2014469 – Trần Ngọc Hoàng Long

1. Giao diện

public class StudentDAO {

public List<Student> getStudents() throws SQLException {

DataHelper dataHelper = DataHelper.getInstance();

String sql = "Select \* from [dbo].[SinhVien]";

Connection conn = dataHelper.getConnection();

List<Student> students = new ArrayList<>();

try {

Statement selectStatement = conn.createStatement();

ResultSet result = selectStatement.executeQuery(sql);

while (result.next()) {

String maSv = result.getString("masv");

String ten = result.getString("hoten");

String email = result.getString("email");

String sodt = result.getString("sodt");

int gioiTinh = result.getInt("gioitinh");

String diaChi = result.getString("diachi");

Student student = new Student(maSv, ten, email, sodt, diaChi, gioiTinh);

students.add(student);

}

} catch (SQLException ex) {

ex.printStackTrace();

} finally {

dataHelper.closeConnection();

}

return students;

}

public boolean insert(Student std) throws Exception {

DataHelper dataHelper = DataHelper.getInstance();

String sql = "insert into [dbo].[SinhVien](masv, hoten, email, sodt, gioitinh, diachi) values (?, ?, ? ,?,?,?)";

try {

Connection conn = dataHelper.getConnection();

PreparedStatement preparedStatement = conn.prepareStatement(sql);

preparedStatement.setString(1, std.getMaSV());

preparedStatement.setString(2, std.getHoTen());

preparedStatement.setString(3, std.getEmail());

preparedStatement.setString(4, std.getSoDT());

preparedStatement.setInt(5, std.getGioiTinh());

preparedStatement.setString(6, std.getDiaChi());

return preparedStatement.executeUpdate() > 0;

} catch (SQLException ex) {

throw ex;

} finally {

dataHelper.closeConnection();

}

}

public boolean delete(Student std) throws SQLException {

DataHelper dataHelper = DataHelper.getInstance();

String sql = "delete from [dbo].[SinhVien] where [masv] = ?";

try {

Connection conn = dataHelper.getConnection();

PreparedStatement preparedStatement = conn.prepareStatement(sql);

preparedStatement.setString(1, std.getMaSV());

return preparedStatement.executeUpdate() > 0;

} catch (SQLException ex) {

throw ex;

} finally {

dataHelper.closeConnection();

}

}

public boolean edit(Student std) throws Exception {

DataHelper dataHelper = DataHelper.getInstance();

String sql = """

UPDATE [dbo].[SinhVien]

SET [masv] = ?

,[hoten] = ?

,[email] =?

,[sodt] = ?

,[gioitinh] =?

,[diachi] = ?

WHERE [masv] = ?

""";

try {

Connection conn = dataHelper.getConnection();

PreparedStatement preparedStatement = conn.prepareStatement(sql);

preparedStatement.setString(1, std.getMaSV());

preparedStatement.setString(2, std.getHoTen());

preparedStatement.setString(3, std.getEmail());

preparedStatement.setString(4, std.getSoDT());

preparedStatement.setInt(5, std.getGioiTinh());

preparedStatement.setString(6, std.getDiaChi());

preparedStatement.setString(7, std.getMaSV());

return preparedStatement.executeUpdate() > 0;

} catch (SQLException ex) {

throw ex;

} finally {

dataHelper.closeConnection();

}

}

}

public class StudentModelTable extends AbstractTableModel {

private final String[] columnNames = {"Mã sinh viên", "Tên sinh viên", "Email", "Số điện thoại", "Giới tính", "Địa chỉ"};

private List<Student> students;

public StudentModelTable() {

students = new ArrayList<>();

}

public void setData(List<Student> students) {

this.students = students;

fireTableDataChanged();

}

@Override

public int getRowCount() {

return students.size();

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public String getColumnName(int column) {

return columnNames[column];

}

@Override

public Object getValueAt(int rowIndex, int columnIndex) {

Student student = students.get(rowIndex);

switch (columnIndex) {

case 0 -> {

return student.getMaSV();

}

case 1 -> {

return student.getHoTen();

}

case 2 -> {

return student.getEmail();

