**Enterprise Architecture Assignment**

**A yellow and blue sign with a diamond shaped design

AI-generated content may be incorrect.**

**Higher National Diploma in Information Technology**

**Sri Lanka Institute of Advanced Technological Education -Gampaha**

**Submitted By : S.A.N.S.Subasingha - GAM/IT/2022/F/0066**

**Date of submission : 19/05/2025**

Table of Contents

[1. Introduction 3](#_Toc198557484)

[2. Java Thread 3](#_Toc198557485)

[2.1 What is a Thread? 3](#_Toc198557486)

[2.2 Thread Lifecycle 3](#_Toc198557487)

[2.3 Creating Threads 4](#_Toc198557488)

[2.4 Thread Methods 6](#_Toc198557489)

[2.5 Synchronization 6](#_Toc198557490)

[2.6 Using ExecutorService for Thread Pooling 8](#_Toc198557491)

[2.7 Thread Lifecycle Example 9](#_Toc198557491)

[2.8 Real-world Use 10](#_Toc198557491)

[3. JDBC (Java Database Connectivity) 10](#_Toc198557492)

[3.1 What is JDBC? 10](#_Toc198557493)

[3.2 Architecture 10](#_Toc198557494)

[3.3 Steps to Connect Database 11](#_Toc198557495)

[3.4 PreparedStatement Example 12](#_Toc198557496)

[3.5 Test the Application 12](#_Toc198557496)

[3.6 Best Practices 13](#_Toc198557497)

[4. XML (eXtensible Markup Language) 13](#_Toc198557498)

[4.1 What is XML? 13](#_Toc198557499)

[4.2 XML Structure 13](#_Toc198557500)

[4.3 Parsing XML in Java 14](#_Toc198557501)

[4.4 Reading Example with DOM 14](#_Toc198557502)

[4.5 XML in Java Applications 14](#_Toc198557503)

[5. Servlet 15](#_Toc198557504)

[5.1 What is a Servlet? 15](#_Toc198557505)

[5.2 Servlet Lifecycle 15](#_Toc198557506)

[5.3 Creating a Servlet 15](#_Toc198557507)

[5.4 Servlet Configuration 15](#_Toc198557508)

[5.5 Java Servlet with Database CRUD Operations 16](#_Toc198557508)

[5.6 Session Management 17](#_Toc198557509)

[5.7 Real-world Use 17](#_Toc198557510)

[6. Conclusion 18](#_Toc198557511)

[7. References 18](#_Toc198557512)

# Introduction Java is a strong and flexible programming language used to create websites, desktop programs, and large business systems. This report explains four main Java technologies:

 **Java Threads** – help run many tasks at the same time

 **JDBC (Java Database Connectivity)** – lets Java connect to and use data from databases

 **XML (eXtensible Markup Language)** – used to store and share data in an organized way

 **Servlets** – allow Java programs to work with web servers and respond to users' requests

# Java Thread

## 2.1 What is a Thread?

## A thread in Java is a small part of a program that can run on its own. Java allows many threads to run at the same time, which is called multithreading. This helps the computer use its power better by doing many tasks at once.

## 2.2 Thread Lifecycle

In Java, a thread goes through different steps, called its lifecycle. These steps are:

* **New** – The thread is created but not started yet.
* **Runnable** – The thread is ready to run and waiting for the CPU.
* **Running** – The thread is currently being executed by the CPU.
* **Blocked/Waiting** – The thread is paused and waiting for something (like a resource or another thread).
* **Terminated** – The thread has finished its work and stopped running.

## 2.3 Creating Threads

**1. Extending Thread class**

public class SimpleThread extends Thread {

@Override

public void run() {

System.out.println(Thread.currentThread().getId() + " is executing the thread.");}

