HNDIT4232 - Enterprise Architecture

(Java Thread/JDBC/XML/Servlet)

Sri Lanka Institute of Advanced Technological Education

Department of Information Technology

GAM/IT/2022/F/0046 – P.I.Kaushani

Java Thread

Practical 01

```
public class SimpleThread extends Thread{
   @Override
   public void run(){
   System.out.println(Thread.currentThread().getId()+"is executing the thread.");
   }
}


public class MultiThreadApp {
public static void main(String[] args) {
       SimpleThread thread1=new SimpleThread();
       SimpleThread thread2=new SimpleThread();


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

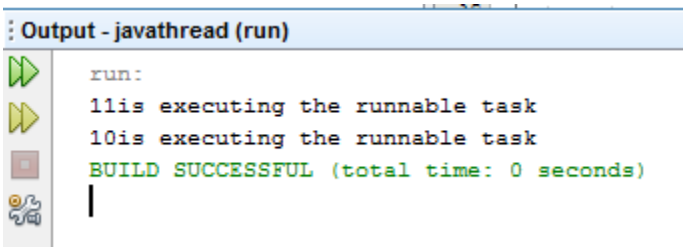
Output

```
10is executing the thread.
11is executing the thread.
BUILD SUCCESSFUL (total time: 1 second)
```

Practical 02

```
public class RunnableTask implements Runnable {
 @Override
 public void run(){
 System.out.println(Thread.currentThread().getId()+"is executing the runnable task");
 }
 }

public static void main(String[] args)throws InterruptedException {
        RunnableTask task1=new RunnableTask();
        RunnableTask task2=new RunnableTask();

         Thread thread1=new Thread(task1);
         Thread thread2=new Thread(task2);

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

output



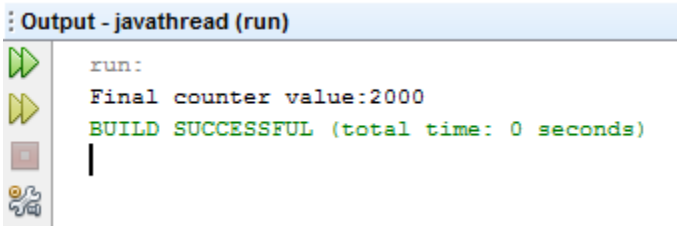
Practical 03

```
public class Counter {
    private int count = 0;
    public synchronized void increment (){
    count++;
    }
    public int getCount(){
    return count;
    }
    }

public class SynchronizedExample extends Thread{
     private Counter counter;

    public SynchronizedExample(Counter counter){
    this.counter = counter;

    }
    public void run(){
    for (int i = 0;i<1000;i++){
    counter.increment();
    }
    }
    }
```

```
Counter counter =new Counter();
```

```
        Thread thread1 = new SynchronizedExample(counter);
        Thread thread2 = new SynchronizedExample(counter);

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

        thread1.join();
        thread2.join();

        System.out.println("Final counter value:" + counter.getCount());
    }

}
```

Output



Practical 04

```
public class Task implements Runnable{
   private int taskId;

   public Task (int taskId){
   this.taskId=taskId;
   }
   @Override
   public void run(){
   System.out.println("Task" +taskId+"is being processed by"+Thread.currentThread().getName());
   }
  }

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ThreadPoolExample {
 public static void main(String[] args) {
      ExecutorService executorService = Executors.newFixedThreadPool(3);

      for(int i= 1;i<=5;i++){
      executorService.submit(new Task(i));
      }
      executorService.shutdown();
       }
  }
```

Output

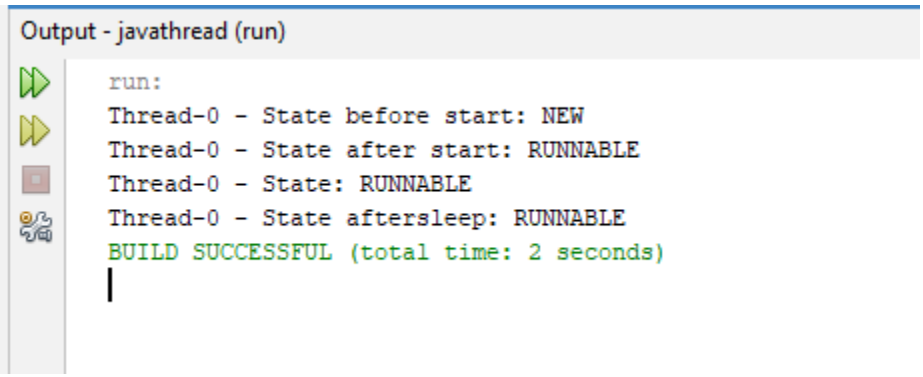
```
Output - ThreadPoolExample (run)
run:
Task2is being processed bypool-1-thread-2
Task1is being processed bypool-1-thread-1
Task3is being processed bypool-1-thread-3
Task4is being processed bypool-1-thread-1
Task5is being processed bypool-1-thread-3
BUILD SUCCESSFUL (total time: 1 second)
```

