" +
            "address=" + address +
            ", port=" + port +
            '}';
    }
}

private static final Set<InetAddressAndPort> inetAddressAndPorts =
    Collections.synchronizedSet(new LinkedHashSet<>());

public static void main(String[] args) throws SocketException {
    final DatagramSocket socket = new DatagramSocket(5000);
    while (true) {
        try {
            handle(socket);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

private static void handle(DatagramSocket socket) throws Exception {
    final byte[] buffer = new byte[1024];
    final DatagramPacket packet = new DatagramPacket(
        buffer,
        0,
        buffer.length
    );
    socket.receive(packet);
    final String message = new String(packet.getData(), 0,
        packet.getLength());

    final InetAddress currentAddr = packet.getAddress();
    final int currentPort = packet.getPort();
    final InetAddressAndPort elem = new InetAddressAndPort(currentAddr,
        currentPort);
    System.out.println("Receive message=" + message + " from " + elem);

    if (message.equalsIgnoreCase("/join")) {
        inetAddressAndPorts.add(elem);

        inetAddressAndPorts.forEach(e -> {

```

```

try {
    final byte[] bytes;

    if (e.equals(elem)) {
        bytes = ">>> You joined chat room".getBytes();
    } else {
        bytes = String.format(">>> %s:%s joined chat room",
currentAddr.getHostName(), currentPort).getBytes();
    }

    socket.send(new DatagramPacket(bytes, 0, bytes.length,
e.address, e.port));
} catch (IOException ex) {
    ex.printStackTrace();
}
});
} else if (message.equalsIgnoreCase("/quit")) {
    inetAddressAndPorts.forEach(e -> {
        try {
            final byte[] bytes;

            if (e.equals(elem)) {
                bytes = "<<< You left chat room".getBytes();
            } else {
                bytes = String.format("<<< %s:%s left chat room",
currentAddr.getHostName(), currentPort).getBytes();
            }

            socket.send(new DatagramPacket(bytes, 0, bytes.length,
e.address, e.port));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    });
    inetAddressAndPorts.remove(elem);
} else {
    inetAddressAndPorts.forEach(e -> {
        try {
            final byte[] bytes;

            if (e.equals(elem)) {
                bytes = String.format("[MESSAGE] You: %s",
message).getBytes();
            } else {
                bytes = String.format("[MESSAGE] %s:%s: %s",
currentAddr.getHostName(), currentPort, message).getBytes();
            }

            socket.send(new DatagramPacket(bytes, 0, bytes.length,
e.address, e.port));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    });
}
}

```



```
    });
  }
}
```

File Client.java

```
package com.hoc.thltn2.bai3;

import javax.swing.*;
import java.awt.*;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Client {
    private static class MainPanel extends JPanel {

        private JTextArea textArea;
        private JTextField textField;

        private final DatagramSocket socket;
        private final ExecutorService executor =
            Executors.newSingleThreadExecutor();

        MainPanel() throws IOException {
            socket = new DatagramSocket();
            setupUI();
            listen();
        }

        private void setupUI() {
            setPreferredSize(new Dimension(500, 400));
            setBackground(Color.WHITE);

            setLayout(null);
            textArea = new JTextArea();
            final JScrollPane scrollPane = new JScrollPane(textArea);
            scrollPane.setBounds(0, 0, 500, 350);
            this.add(scrollPane);

            textField = new JTextField("", 50);
            textField.setBounds(0, 350, 500, 50);
            this.add(textField);

            textField.addActionListener(e -> {
                executor.submit(() -> {
                    try {
                        final byte[] bytes = textField.getText().getBytes();
                        final DatagramPacket p = new DatagramPacket(bytes, 0,
```

```

bytes.length, InetAddress.getByName("localhost"), 5000);
    socket.send(p);
    SwingUtilities.invokeLater(() -> textField.setText(""));
} catch (Exception ex) {
    ex.printStackTrace();
}
});
});
}

private void listen() {
    new Thread(() -> {
        while (true) {
            try {
                final byte[] buffer = new byte[1024];
                final DatagramPacket p = new DatagramPacket(buffer, 0,
buffer.length);
                socket.receive(p);
                final String text = new String(p.getData(), 0,
p.getLength());