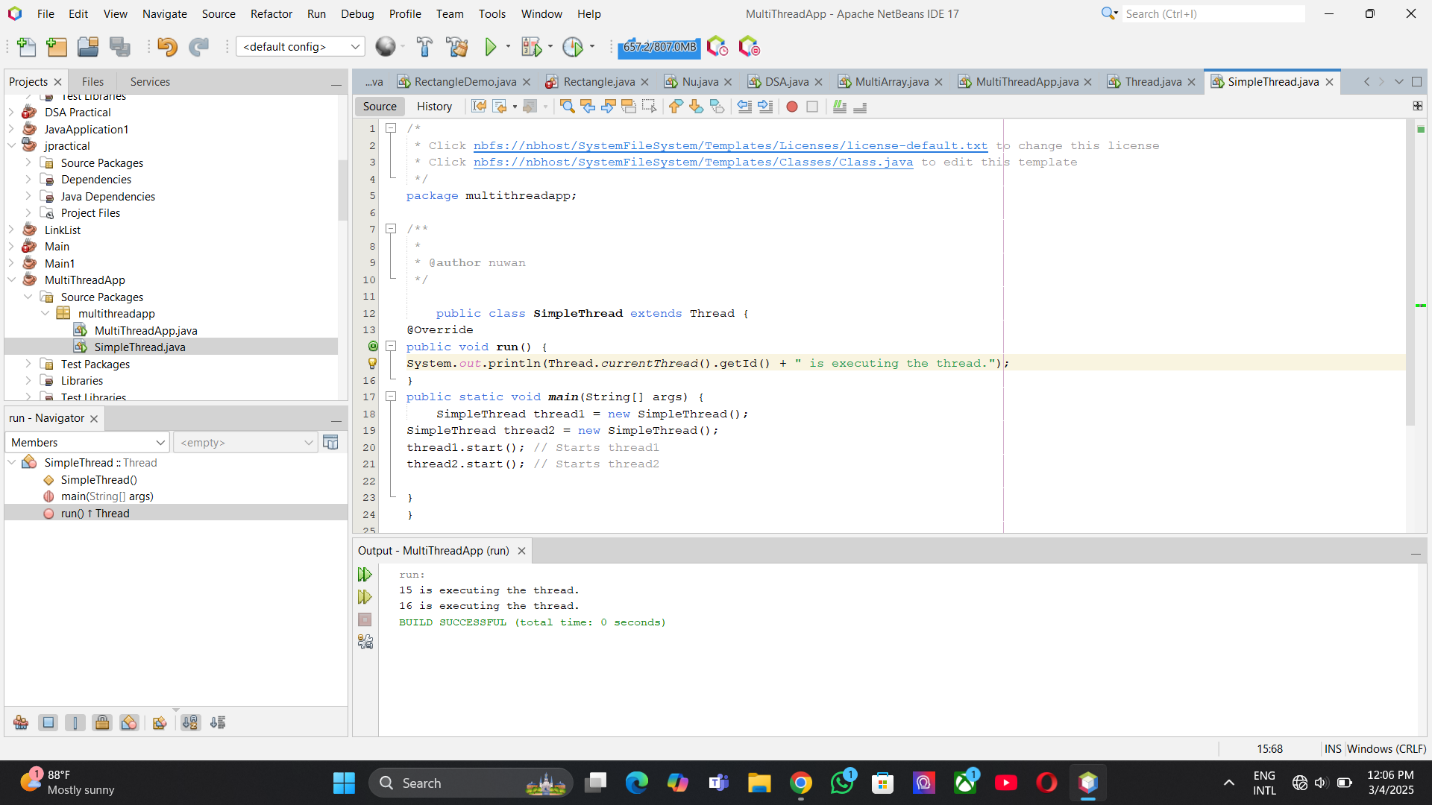
public static void main(String[] args) {

SimpleThread thread1 = new SimpleThread();

SimpleThread thread2 = new SimpleThread();

thread1.start(); // Starts thread1

thread2.start(); // Starts thread2 }}



**2. Implementing Runnable interface**

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) {

RunnableTask task1 = new RunnableTask();

RunnableTask task2 = new RunnableTask();