Practical 05

```
public class ThreadLifecycleExample extends Thread {
    @Override
    public void run() {
System.out.println(Thread.currentThread().getName() + " - State:
+Thread.currentThread().getState());
try {
Thread.sleep(2000); // Simulate waiting state
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println(Thread.currentThread().getName() + " - State aftersleep: " +
Thread.currentThread().getState());
}
}
```

```
public class Javathread {
public static void main(String[] args) {
ThreadLifecycleExample thread = new ThreadLifecycleExample();
System.out.println(thread.getName() + " - State before start: " +thread.getState());
thread.start(); // Start the thread
System.out.println(thread.getName() + " - State after start: " +thread.getState());
}}
```

Output:



JDBC

Main.java

```
package jdbcexamplea;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class main {
public static void main(String[] args) {
// Add employees
employeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
employeeDAO.addEmployee("Bob Marley", "Manager", 80000);

// Update employee

employeeDAO.updateEmployee( 1,"John Doe", "Senior Software Engineer", 90000);
// Get all employees
List<Employee> employees = employeeDAO.getAllEmployees();
employees.forEach(System.out::println);
// Delete employee
employeeDAO.deleteEmployee(2);
}
}
```

employeeDAO.java

```
package jdbcexamplea;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class employeeDAO {
   // Create an employee
public static void addEmployee(String name, String position, double salary) {
String sql = "INSERT INTO employees (name, position, salary) VALUES(?, ?, ?)";
try (Connection conn = databaseconnec.getConnection();
PreparedStatement stmt = conn.prepareStatement(sql)) {
stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);
int rowsAffected = stmt.executeUpdate();
System.out.println("Employee added successfully. Rows affected: "+ rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
public static List <Employee> getAllEmployees() {
List<Employee> employees = new ArrayList<>();
String sql =("SELECT * FROM employees") ;
try (Connection conn = databaseconnec.getConnection();
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(sql)) {
while (rs.next()) {
```

```
Employee employee = new Employee(
rs.getInt("id"),
rs.getString("name"),
rs.getString("position"),
rs.getDouble("salary")
);
employees.add(employee);
}
} catch (SQLException e) {
e.printStackTrace();
}
return employees;
}
// Update an employee's information
public  static void updateEmployee(int id, String name, String position, double salary) {
   String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?"; // Corrected SQL
   try (Connection conn = databaseconnec.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
       stmt.setString(1, name);
       stmt.setString(2, position);
       stmt.setDouble(3, salary);
       stmt.setInt(4, id);
       int rowsAffected = stmt.executeUpdate();
       System.out.println("Employee updated successfully. Rows affected: " + rowsAffected);
   } catch (SQLException e) {
       e.printStackTrace();
   }
}
// Delete an employee
```

```
public static void deleteEmployee(int id) {
String sql = "DELETE FROM employees WHERE id = ?";
try (Connection conn = databaseconnec.getConnection();
PreparedStatement stmt = conn.prepareStatement(sql)) {
stmt.setInt(1, id);
int rowsAffected = stmt.executeUpdate();
System.out.println("Employee deleted successfully. Rows affected: " + rowsAffected);
} catch (SQLException e) {
e.printStackTrace();
}
}
}
```

Databaseconnec.java

```
package jdbcexamplea;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class databaseconnec {
private static final String URL ="jdbc:mysql://localhost:3306/employee_db"; // Database URL
private static final String USER = "root"; // Your MySQL username
private static final String PASSWORD = ""; // Your MySQL password
public static Connection getConnection() throws SQLException {
try {
// Load the JDBC driver
```

```
Class.forName("com.mysql.cj.jdbc.Driver");
// Return the database connection
return DriverManager.getConnection(URL, USER, PASSWORD);
} catch (ClassNotFoundException | SQLException e) {
System.out.println("Connection failed:" + e.getMessage());
throw new SQLException("Failed to establish connection");
}
}
}
```

Employee.java

```
package jdbcexamplea;

public class Employee {
private int id;
private String name;
private String position;
private double salary;
public Employee(int id, String name, String position, double salary) {
this.id = id;
this.name = name;
this.position = position;
this.salary = salary;
}
// Getters and setters
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }

public String getPosition() { return position; }

public void setPosition(String position) { this.position = position; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

@Override

public String toString() {

   return "Employee{id=" + id + ", name='" + name + "', position='" + position + "', salary=" + salary + '}';

}

}
```

