# Багийн гишүүд:

# B232270805 Г.Үлэмж B232270141 Э.Энхбат B232270051 Х.Шинэжаргал

# Мини төсөл: Sales System — Log4j v2 ашиглан лог хийх тайлан

## 1. Төслийн танилцуулга

Энэ мини төслийн зорилго нь хэрэглэгчийн худалдан авалт бүртгэх Sales System GUI-г Java ашиглан хөгжүүлж, түүний үйл явц, алдааг **Log4j v2** ашиглан бүртгэх, логийг inventory.log файл болон консолд хадгалах явдал юм. Энэхүү систем нь хэрэглэгчийн мэдээлэл авч, бүтээгдэхүүн захиалгыг удирдаж, нийт үнийг тооцоолж, захиалгыг хадгалах боломжтой. GUI нь Java Swing дээр суурилсан бөгөөд төслийн бүтцийг Maven ашиглан зохион байгуулсан.

## 2. Гол онцлогууд:

Бүтээгдэхүүн жагсаалт (Mouse, Keyboard, Laptop)

Захиалга үүсгэх, нийт үнийг тооцоолох

Захиалгыг Desktop дээр .txt файл болгон хадгалах

Log4j2 ашиглан үйлдэл бүрийг inventory.log файлд бичих

System.out устгаж, зөвхөн logger.info/error ашигласан

JUnit тестээр Product классын ажиллах чадварыг шалгасан

## 3. Төслийн шаардлагын тодорхойлолт

Систем нь хэрэглэгчийн нэр, и-мэйл, утсыг авах бөгөөд тухайн хэрэглэгчийн нэр дээр бүтээгдэхүүн захиалдаг. ComboBox дээрээс бараа сонгож, тоо оруулан захиалгад нэмдэг. Тус бүрийн барааны үнэ, нийт дүн нь GUI-д харагдах ба "Save Order" товчоор захиалгыг хадгална. Захиалга бүрийг Log4j2 ашиглан лог файлд тэмдэглэж, алдаа гарвал logger.error(...) бичдэг.

Шаардлагын дагуу дараах нөхцөлүүд биелсэн:

Maven dependency ашигласан

log4j2.xml тохируулсан

Консол болон файл руу зэрэг лог бичсэн

GUI дээр логик зөв ажилласан

## 4. Кодын дизайн, бүтэц (UML)

**Product** – барааны нэр, үнэ, нөөц хадгална  
**Customer** – хэрэглэгчийн нэр, и-мэйл, утас хадгална  
**OrderItem** – бараа ба тоо хэмжээ  
**Order** – захиалгад бүртгэгдсэн бүх бараа болон нийт үнэ  
**SalesSystemGUI** – GUI-г хариуцна, лог болон файл хадгалалт багтсан

## 5. Сурсан зүйл, бэрхшээл, сайжруулалт

### Сурсан зүйл:

Log4j2-г зөв тохируулах (log4j2.xml, dependency)

GUI-д лог бүртгэх нь дебаг хийхэд хялбар болдог

JUnit ашиглан классын логикийг баталгаажуулах

### Бэрхшээл:

log4j2.xml буруу байршилд байвал лог үүсдэггүй

GUI эхлүүлэхэд зарим exception хяналт хэрэгтэй болсон

### Ирээдүйн сайжруулалт:

SQLite/MySQL өгөгдлийн сан руу захиалгыг хадгалах

Spring Boot ашиглан вэб хувилбар болгон өргөжүүлэх

## 6. Кодын баримтжуулалтын тайлбар

Бүх класст Javadoc-style тайлбар, toString() override хийсэн. GUI логикийг дотроо comment-оор тайлбарласан. Оролт шалгах, алдаа шидэх логик бүрэн баримтжуулсан.