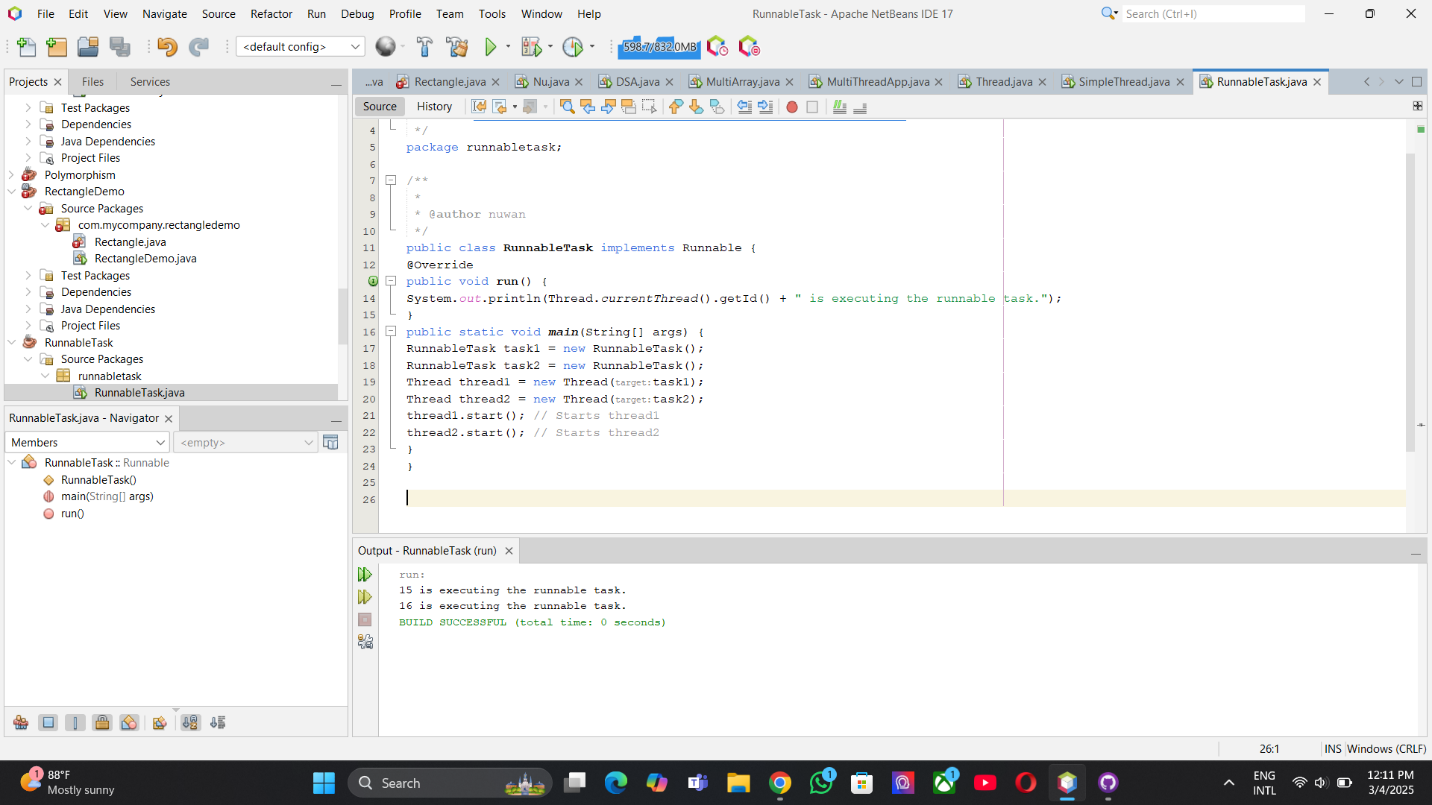
Thread thread1 = new Thread(task1);

Thread thread2 = new Thread(task2);

thread1.start(); // Starts thread1

thread2.start(); // Starts thread2

}}



**3.Using ExecutorService (preferred for managing thread pools)**

## 2.4 Thread Methods

Common methods include: start(), run(), sleep(ms), join(), yield(), wait(), notify().

