Advanced Java [Day – 4]

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

Question 1: Library Management System:

Problem Statement:
A public library aims to digitize its catalog to manage books efficiently. The system should allow:

1. Adding New Books: When a new book is acquired, its details should be saved.
2. Retrieving Book Details: Fetch details of a book using its unique ID.
3. Updating Book Information: Modify existing book details, such as updating the number of available copies.
4. Deleting Book Records: Remove records of books that are no longer in circulation.

Code:

DbConnection.java

```
package com.library;

import java.sql.*;

public class DBUtil {
   public static Connection getConnection() throws Exception {
      Class.forName("com.mysql.cj.jdbc.Driver");
      return DriverManager.getConnection(
         "jdbc:mysql://localhost:3306/LibraryDB", "root", "12345");
   }
}
```

AddBookServlet.java

```
package com.library;
import java.io.*;
import jakarta.servlet.http.*;
import java.sql.*;
public class AddBookServlet extends HttpServlet {
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException {
      try (Connection con = DBUtil.getConnection()) {
         PreparedStatement ps = con.prepareStatement("INSERT INTO books (title, author, publisher,
available_copies) VALUES (?, ?, ?, ?)");
         ps.setString(1, req.getParameter("title"));
         ps.setString(2, req.getParameter("author"));
         ps.setString(3, req.getParameter("publisher"));
         ps.setInt(4, Integer.parseInt(req.getParameter("copies")));
         ps.executeUpdate();
         res.sendRedirect("index.jsp");
      } catch (Exception e) {
         res.getWriter().println("Error: " + e.getMessage());
      }
```

```
    }
}
```

ViewBookServlet.java

```
package com.library;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.sql.*;
public class ViewBookServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException,
ServletException {
        int id = Integer.parseInt(req.getParameter("id"));
        try (Connection con = DBUtil.getConnection()) {
            PreparedStatement ps = con.prepareStatement("SELECT * FROM books WHERE id = ?");
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                Book book = new Book();
                book.setId(rs.getInt("id"));
                book.setTitle(rs.getString("title"));
                book.setAuthor(rs.getString("author"));
                book.setPublisher(rs.getString("publisher"));
                book.setAvailableCopies(rs.getInt("available_copies"));
                req.setAttribute("book", book);
                RequestDispatcher rd = req.getRequestDispatcher("viewBook.jsp");
                rd.forward(req, res);
            } else {
                res.getWriter().println("Book not found.");
            }
        } catch (Exception e) {
            res.getWriter().println("Error: " + e.getMessage());
        }
    }
}
```

DeleteBookServlet.java

```
package com.library;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.sql.*;
public class DeleteBookServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException {
        int id = Integer.parseInt(req.getParameter("id"));
        try (Connection con = DBUtil.getConnection()) {
            PreparedStatement ps = con.prepareStatement("DELETE FROM books WHERE id=?");
            ps.setInt(1, id);
            ps.executeUpdate();
            res.sendRedirect("index.jsp");
        } catch (Exception e) {
            res.getWriter().println("Error: " + e.getMessage());
        }
    }
```

Question 2: Traffic Light Controlled Intersection

Requirements:

A. Design a traffic light system at an intersection where two roads cross.
B. Only one road can have a green light at a time to prevent accidents.
C. Cars arrive at the intersection and must wait if their road's light is red.

Explanation: Each car is represented as a thread extending the Thread class. The traffic light system controls access to the intersection, allowing cars to pass only when their road has a green light.

Code:

Car.java

```
package traffic;
public class Car extends Thread {
    private String carName;
    private String road;
    private TrafficLightController controller;
    public Car(String carName, String road, TrafficLightController controller) {
        this.carName = carName;
        this.road = road;
        this.controller = controller;
    }
    @Override
    public void run() {
        try {
            controller.enterIntersection(carName, road);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

TrafficLightController.java

```
package traffic;
public class TrafficLightController {
    private String greenRoad = "A";
    private final Object lock = new Object();
    public void enterIntersection(String carName, String road) throws InterruptedException {
        synchronized (lock) {
            while (!greenRoad.equals(road)) {
                System.out.println(" 🚗 " + carName + " waiting on Road " + road + " (Red Light)");
                lock.wait();
            }
            System.out.println(" ✅ " + carName + " is passing through Road " + road + " (Green Light)");
```

```
            Thread.sleep(1000);
        }
    }
    public void switchLight() {
        synchronized (lock) {
            greenRoad = greenRoad.equals("A") ? "B" : "A";
            System.out.println("\n 🚦 Traffic light switched: Now GREEN for Road " + greenRoad + "\n");
            lock.notifyAll();  // Wake up waiting cars
        }
    }
}
```

Main.java

```
package traffic;
public class Main {
    public static void main(String[] args) {
        TrafficLightController controller = new TrafficLightController();
        new Thread(() -> {
            while (true) {
                try {
                    Thread.sleep(5000);
                    controller.switchLight();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }).start();
        String[] roads = {"A", "B"};
        for (int i = 1; i <= 10; i++) {
            String road = roads[i % 2];
            Car car = new Car("Car-" + i, road, controller);
            car.start();
            try {
                Thread.sleep(1000 + (int)(Math.random() * 2000));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```