}

case 3 -> {

return student.getSoDT();

}

case 4 -> {

return student.getDiaChi();

}

case 5 -> {

return student.getGioiTinh();

}

default ->

throw new AssertionError();

}

}

}

public class StudentController {

private FormStudent studentView;

private StudentDAO studentDAO;

public StudentController(FormStudent studentView) {

this.studentDAO = new StudentDAO();

this.studentView = studentView;

}

public void showStudentView() {

List<Student> students = null;

try {

students = studentDAO.getStudents();

} catch (Exception ex) {

System.out.println(ex);

}

studentView.showStudents(students);

studentView.setVisible(true);

studentView.addTableSelectionListener(new TableSelectionListener());

studentView.addAddStudentListener(new AddStudentListener());

studentView.addEditStudentListener(new EditStudentListener());

studentView.addRemoveStudentListener(new DeleteStudentListener());

}

class AddStudentListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

Student std = studentView.getStudentData();

if (std != null) {

boolean sucess = false;

try {

sucess = studentDAO.insert(std);

} catch (Exception ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

if (sucess) {

try {

studentView.showStudents(studentDAO.getStudents());

} catch (SQLException ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

studentView.showMessag("Them thanh cong");

} else {

studentView.showMessag("Them that bai");

}

}

}

}

class EditStudentListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

Student std = studentView.getStudentData();

if (std != null) {

boolean sucess = false;

try {

sucess = studentDAO.edit(std);

} catch (Exception ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

if (sucess) {

try {

studentView.showStudents(studentDAO.getStudents());

} catch (SQLException ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

studentView.showMessag("Chinh sua thanh cong");

} else {

studentView.showMessag("Chinh sua that bai");

}

}

}

}

class DeleteStudentListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

Student std = studentView.getStudentData();

if (std != null) {

boolean sucess = false;

try {

sucess = studentDAO.delete(std);

} catch (SQLException ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

if (sucess) {

try {

studentView.showStudents(studentDAO.getStudents());

} catch (SQLException ex) {

Logger.getLogger(StudentController.class.getName()).log(Level.SEVERE, null, ex);

}

studentView.showMessag("Xoa thanh cong");

} else {

studentView.showMessag("Xoa that bai");

}

}

}

}

class TableSelectionListener implements ListSelectionListener {

@Override

public void valueChanged(ListSelectionEvent e) {

studentView.fillInputForm();

}

}

}

public class DataHelper {

private static DataHelper instanceDataHelper;

private DataHelper() {

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

} catch (ClassNotFoundException e) {

System.out.println("Can't load driver");

}

}

public static DataHelper getInstance() throws SQLException {

if (instanceDataHelper == null) {

instanceDataHelper = new DataHelper();

} else if (instanceDataHelper.getConnection().isClosed()) {

instanceDataHelper = new DataHelper();

}

return instanceDataHelper;

}

public Connection getConnection() {

String connUrl = "jdbc:sqlserver://localhost:1433;encrypt=false;database=Student;integeratedSecurity=false;";

String user = "Group5";

String pass = "123";

System.out.println("Dang ket noi");

try {

Connection conn = DriverManager.getConnection(connUrl, user, pass);

System.out.println("Database connected");

return conn;

} catch (SQLException ex) {

System.out.println("Can't connect to database");

throw new RuntimeException(ex);

}

}

public void closeConnection() {

try {

instanceDataHelper.getConnection().close();

System.out.println("Database closed");

} catch (SQLException ex) {

System.out.println("Can't close connection");

throw new RuntimeException(ex);

}

}

}

StudentModelTable studentModelTable;

public FormStudent() {

studentModelTable = new StudentModelTable();

initComponents();

}

public void showStudents(List<Student> students) {

studentModelTable.setData(students);

if (jtableStudent != null && jtableStudent.getRowCount() > 0) {

jtableStudent.getSelectionModel().setSelectionInterval(0, 0);

}

}

public void addTableSelectionListener(ListSelectionListener listener) {

jtableStudent.getSelectionModel().addListSelectionListener(listener);

}

public void addAddStudentListener(ActionListener listener) {

btnSave.addActionListener(listener);