## 2.5 Synchronization

class Counter {

private int count = 0;

// Synchronized method to ensure thread-safe access to the counter

public synchronized void increment() {

count++;

}

public int getCount() {

return count;

}

} }

**Used to prevent race conditions in multithreading using synchronized blocks or methods**

public class SynchronizedExample extends Thread {

private Counter counter;

public SynchronizedExample(Counter counter) {

this.counter = counter;

}

@Override

public void run() {

for (int i = 0; i < 1000; i++) {

counter.increment();

}

}

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

Counter counter = new Counter();

// Create and start multiple threads

Thread thread1 = new SynchronizedExample(counter);

Thread thread2 = new SynchronizedExample(counter);

thread1.start();

thread2.start();

// Wait for threads to finish

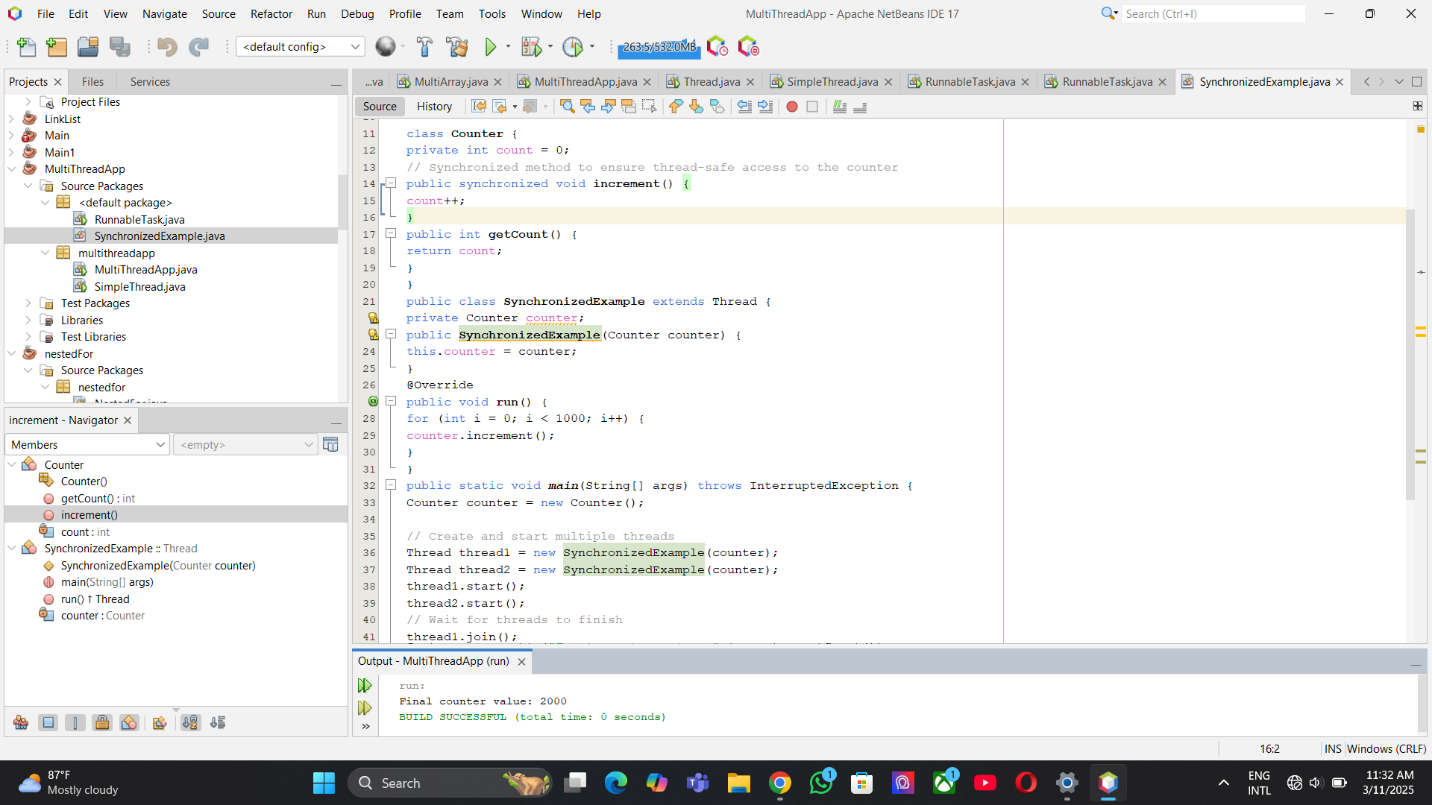
thread1.join();

thread2.join();

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

}

}



## 2.6 Using ExecutorService for Thread Pooling

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

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());