**Customer.java**

package sales;

public class Customer {

    private String name;

    private String email;

    private String phone;

    public Customer(String name, String email, String phone) {

        if (name == null || name.trim().isEmpty()) {

            throw new IllegalArgumentException("Name cannot be empty.");

        }

        if (!email.matches("^[\\w.-]+@[\\w.-]+\\.[a-zA-Z]{2,6}$")) {

            throw new IllegalArgumentException("Invalid email format.");

        }

        if (!phone.matches("^\\d{8}$")) {

            throw new IllegalArgumentException("Phone number must be 8 digits.");

        }

        this.name = name.trim();

        this.email = email.trim();

        this.phone = phone.trim();

    }

    public String getName() {

        return name;

    }

    public String getEmail() {

        return email;

    }

    public String getPhone() {

        return phone;

    }

    @Override

    public String toString() {

        return String.format("Customer: %s\nEmail: %s\nPhone: %s", name, email, phone);

    }

}

**Log4j2.xml**

<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="WARN">

    <Properties>

        <Property name="logDir">logs</Property>

    </Properties>

    <Appenders>

        <Console name="Console" target="SYSTEM\_OUT">

            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>

        </Console>

        <File name="FileLogger"

              fileName="${logDir}/inventory.log"

              append="true"

              createOnDemand="true">

            <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1} - %m%n"/>

        </File>

    </Appenders>

    <Loggers>

        <Root level="info">

            <AppenderRef ref="Console"/>

            <AppenderRef ref="FileLogger"/>

        </Root>

    </Loggers>

</Configuration>

**Order.java**

package sales;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

public class Order {

    private static final Logger logger = LogManager.getLogger(Order.class);

    private Customer customer;

    private List<OrderItem> items;

    public Order(Customer customer) {

        if (customer == null) {

            throw new IllegalArgumentException("Customer cannot be null.");

        }

        this.customer = customer;

        this.items = new ArrayList<>();

    }

    public boolean addItem(Product product, int quantity) {

        if (quantity > product.getStockQuantity()) {

            logger.warn("Not enough stock for product: " + product.getName());

            return false;

        }

        product.reduceStock(quantity);

        items.add(new OrderItem(product, quantity));

        logger.info("Added item to order: " + product.getName() + " x" + quantity);

        return true;

    }

    public List<OrderItem> getItems() {

        return Collections.unmodifiableList(items);

    }

    public double getTotalPrice() {

        return items.stream().mapToDouble(OrderItem::getSubtotal).sum();

    }

    public Customer getCustomer() {

        return customer;

    }

    public boolean isEmpty() {

        return items.isEmpty();

    }

    public void saveToFile() {

        try {

            String userHome = System.getProperty("user.home");

            File file = new File(userHome + "/Desktop/order\_summary.txt");

            file.getParentFile().mkdirs();

            try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {

                writer.write(customer.toString());

                writer.newLine();

                writer.write("----- Ordered Items -----");

                writer.newLine();

                for (OrderItem item : items) {

                    writer.write(item.toString());

                    writer.newLine();

                }

                writer.write("-------------------------");

                writer.newLine();

                writer.write(String.format("Total: $%.2f", getTotalPrice()));

                writer.newLine();

            }

            logger.info("Order saved to file: " + file.getAbsolutePath());

        } catch (IOException e) {

            logger.error("Failed to save order to file.", e);

        }

    }

}

**OrderItem.java**

package sales;

public class OrderItem {

    private Product product;

    private int quantity;

    public OrderItem(Product product, int quantity) {

        if (product == null) {

            throw new IllegalArgumentException("Product cannot be null.");

        }

        if (quantity <= 0) {

            throw new IllegalArgumentException("Quantity must be positive.");

        }

        this.product = product;

        this.quantity = quantity;

    }

    public Product getProduct() {

        return product;

    }

    public int getQuantity() {

        return quantity;

    }

    public double getSubtotal() {

        return product.getPrice() \* quantity;

    }

    @Override

    public String toString() {

        return String.format("%s x%d = $%.2f", product.getName(), quantity, getSubtotal());

    }

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

                             http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.sales</groupId>

    <artifactId>sales-system</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>Sales System</name>

    <description>Java GUI Sales Management System with Logging</description>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