}

public void addRemoveStudentListener(ActionListener listener) {

btnDelete.addActionListener(listener);

}

public void addEditStudentListener(ActionListener listener) {

btnEdit.addActionListener(listener);

}

public Student getStudentData() {

String masv = txtMaSV.getText();

String ten = txtHoTen.getText();

String email = txtEmail.getText();

String soDt = txtSoDT.getText();

String diaChi = txtDiaChi.getText();

int gioiTinh = rdNam.isSelected() ? 1 : 0;

return new Student(masv, ten, email, soDt, diaChi, gioiTinh);

}

public void fillInputForm() {

int row = jtableStudent.getSelectedRow();

if (row >= 0) {

txtMaSV.setText(jtableStudent.getValueAt(row, 0).toString());

txtHoTen.setText(jtableStudent.getValueAt(row, 1).toString());

txtEmail.setText(jtableStudent.getValueAt(row, 2).toString());

txtSoDT.setText(jtableStudent.getValueAt(row, 3).toString());

txtDiaChi.setText(jtableStudent.getValueAt(row, 4).toString());

if (Integer.parseInt(jtableStudent.getValueAt(row, 5).toString()) == 1) {

rdNam.setSelected(true);

} else {

rdNu.setSelected(true);

}

}

}

public void showMessag(String msg) {

JOptionPane.showMessageDialog(this, msg);

}

public class OnTap {

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

FormStudent view = new FormStudent();

StudentController controller = new StudentController(view);

controller.showStudentView();

});

}

}

<dependencies>

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<version>8.2.2.jre13</version>

</dependency>

</dependencies>

private void filter(String query) {

TableRowSorter<QuanAoTableModel> tr = new TableRowSorter<>(quanAoTableModel);

jtableQA.setRowSorter(tr);

tr.setRowFilter(RowFilter.regexFilter("(?i)" + Pattern.quote(query)));

}

private void txtSearchKeyReleased(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_txtSearchKeyReleased

String query = txtSearch.getText().trim().toLowerCase();

filter(query);

}

String xuatXu = (String) cboXuatXu.getSelectedItem();

public void setValuesStringToComboBox(List<String> values, DefaultComboBoxModel modelCombo, JComboBox comboBox) {

modelCombo.removeAllElements();

for (String value : values) {

System.out.println(value);

modelCombo.addElement(value);

}

comboBox = new JComboBox(modelCombo);

if (comboBox.getItemCount() > 0) {

comboBox.setSelectedIndex(0);

}

}

1. Socket

public class TCPServer {

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(6787);

System.out.println("Dang lang nghe...");

while (true) {

Socket socket = serverSocket.accept();

// doc du lieu nhan duoc

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

// ghi du lieu doc duoc tu phia client

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

String line = reader.readLine();

System.out.println("Nhan: " + line);

System.out.println("Gui" + line.toUpperCase());

// gui du lieu den phia client

out.print(line.toUpperCase());

// xoa bo dem di

out.flush();

// dong ket noi

reader.close();

out.close();

socket.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class TCPClient {

public static void main(String[] args) {

try {

Socket socket = new Socket("127.0.0.1", 6787);

System.out.println("Dang ket noi...");

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

// ghi du lieu doc duoc

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

Scanner s = new Scanner(System.in);

System.out.print("Nhap du lieu: ");

String st = s.nextLine();

System.out.println("Gui" + st);

out.println(st);

out.flush();

// doc du lieu nhan duoc

String line = reader.readLine();

System.out.println("Nhan: " + line);

// xoa thu vien di

out.flush();

// dong ket noi

reader.close();

out.close();

socket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class UDPServer {

public static void main(String[] args) {

try {

DatagramSocket datagramSocket = new DatagramSocket(8078);

byte[] read = new byte[1024];

byte[] write = null;

System.out.println("Dang cho ket noi...");

while (true) {

DatagramPacket receive = new DatagramPacket(read, read.length);

datagramSocket.receive(receive);

String line = new String(receive.getData());

write = line.toUpperCase().getBytes();

InetAddress host = receive.getAddress();

int port = receive.getPort();