}

}

public class ThreadPoolExample {

public static void main(String[] args) {

// Create a thread pool with 3 threads

ExecutorService executorService = Executors.newFixedThreadPool(3);

// Submit tasks to the pool

for (int i = 1; i <= 5; i++) {

executorService.submit(new Task(i));

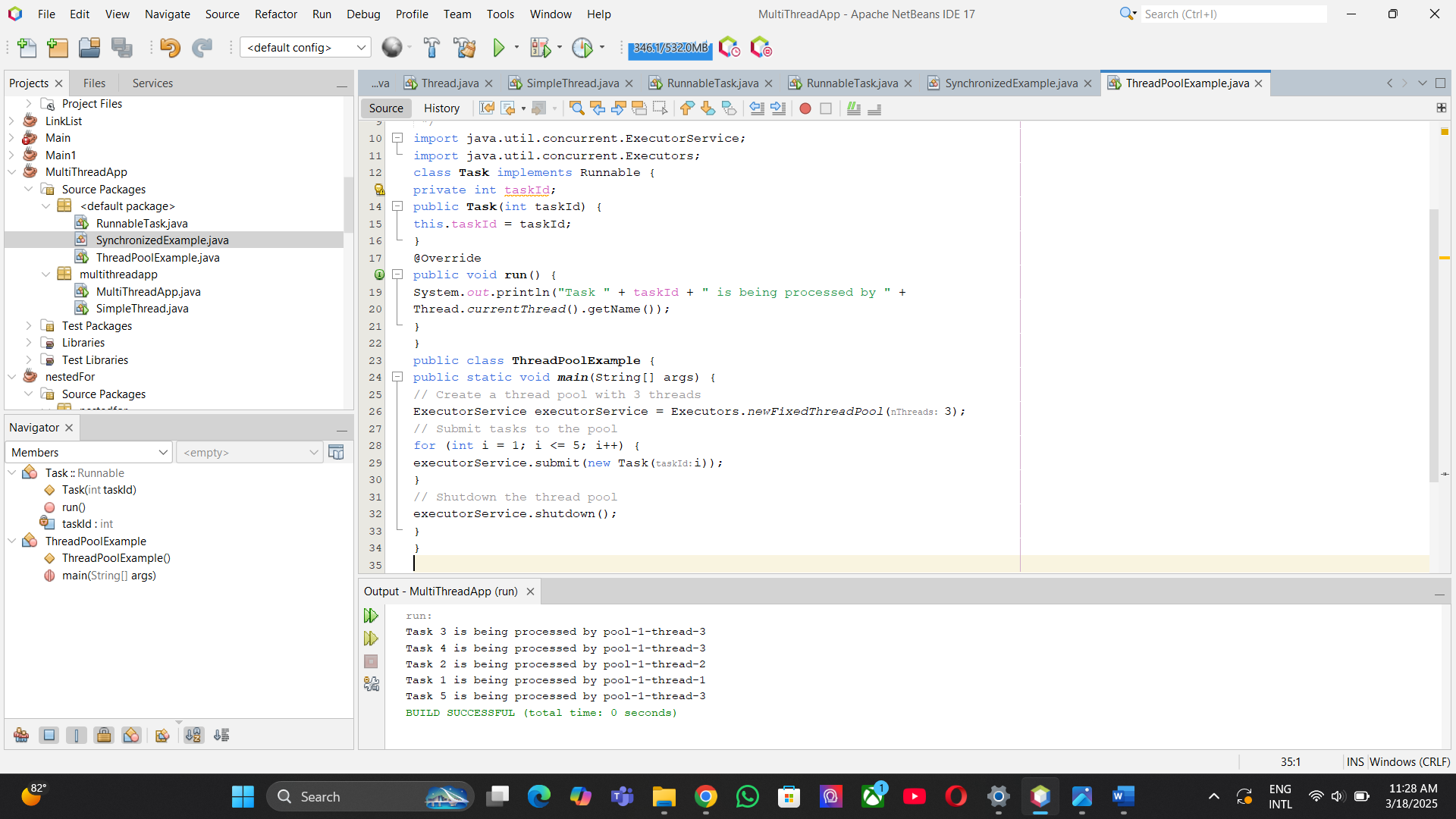
}

// Shutdown the thread pool

executorService.shutdown();