Output

				id	name	position	salary
☐	Edit	Copy	Delete	1	John Doe	Senior Software Engineer	90000.00
☐	Edit	Copy	Delete	3	Steve Brown	Team Lead	85000.00
☐	Edit	Copy	Delete	4	Alice Cooper	Developer	70000.00
☐	Edit	Copy	Delete	5	Bob Marley	Manager	80000.00
☐	Edit	Copy	Delete	6	Alice Cooper	Developer	70000.00
☐	Edit	Copy	Delete	7	Bob Marley	Manager	80000.00

XML

books.xml

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

<library>
<book>
<title>The Great Gatsby</title>
<author>F. Scott Fitzgerald</author>
<year>1925</year>
<genre>Fiction</genre>
</book>
<book>
<title>To Kill a Mockingbird</title>
<author>Harper Lee</author>
<year>1960</year>
<genre>Fiction</genre>
</book>
<book>
<title>1984</title>
<author>George Orwell</author>
<year>1949</year>
<genre>Dystopian</genre>
</book>
</library>
```

XmlParser.java

```
package xml;
import org.w3c.dom.*;
import javax.xml.parsers.*;
public class XmlParser {
    public static void main(String[] args) {
try {
// Create a new DocumentBuilderFactory and DocumentBuilder
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();

// Parse the XML file
Document document = builder.parse("D:\\java exercises\\XML\\src\\xml\\books.xml");

// Normalize the document
document.getDocumentElement().normalize();

// Get the root element (library)
 NodeList nodeList = document.getElementsByTagName("book");

 // Loop through each book in the XML document
for (int i = 0; i < nodeList.getLength(); i++) {
Node node = nodeList.item(i);

if (node.getNodeType() == Node.ELEMENT_NODE) {
Element element = (Element) node;
```

```
// Get and print the details of each book
String title = element.getElementsByTagName("title").item(0).getTextContent();
String author = element.getElementsByTagName("author").item(0).getTextContent();
String year = element.getElementsByTagName("year").item(0).getTextContent();
String genre = element.getElementsByTagName("genre").item(0).getTextContent();

System.out.println("Title: " + title);
System.out.println("Author: " + author);
System.out.println("Year: " + year);
System.out.println("Genre: " + genre);
System.out.println("------------");
}
}

} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Output:

Output - XML (run)

```
run:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Year: 1925
Genre: Fiction
-----------
Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Genre: Fiction
-----------
Title: 1984
Author: George Orwell
Year: 1949
Genre: Dystopian
-----------
BUILD SUCCESSFUL (total time: 0 seconds)
```

Stock Management

Database Setup (MySQL example):

```
CREATE DATABASE stock_management;
USE stock_management;
CREATE TABLE stock (
id INT AUTO_INCREMENT PRIMARY KEY,
product_name VARCHAR(255),
quantity INT
);
```

HTML Form (stockForm.html):

```
<!DOCTYPE html>
<html>
<head><title>Stock Management</title></head>
<body>
<h2>Manage Stock</h2>
<form action="stockAction" method="POST">
Product Name: <input type="text" name="product_name" required><br>
Quantity: <input type="number" name="quantity" required><br>
<input type="submit" name="action" value="Add Product">
<input type="submit" name="action" value="Update Product">
<input type="submit" name="action" value="Delete Product">
</form>
</body>
</html>
```

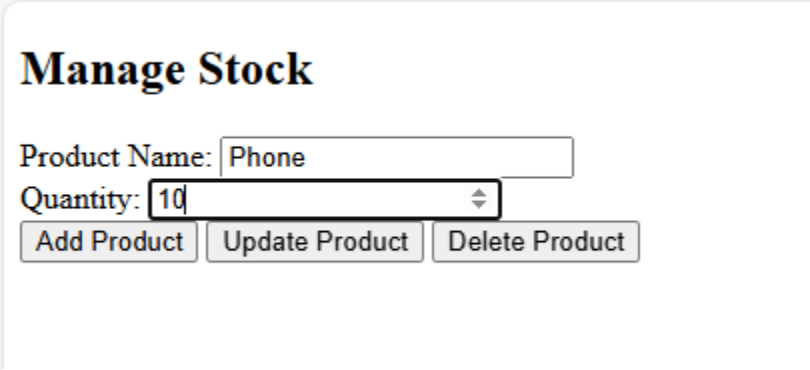
Servlet Code (StockManagementServlet.java):

```
package com.example;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/stockAction")
public class StockManagementServlet extends HttpServlet {
private Connection getConnection() throws SQLException {
String url = "jdbc:mysql://localhost:3306/stock_management";
String username = "root";
String password = "root"; // replace with your database password
return DriverManager.getConnection(url, username, password);
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
String action = request.getParameter("action");
String productName = request.getParameter("product_name");
int quantity = Integer.parseInt(request.getParameter("quantity"));
try (Connection conn = getConnection()) {
```

```
switch(action) {
case "Add Product":
try (PreparedStatement stmt = conn.prepareStatement(
"INSERT INTO stock (product_name, quantity)
VALUES (?, ?)")) {
stmt.setString(1, productName);
stmt.setInt(2, quantity);
stmt.executeUpdate();

response.getWriter().write("<h1>Product Added

Successfully</h1>");
}
break;

case "Update Product":
try (PreparedStatement stmt = conn.prepareStatement(
"UPDATE stock SET quantity = ? WHERE product_name
= ?")) {
stmt.setInt(1, quantity);
stmt.setString(2, productName);

stmt.executeUpdate();

response.getWriter().write("<h1>Product Updated

Successfully</h1>");
```

```
}
break;

case "Delete Product":
try (PreparedStatement stmt = conn.prepareStatement(
"DELETE FROM stock WHERE product_name = ?")) {
stmt.setString(1, productName);

stmt.executeUpdate();

response.getWriter().write("<h1>Product Deleted
Successfully</h1>");
}
break;
default:
response.getWriter().write("<h1>Invalid Action</h1>");
}
} catch (SQLException e) {

e.printStackTrace();
response.getWriter().write("<h1>Database Error: " +
e.getMessage() + "</h1>");
}
}
}
```