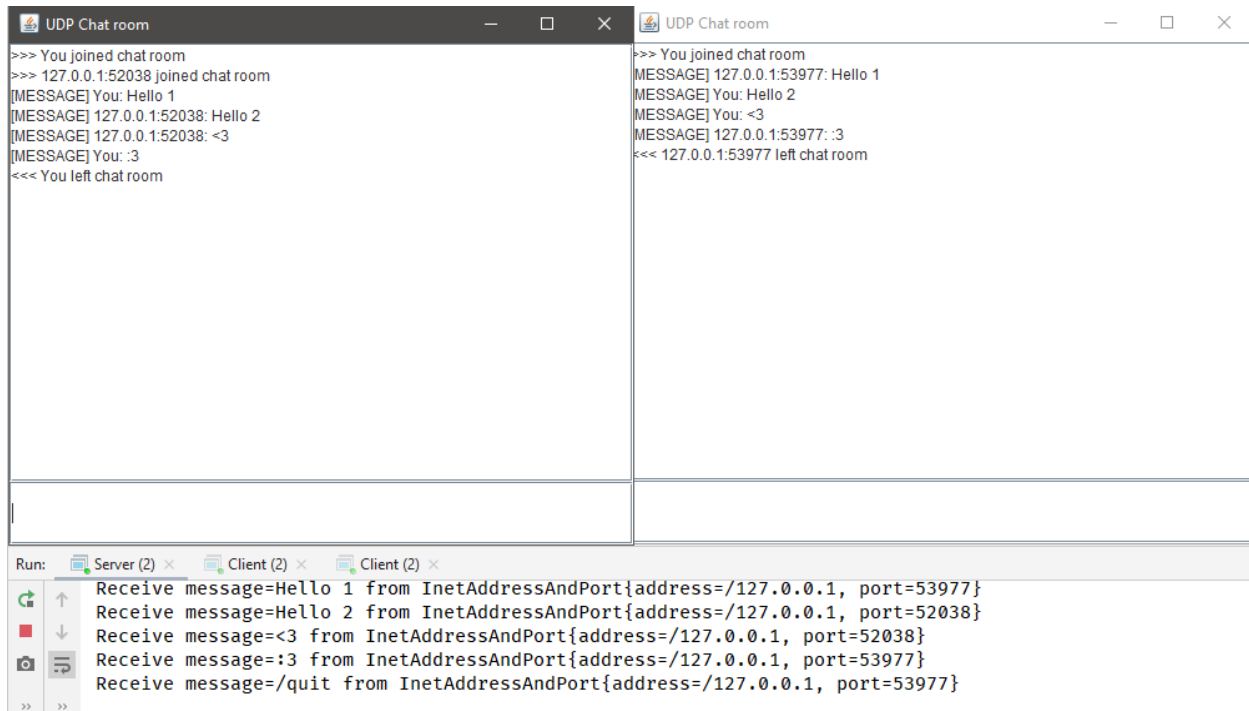
                System.out.println(text);
                SwingUtilities.invokeLater(() -> textArea.append(text +
"\n"));
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }).start();
}