}

}



## 2.7 Thread Lifecycle Example

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 static void main(String[] args) {

ThreadLifecycleExample thread = new ThreadLifecycleExample();

System.out.println(thread.getName() + " - State before start: " +

thread.getState());

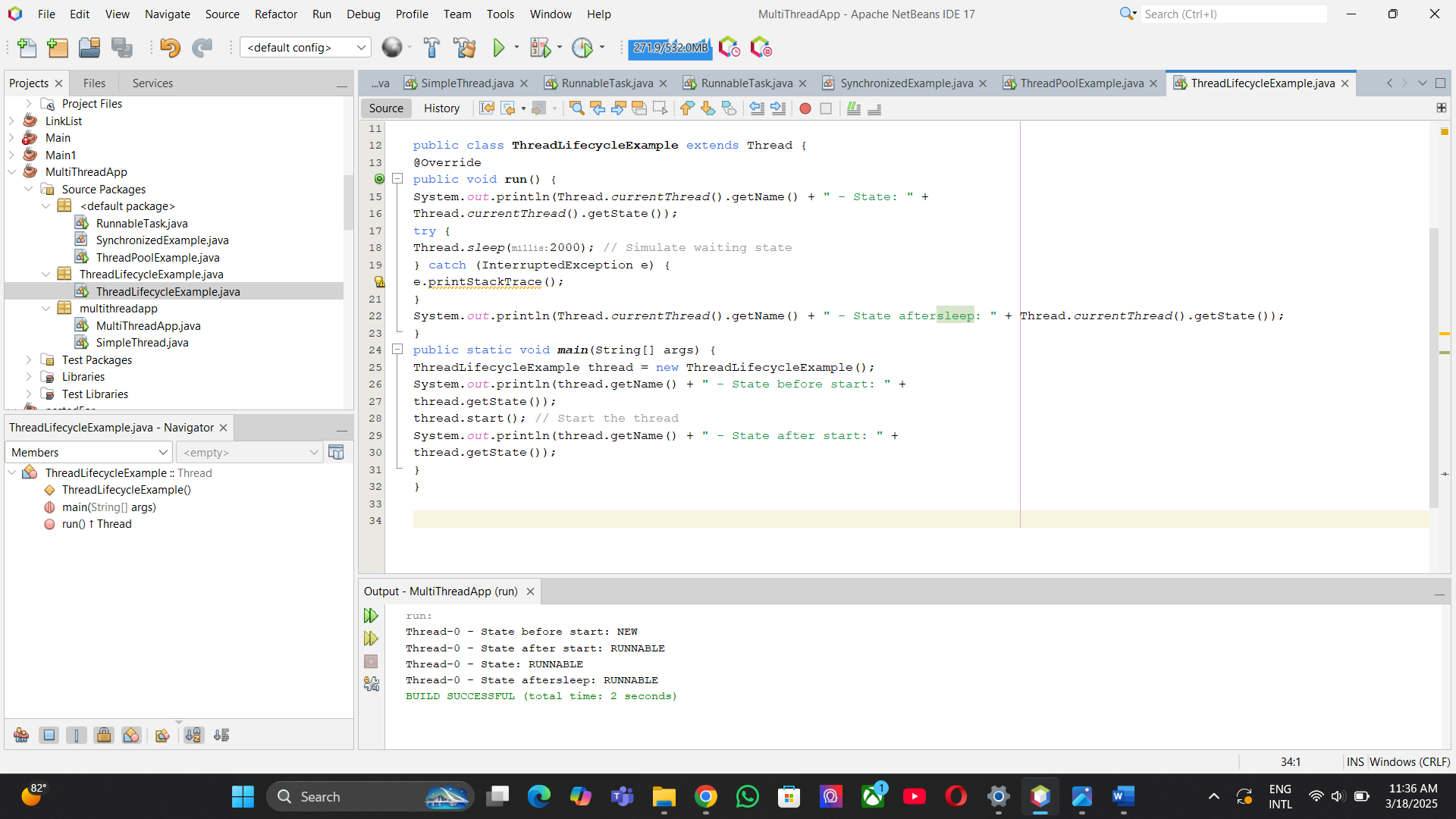
thread.start(); // Start the thread

System.out.println(thread.getName() + " - State after start: " +

thread.getState());

}

}



## 2.8 Real-world Use

Threads are used in real life for things like games, animations, downloading multiple files at once, and handling many users on servers.

