Advanced Java [Day – 2]

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

Question 1: Inter-Thread Communication: wait(), notify(), notifyAll()

Producer-Consumer Problem

1. **One thread (producer) generates data and adds it to a shared queue, while another thread (consumer) retrieves data from the queue.**
2. **Proper synchronization ensures that the producer doesn't add data when the queue is full, and the consumer doesn't remove data when the queue is empty.**

Code:

```
class SharedQueue {
    private Queue<Integer> queue = new LinkedList<>();
    private final int CAPACITY = 5;
    public synchronized void produce(int value) throws InterruptedException {
        while (queue.size() == CAPACITY) {
            System.out.println("Queue is full. Producer is waiting...");
            wait();
        }
        queue.add(value);
        System.out.println("Produced: " + value);
        notify();
    }
    public synchronized void consume() throws InterruptedException {
        while (queue.isEmpty()) {
            System.out.println("Queue is empty. Consumer is waiting");
            wait();
        }
        int value = queue.poll();
        System.out.println("Consumed: " + value);
        notify();
    }
}
class Producer extends Thread {
    private SharedQueue queue;

    public Producer(SharedQueue queue) {
        this.queue = queue;
    }
    public void run() {
        int value = 0;
        try {
            while (true) {
                queue.produce(value++);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
class Consumer extends Thread {
    private SharedQueue queue;
    public Consumer(SharedQueue queue) {
        this.queue = queue;
    }
    public void run() {
        try {
            while (true) {
                queue.consume();
                Thread.sleep(1500);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
class ProducerConsumerDemo {
    public static void main(String[] args) {
        SharedQueue queue = new SharedQueue();
        Producer producer = new Producer(queue);
        Consumer consumer = new Consumer(queue);
        producer.start();
        consumer.start();
    }
}
```

Question 2: Write a java servlet that accepts user feedback via an HTML form stores the feedback in the users session, and display a personalized message.

Code:

HTML Code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Feedback Form</title>
</head>
<body>
    <div class="container">
        <h2>User Feedback</h2>
        <form action="FeedbackServlet" method="post">
            <label for="name">Your Name:</label>
            <input type="text" name="name" required>
            <label for="feedback">Your Feedback:</label>
            <textarea name="feedback" rows="5" cols="40" required></textarea>
        <input type="submit" value="Submit Feedback">
      </form>
   </div>
</body>
</html>
```

Servlet Code:

```
package com.feedback;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import jakarta.servlet.annotation.WebServlet;
@WebServlet("/FeedbackServlet")
public class FeedbackServlet extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      String name = request.getParameter("name");
      String feedback = request.getParameter("feedback");
      HttpSession session = request.getSession();
      session.setAttribute("username", name);
      session.setAttribute("userFeedback", feedback);
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<!DOCTYPE html>");
      out.println("<html><head><title>Feedback Received</title>");
      out.println("<style>");
      out.println("body { font-family: 'Segoe UI', sans-serif; background-color: #f8f8f8; margin: 0; padding: 0;
display: flex; height: 100vh; align-items: center; justify-content: center; }");
      out.println(".card { background: #fff; padding: 30px 40px; border-radius: 10px; box-shadow: 0 8px 16px
rgba(0,0,0,0.1); width: 450px; }");
      out.println("h2 { color: #333; margin-top: 0; }");
      out.println("blockquote { background: #f1f1f1; padding: 10px 15px; border-left: 4px solid #4CAF50; }");
      out.println("</style>");
      out.println("</head><body>");
      out.println("<div class='card'>");
      out.println("<h2>Thank you, " + name + "!</h2>");
      out.println("<p>Your feedback has been recorded as:</p>");
      out.println("<blockquote>" + feedback + "</blockquote>");
      out.println("</div>");
out.println("</body></html>");}}
```