        <maven.compiler.source>11</maven.compiler.source>

        <maven.compiler.target>11</maven.compiler.target>

        <log4j.version>2.22.1</log4j.version>

        <junit.version>4.13.2</junit.version>

    </properties>

    <dependencies>

        <!-- Log4j -->

        <dependency>

            <groupId>org.apache.logging.log4j</groupId>

            <artifactId>log4j-api</artifactId>

            <version>${log4j.version}</version>

        </dependency>

        <dependency>

            <groupId>org.apache.logging.log4j</groupId>

            <artifactId>log4j-core</artifactId>

            <version>${log4j.version}</version>

        </dependency>

        <!-- JUnit -->

        <dependency>

            <groupId>junit</groupId>

            <artifactId>junit</artifactId>

            <version>${junit.version}</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Compiler -->

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.11.0</version>

                <configuration>

                    <source>${maven.compiler.source}</source>

                    <target>${maven.compiler.target}</target>

                </configuration>

            </plugin>

            <!-- Exec Plugin -->

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>sales.SalesSystemGUI</mainClass>

                </configuration>

            </plugin>

            <!-- Assembly for jar-with-dependencies -->

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-assembly-plugin</artifactId>

                <version>3.6.0</version>

                <configuration>

                    <archive>

                        <manifest>

                            <mainClass>sales.SalesSystemGUI</mainClass>

                        </manifest>

                    </archive>

                    <descriptorRefs>

                        <descriptorRef>jar-with-dependencies</descriptorRef>

                    </descriptorRefs>

                </configuration>

                <executions>

                    <execution>

                        <id>make-assembly</id>

                        <phase>package</phase>

                        <goals>

                            <goal>single</goal>

                        </goals>

                    </execution>

                </executions>

            </plugin>

        </plugins>

    </build>

</project>

**Product.java**

package sales;

public class Product {

    private String name;

    private double price;

    private int stockQuantity;

    public Product(String name, double price, int stockQuantity) {

        if (!name.matches("[a-zA-Z0-9\\s]+")) {

            throw new IllegalArgumentException("Product name must contain only letters and numbers.");

        }

        if (price <= 0) {

            throw new IllegalArgumentException("Price must be positive.");

        }

        if (stockQuantity < 0) {

            throw new IllegalArgumentException("Stock cannot be negative.");

        }

        this.name = name;

        this.price = price;

        this.stockQuantity = stockQuantity;

    }

    public String getName() {

        return name;

    }

    public double getPrice() {

        return price;

    }

    public int getStockQuantity() {

        return stockQuantity;

    }

    public void reduceStock(int qty) {

        if (qty < 0 || qty > stockQuantity) {

            throw new IllegalArgumentException("Invalid quantity.");

        }

        stockQuantity -= qty;

    }

    public void increaseStock(int qty) {

        if (qty < 0) {

            throw new IllegalArgumentException("Cannot add negative stock.");

        }

        stockQuantity += qty;

    }

    @Override

    public String toString() {

        return name + " - $" + price;

    }

}

**ProductTest.java**

package sales;

import org.junit.Test;

import static org.junit.Assert.\*;

public class ProductTest {

    @Test

    public void testAllProductScenarios() {

        // 1. Test valid product creation

        Product laptop = new Product("Laptop", 999.99, 10);

        assertEquals("Laptop", laptop.getName());

        assertEquals(999.99, laptop.getPrice(), 0.001);

        assertEquals(10, laptop.getStockQuantity());

        // 2. Test invalid product name

        try {

            new Product("!@#$", 100, 5);

            fail("Should throw IllegalArgumentException for invalid name");

        } catch (IllegalArgumentException e) {

            // Expected

        }

        // 3. Test negative price

        try {

            new Product("Phone", -50, 5);

            fail("Should throw IllegalArgumentException for negative price");

        } catch (IllegalArgumentException e) {

            // Expected

        }

        // 4. Test negative stock quantity

        try {

            new Product("Phone", 100, -1);

            fail("Should throw IllegalArgumentException for negative stock");

        } catch (IllegalArgumentException e) {

            // Expected

        }

        // 5. Test reduceStock beyond available

        Product tablet = new Product("Tablet", 200, 3);

        try {

            tablet.reduceStock(5);

            fail("Should throw IllegalArgumentException when reducing beyond stock");

        } catch (IllegalArgumentException e) {

            // Expected

        }

        // 6. Test increaseStock

        Product camera = new Product("Camera", 300, 2);

        camera.increaseStock(5);

        assertEquals(7, camera.getStockQuantity());