DatagramPacket send = new DatagramPacket(write, write.length, host, port);

System.out.println("Gui di: " + line.toUpperCase());

datagramSocket.send(send);

// close

// datagramSocket.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class UDPClient {

public static void main(String[] args) {

try {

DatagramSocket datagramSocket = new DatagramSocket();

byte[] read = new byte[1024];

byte[] write = null;

Scanner s = new Scanner(System.in);

System.out.println("Nhap du lieu: ");

String st = s.nextLine();

System.out.println("Dang ket noi...");

write = st.getBytes();

InetAddress host;

host = InetAddress.getByName("localhost");

int port = 8078;

DatagramPacket send = new DatagramPacket(write, write.length, host, port);

System.out.println("Gui di: " + st);

datagramSocket.send(send);

DatagramPacket receive = new DatagramPacket(read, read.length);

datagramSocket.receive(receive);

String line = new String(receive.getData());

System.out.println("Nhan: " + line);

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class ServerThread implements Runnable {

Socket client;

JTextArea txtClient;

ServerThread(Socket client, JTextArea txtClient) {

this.client = client;

this.txtClient = txtClient;

}

@Override

public void run() {

try {

BufferedReader reader = new BufferedReader(new InputStreamReader(client.getInputStream()));

String line;

line = reader.readLine();

txtClient.append(line);

} catch (Exception e) {

e.printStackTrace();

}

}

}

public class ClientThread implements Runnable {

Socket client;

JTextArea txtServer;

ClientThread(Socket client, JTextArea txtServer) {

this.client = client;

this.txtServer = txtServer;

}

@Override

public void run() {

try {

BufferedReader reader = new BufferedReader(new InputStreamReader(client.getInputStream()));

String line = reader.readLine();

System.out.println("Nhan: " + line);

txtServer.append(line);

} catch (Exception e) {

}

}

}

public class server extends javax.swing.JFrame {

ServerSocket serverSocket = null;

Socket client = null;

OutputStream out;

PrintStream ps;

int port;

/\*\*

\* Creates new form server

\*/

public server() {

initComponents();

this.setLocationRelativeTo(null);

btnSend.setEnabled(false);

}

private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {

try {

port = Integer.parseInt(txtPort.getText());

serverSocket = new ServerSocket(port);

client = serverSocket.accept();

btnStart.setEnabled(false);

btnSend.setEnabled(true);

out = client.getOutputStream();

ps = new PrintStream(out);

Thread t = new Thread(new ServerThread(client, txtClient));

t.start();

} catch (Exception e) {

e.printStackTrace();

}

}

private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {

String s = txtSend.getText().trim();

txtClient.append("\n" + s);

ps.println(s);

}

public class client extends javax.swing.JFrame {

int port;

Socket client;

OutputStream out;

PrintStream ps;

/\*\*

\* Creates new form client

\*/

public client() {

initComponents();

this.setLocationRelativeTo(null);

btnSend.setEnabled(false);

}

private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try {

port = Integer.parseInt(txtPort.getText().trim());

client = new Socket("127.0.0.1", port);

out = client.getOutputStream();

ps = new PrintStream(out);

btnConnect.setEnabled(false);

btnSend.setEnabled(true);

Thread t = new Thread(new ClientThread(client, txtServer));

t.start();

} catch (Exception e) {

e.printStackTrace();

}

}

private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String s = txtSend.getText().trim();

txtServer.append("\n" + s);

ps.println(s);

}

1. OOP

@Override

public String toString() {

String format = "%-12s | %-,29fđ | %-20s | \n";

return String.format(format, accountNumber, balance, isPremium() ? "Prenium" : "No Prenium");

}

public static boolean checkStringIsNumberic(String strNum) {

if (strNum == null) {

return false;

}

try {

Double.parseDouble(strNum);

} catch (NumberFormatException ex) {

return false;

}

return true;

}

private static final Scanner scanner = new Scanner(System.in);

cccd = scanner.nextLine();

public SavingAccount(String accountNumber, double balance) {

super(accountNumber, balance);

}

public static String formatMoney(double money) {

new String();

return String.format("%,.3fđ", money);

}