}

public static void main(String[] args) throws IOException {
    SwingUtilities.invokeLater(() -> {
        final JFrame frame = new JFrame("UDP Chat room");
        try {
            frame.add(new MainPanel());
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
        frame.pack();
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setVisible(true);
    });
}
}

```

Triển khai và kết quả



Hình 6. Kết quả chạy chương trình bài 2.3

III. BÀI THỰC HÀNH SỐ 3

Thiết kế website quản lý việc cần làm to-do sử dụng JSP/Servlet với các trang:

- Đăng nhập vào hệ thống
- Đăng kí vào hệ thống
- Xem danh sách todo
- Thêm todo
- Xóa todo
- Chỉnh sửa todo

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/jsp/servlet_war_exploded/login.jsp'. The page title is 'Login'. Below the title, there is a green horizontal bar. The form contains two input fields: 'User Name:' with the value 'user01' and 'Password:' with masked characters '*****'. A blue 'Submit' button is located below the password field. The browser's taskbar at the bottom shows various application icons and the system clock indicating 6:36 PM on 12/21/2019.

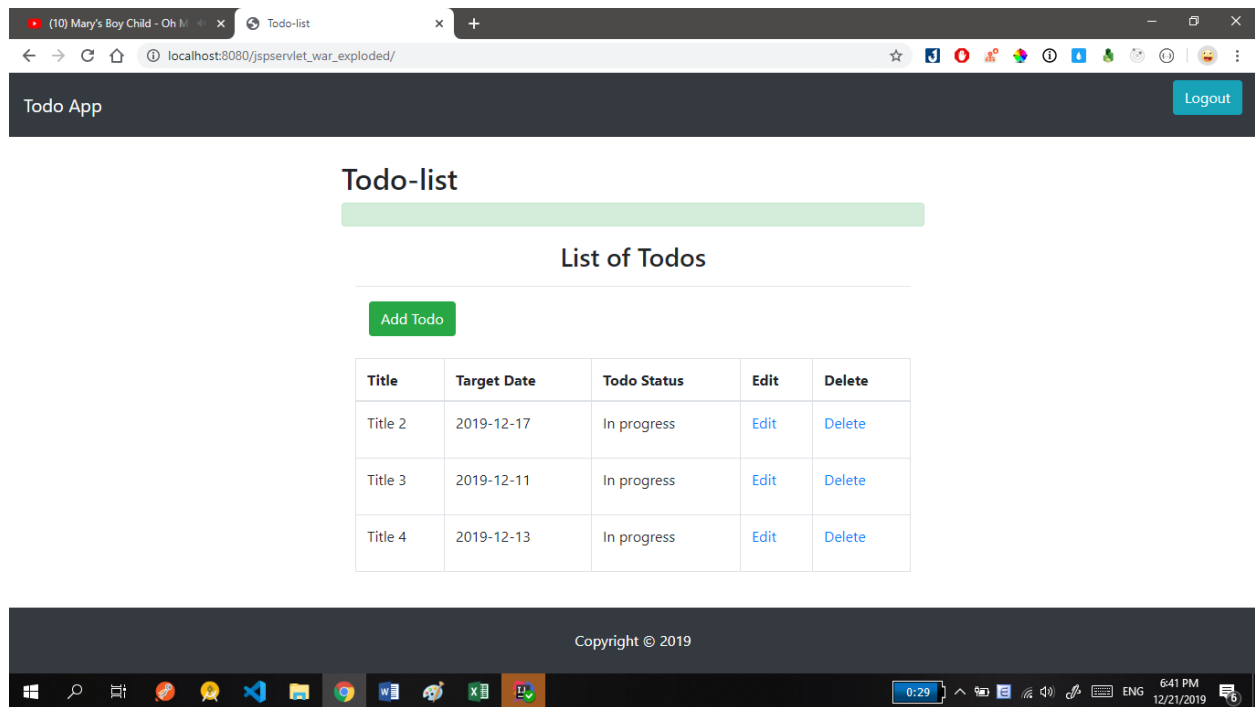
Copyright © 2019

Hình 7. Giao diện đăng nhập

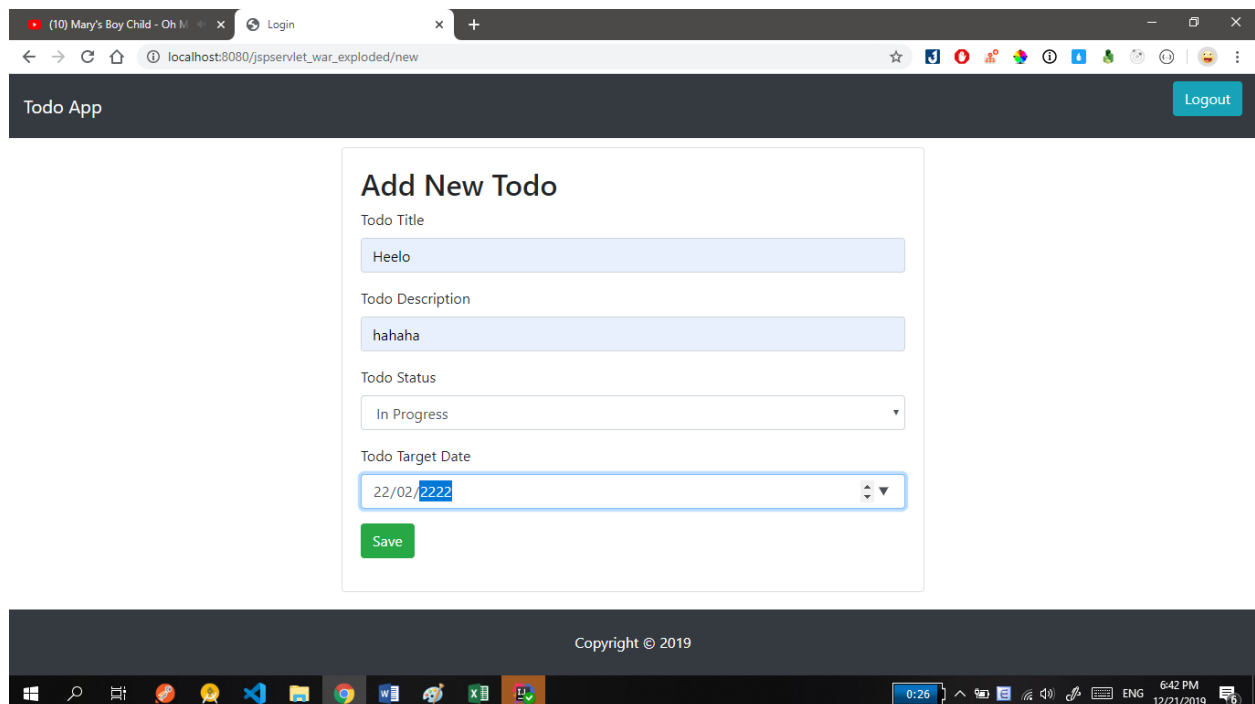
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/jsp/servlet_war_exploded/register.jsp'. The page title is 'Register'. Below the title, there is a green horizontal bar. The form is titled 'User Register Form' and contains four input fields: 'First Name:' with the value 'Nguyễn', 'Last Name:' with the value 'Thái Học', 'User Name:' with the value 'hoc081098@gmail.com', and 'Password:' with masked characters '*****'. A blue 'Submit' button is located below the password field. The browser's taskbar at the bottom shows various application icons and the system clock indicating 6:37 PM on 12/21/2019.

Copyright © 2019

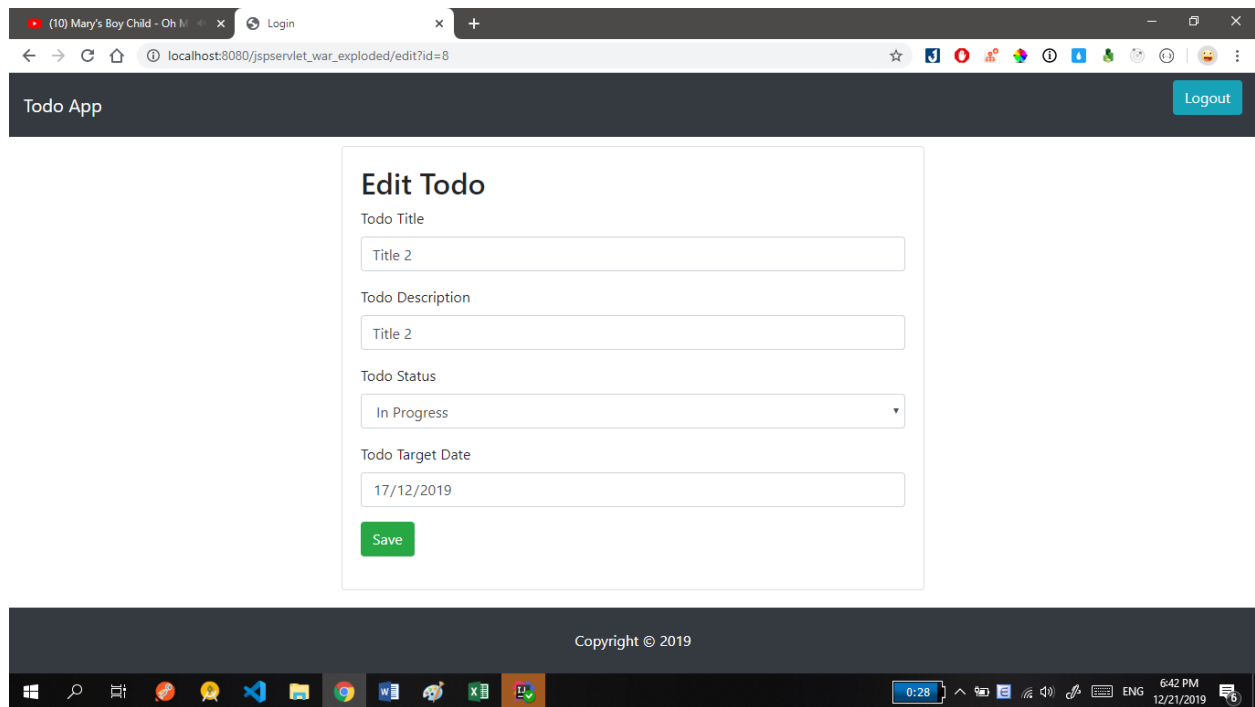
Hình 8. Giao diện đăng kí



Hình 9. Giao diện danh sách todo



Hình 10. Giao diện thêm todo



Hình 11. Giao diện chỉnh sửa todo