# 3. JDBC (Java Database Connectivity)

## 3.1 What is JDBC?

## JDBC is a tool in Java that helps programs connect to databases like MySQL, Oracle, and PostgreSQL to get or save data.

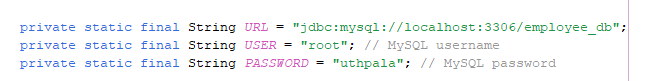
## 3.2 Architecture

Components: DriverManager, Connection, Statement/PreparedStatement, ResultSet.

## 3.3 Steps to Connect Database

1. Load driver

  
2. Create connection





3. Create statement





4. Execute query



A black and white text

AI-generated content may be incorrect.

5. Process results

A computer code with text

AI-generated content may be incorrect.

6. Close connection



## 3.4 PreparedStatement Example

PreparedStatement ps = con.prepareStatement("INSERT INTO user VALUES(?, ?)");

A computer screen shot of a code

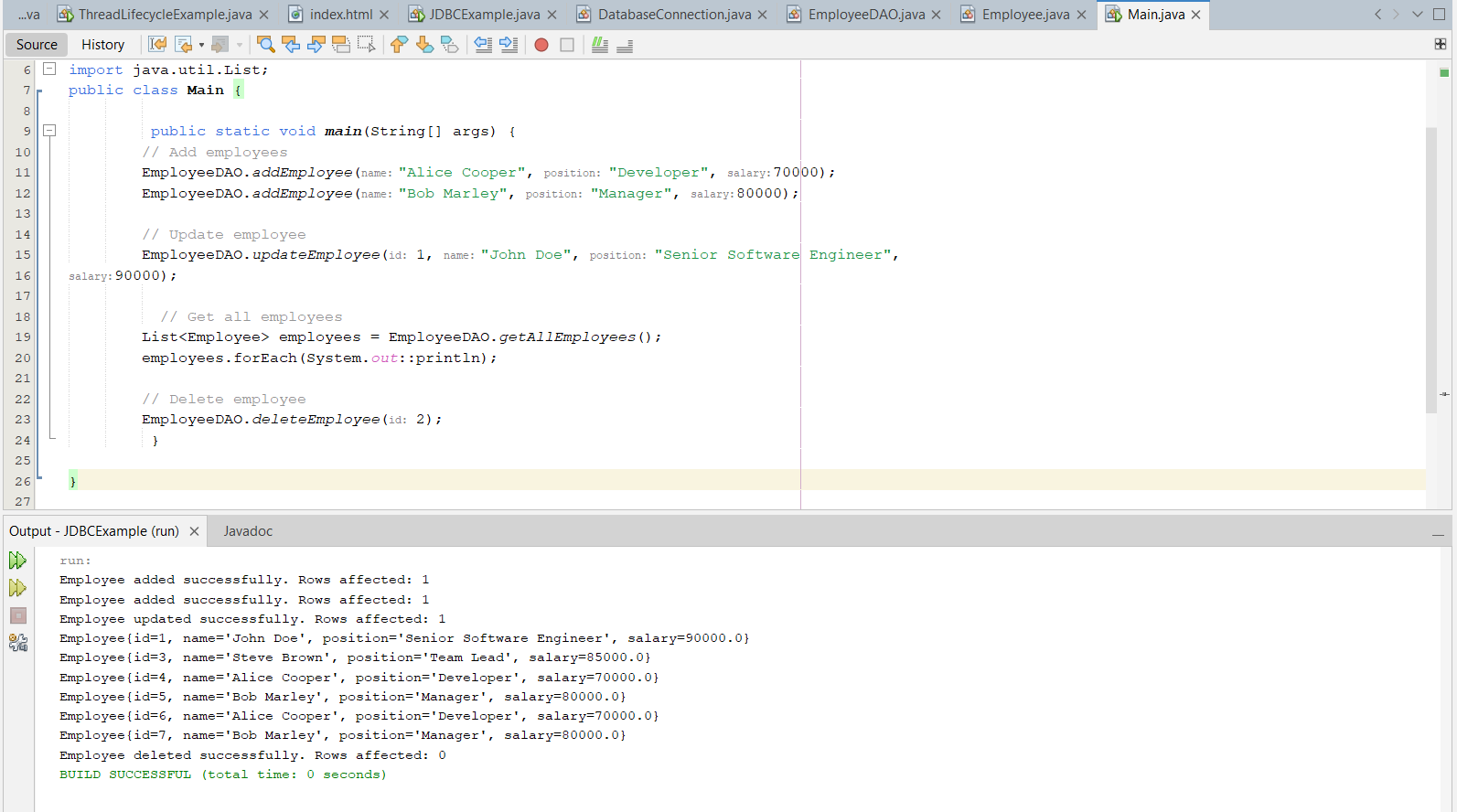
AI-generated content may be incorrect.

