CS2002-1

Lab Programs by

Harshith

NNM24IS092

Submitted to: Dr. Martis

QUESTION: Scenario
In your college notice board system, one thread produces messages (like announcements), and another thread consumes them for display. The producer and consumer must coordinate using wait/notify.

Problem Statement

1. Create a class MessageBoard with:
   - A private String message.
   - A boolean field hasMessage.
   - Method put(String msg) that waits if hasMessage is true, stores the message, sets hasMessage = true, and calls notify().
   - Method get() that waits if hasMessage is false, retrieves the message, sets hasMessage = false, and calls notify().
2. Create a Producer thread that sends 3 messages: "Exam on Monday", "Holiday on Tuesday", "Workshop on Wednesday".
3. Create a Consumer thread that prints each received message.
4. In the Main class, run both threads together.

Github Link: https://github.com/Harshith161/Java-Progs

Code:

```
 package producerconsumer;


class MessageBoard {

  private String message;

  private boolean hasMessage = false;


  public synchronized void put(String msg) {

    while (hasMessage) {

      try {
            wait();
         } catch (InterruptedException e) {
           Thread.currentThread().interrupt();
           }
       }
       message = msg;
       hasMessage = true;
       System.out.println("Producer sends: " + msg);
       notify();
   }

   public synchronized String get()
   {
       while (!hasMessage)
       {
           try {
               wait();
           } catch (InterruptedException e) {
           Thread.currentThread().interrupt();
           }
       }
       String msg = message;
       hasMessage = false;
       notify();
       return msg;
   }
}

class Producer extends Thread {
    private MessageBoard board;


    public Producer(MessageBoard b) {
      this.board = b;
    }


    @Override
    public void run() {
      String[] msgs = {
        "Exam on Monday",
        "Holiday on Tuesday",
        "Workshop on Wednesday"
      };


      for (String msg : msgs)
      {
        board.put(msg);
        try
        {
          Thread.sleep(1000);
        } catch (InterruptedException e) {
         Thread.currentThread().interrupt();
        }
      }
      board.put("DONE");
    }
  }


  class Consumer extends Thread {
    private MessageBoard board;

    public Consumer(MessageBoard b) {
        this.board = b;
    }

    @Override
    public void run() {
        String msg;
        while (!(msg = board.get()).equals("DONE")) {
            System.out.println("Consumer reads: " + msg);
        }
    }
}

public class ProducerConsumerDemo
{
    public static void main(String[] args) throws InterruptedException {
        MessageBoard board = new MessageBoard();
        Producer p = new Producer(board);
        Consumer c = new Consumer(board);

        p.start();
        c.start();

        p.join();
        c.join();
    }
}
```

Output:

```
Producer sends: Exam on Monday

Consumer reads: Exam on Monday

Producer sends: Holiday on Tuesday

Consumer reads: Holiday on Tuesday

Producer sends: Workshop on Wednesday

Consumer reads: Workshop on Wednesday

Producer sends: DONE
```