    }

}

**SalesSystemGUI.java**

package sales;

import java.awt.BorderLayout;

import java.awt.FlowLayout;

import java.util.List;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

public class SalesSystemGUI extends JFrame {

    private static final Logger logger = LogManager.getLogger(SalesSystemGUI.class);

    private List<Product> products;

    private Customer customer;

    private Order order;

    private JComboBox<Product> productCombo;

    private JTextField qtyField;

    private JTextArea orderArea;

    private JLabel totalLabel;

    public SalesSystemGUI() {

        logger.info("Application started.");

        setTitle("Sales System");

        setSize(550, 450);

        setLocationRelativeTo(null);

        setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        initProducts();

        initComponents();

        askCustomerInfo();

    }

    private void initProducts() {

        products = List.of(

            new Product("Mouse", 10.0, 50),

            new Product("Keyboard", 30.0, 30),

            new Product("Laptop", 900.0, 10)

        );

    }

    private void initComponents() {

        setLayout(new BorderLayout());

        // Top panel

        JPanel inputPanel = new JPanel(new FlowLayout());

        productCombo = new JComboBox<>(products.toArray(new Product[0]));

        qtyField = new JTextField(5);

        JButton addButton = new JButton("Add");

        inputPanel.add(new JLabel("Product:"));

        inputPanel.add(productCombo);

        inputPanel.add(new JLabel("Quantity:"));

        inputPanel.add(qtyField);

        inputPanel.add(addButton);

        add(inputPanel, BorderLayout.NORTH);

        // Center panel

        orderArea = new JTextArea();

        orderArea.setEditable(false);

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

        // Bottom panel

        JPanel bottomPanel = new JPanel(new BorderLayout());

        totalLabel = new JLabel("Total: $0.00");

        JButton saveButton = new JButton("Place Order");

        bottomPanel.add(totalLabel, BorderLayout.WEST);

        bottomPanel.add(saveButton, BorderLayout.EAST);

        add(bottomPanel, BorderLayout.SOUTH);

        // Add actions

        addButton.addActionListener(e -> addProductToOrder());

        saveButton.addActionListener(e -> saveOrder());

    }

    private void askCustomerInfo() {

        try {

            String name = JOptionPane.showInputDialog(this, "Enter your name:");

            String email = JOptionPane.showInputDialog(this, "Enter your email:");

            String phone = JOptionPane.showInputDialog(this, "Enter your phone number:");

            customer = new Customer(name, email, phone);

            order = new Order(customer);

            logger.info("Customer registered: " + name);

        } catch (IllegalArgumentException ex) {

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

            askCustomerInfo(); // try again

        }

    }

    private void addProductToOrder() {

        try {

            Product selectedProduct = (Product) productCombo.getSelectedItem();

            int qty = Integer.parseInt(qtyField.getText());

            boolean success = order.addItem(selectedProduct, qty);

            if (success) {

                updateOrderDisplay();

                qtyField.setText("");

            } else {

                JOptionPane.showMessageDialog(this, "Not enough stock.", "Stock Error", JOptionPane.WARNING\_MESSAGE);

            }

        } catch (NumberFormatException ex) {

            JOptionPane.showMessageDialog(this, "Invalid quantity.", "Input Error", JOptionPane.ERROR\_MESSAGE);

        }

    }

    private void updateOrderDisplay() {

        orderArea.setText("");

        for (OrderItem item : order.getItems()) {

            orderArea.append(item.toString() + "\n");

        }

        totalLabel.setText(String.format("Total: $%.2f", order.getTotalPrice()));

    }

    private void saveOrder() {

        if (order.isEmpty()) {

            JOptionPane.showMessageDialog(this, "No items in the order.", "Warning", JOptionPane.WARNING\_MESSAGE);

            return;

        }

        order.saveToFile();

        JOptionPane.showMessageDialog(this, "Order saved to Desktop!", "Success", JOptionPane.INFORMATION\_MESSAGE);

        logger.info("Order placed by: " + customer.getName());

        dispose();

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new SalesSystemGUI().setVisible(true));

    }

}

