**3.5** **Test the Application**

**Create Employee.java Class**

## 

**Create Main.java Class**



## 3.6 Best Practices

Use try-with-resources, always close resources, use connection pooling.

# 4. XML (eXtensible Markup Language)

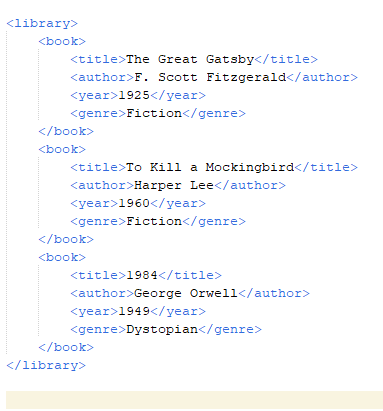
## 4.1 What is XML?

## XML is a language used to store and share data in a way that people can read and that works on any computer system

## 4.2 XML Structure

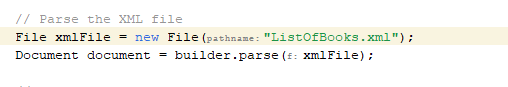
XML stores data using opening and closing tags, and in this example, it shows an employee with an ID and a name inside the <employee> tag.

<employee>  
 <id>101</id>  
 <name>John Doe</name>  
</employee>



## 4.3 Parsing XML in Java

DOM, SAX, and StAX parsers are used to read XML in Java.



## 4.4 Reading Example with DOM

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

A black text on a white background

AI-generated content may be incorrect.

## 4.5 XML in Java Applications

Used in configuration files, data exchange formats, and Android layouts.

# 5. Servlet

## 5.1 What is a Servlet?

## A Servlet is a Java program that runs on a web server to handle web requests and send back responses to users.

## 5.2 Servlet Lifecycle

The Servlet lifecycle has three main steps: init() to start the servlet, service() to handle requests, and destroy() to clean up when it's no longer needed.

## 5.3 Creating a Servlet

@WebServlet("/hello")  
public class HelloServlet extends HttpServlet { ... }

A computer screen shot of a program code

AI-generated content may be incorrect.

## 5.4 Servlet Configuration

Defined in web.xml or using annotations like @WebServlet.





## 5.5 Java Servlet with Database CRUD Operations

## 



## 5.6 Session Management

Session management in Java is done using **Cookies**, **HttpSession**, and **URL Rewriting** to remember user information during a visit to a website.

## 5.7 Real-world Use

Servlets are used in real life for things like login systems, contact forms, and online shopping websites.

# 6. Conclusion

# The Java ecosystem is large, and learning its main parts is important for software developers. Threads help run many tasks at the same time, JDBC lets programs safely connect to databases, XML is used to store and share data in an organized way, and Servlets are used to create web applications that can respond to users. These technologies work together to build powerful applications used in many industries.

# 7. References

* Java Threads – <https://docs.oracle.com/javase/tutorial/>
* JDBC (Java Database Connectivity) – https://www.geeksforgeeks.org/introduction-to-jdbc/
* A Guide to XML in Java – <https://www.baeldung.com/java-xml>
* Servlets Tutorial – <https://www.tutorialspoint.com/servlets/index.htm>
* Portfolio – https://github.com/NuwangiSubasingha/Nuwangi-s\_Portfolio