Output


Manage Stock
Product Name: Phone
Quantity: 10
Add Product
Update Product
Delete Product


Product Added Successfully
Back to Form
id
product_name
quantity
Edit
Copy
Delete
1
Phone
10

Servlet Code (DisplayProductsServlet.java):

```
package com.example;


import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
@WebServlet("/displayProducts")
public class DisplayProductsServlet extends HttpServlet {
   // Reuse your existing connection method
   private Connection getConnection() throws SQLException {
      String url =
"jdbc:mysql://localhost:3306/stock_management?useSSL=false&serverTimezone=UTC";
      String username = "root";
      String password = "316830059";
      return DriverManager.getConnection(url, username, password);
   }
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Stock List</title>");
out.println("<style>");
out.println("table { border-collapse: collapse; width: 50%; margin: 20px auto; }");
out.println("th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }");
out.println("th { background-color: #f2f2f2; }");
out.println("</style>");
out.println("</head>");
out.println("<body>");
out.println("<h1 style='text-align: center;'>Current Stock List</h1>");

try (Connection conn = getConnection();
          Statement stmt = conn.createStatement();
ResultSetrs = stmt.executeQuery("SELECT * FROM stock")) {
out.println("<table>");
out.println("<tr><th>ID</th><th>Product Name</th><th>Quantity</th></tr>");

         while (rs.next()) {
out.println("<tr>");
out.println("<td>" + rs.getInt("id") + "</td>");
out.println("<td>" + rs.getString("product_name") + "</td>");
out.println("<td>" + rs.getInt("quantity") + "</td>");
out.println("</tr>");
         }
```

```
out.println("</table>");
        } catch (SQLException e) {
out.println("<h2 style='color: red; text-align: center;'>Error retrieving stock: "
                    + e.getMessage() + "</h2>");
e.printStackTrace();      }

out.println("<div style='text-align: center; margin-top: 20px;'>");
out.println("<a href='stockForm.html'>Back to Stock Management</a>");
out.println("</div>");
out.println("</body>");
out.println("</html>");
    }
}
```

Updated stockForm.html

```
<!DOCTYPE html>
<html>
<head><title>Stock Management</title>
<style>    body { font-family: Arial, sans-serif; margin: 20px; }
               form { max-width: 500px; margin: 0 auto; padding: 20px; border: 1px solid #ddd; border-
radius: 5px; }
               input[type="text"], input[type="number"] { width: 100%; padding: 8px; margin: 5px 0 15px; }
                 input[type="submit"] { padding: 8px 15px; margin-right: 10px; }
.view-link { display: block; text-align: center; margin-top: 20px; }
</style>
</head>
<body><h2 style="text-align: center;">   Manage Stock   </h2>
<form action="stockAction" method="POST">
              Product Name: <input type="text" name="product_name" required><br>
               Quantity: <input type="number" name="quantity" required><br>
<input type="submit" name="action" value="Add Product">
<input type="submit" name="action" value="Update Product">
<input type="submit" name="action" value="Delete Product">
</form>
<div class="view-link">
<a href="displayProducts"> View All Products   </a>
</div>
</body>
</html>
```

Output

Database

Current Stock List

ID	Product Name	Quantity
1	phone	10
2	iphone	20

Back to Stock Management

			id	product_name	quantity
Edit	Copy	Delete	1	phone	10
Edit	Copy	Delete	2	iphone	20

Check all *With selected:* Edit Copy Delete Export