ASSIGNMENT

WEEK 7

Q1) Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

CODE:


```
import java.util.Random;

public class MultiThreadExample {
    public static void main(String[] args) {
        RandomNumberGenerator generator = new RandomNumberGenerator();
        SquareCalculator squareCalculator = new SquareCalculator(generator);
        CubeCalculator cubeCalculator = new CubeCalculator(generator);

        Thread thread1 = new Thread(generator);
        Thread thread2 = new Thread(squareCalculator);
        Thread thread3 = new Thread(cubeCalculator);

        thread1.start();
        thread2.start();
        thread3.start();
    }
}
```

```
class RandomNumberGenerator implements Runnable {
    private Random random = new Random();

    @Override
    public void run() {
        try {
            while (true) {
                int randomNumber = random.nextInt(100); // Generate a random
integer between 0 and 99
                System.out.println("Generated: " + randomNumber);

                if (randomNumber % 2 == 0) {
                    SquareCalculator.queue(randomNumber);
                } else {
                    CubeCalculator.queue(randomNumber);
                }

                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

class SquareCalculator implements Runnable {
    private static int number;
```

```
    public SquareCalculator(RandomNumberGenerator generator) {
        generator.setSquareCalculator(this);
    }

    public static synchronized void queue(int num) {
        number = num;
    }

    @Override
    public void run() {
        try {
            while (true) {
                if (number % 2 == 0) {
                    int square = number * number;
                    System.out.println("Square: " + square);
                }
                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

class CubeCalculator implements Runnable {
    private static int number;

    public CubeCalculator(RandomNumberGenerator generator) {
        generator.setCubeCalculator(this);
```

```
    }

    public static synchronized void queue(int num) {
        number = num;
    }

    @Override
    public void run() {
        try {
            while (true) {
                if (number % 2 != 0) {
                    int cube = number * number * number;
                    System.out.println("Cube: " + cube);
                }
                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Output:

```
Generated: 42
Generated: 89
Generated: 17
Generated: 58
Generated: 5
Generated: 33
Square: 1764
Cube: 704969
Square: 27889
Square: 3364
Cube: 125
Square: 1089
Generated: 91
Generated: 66
Generated: 27
Square: 4356
Cube: 19683
Square: 4356
Cube: 729
Square: 729
```

Q2) Write a java program for to solve producer consumer problem in which a producer produces a value and consumer consume the value before producer generate the next value.

CODE:

```
import java.util.LinkedList;

public class ProducerConsumer {
    public static void main(String[] args) {
        Buffer buffer = new Buffer(5); // Create a buffer of size 5
        Thread producerThread = new Thread(new Producer(buffer));
        Thread consumerThread = new Thread(new Consumer(buffer));

        producerThread.start();
        consumerThread.start();
    }
}

class Buffer {
    private LinkedList<Integer> buffer;
    private int maxSize;

    public Buffer(int maxSize) {
        this.buffer = new LinkedList<>();
        this.maxSize = maxSize;
    }

    public synchronized void produce(int item) throws InterruptedException {
        while (buffer.size() == maxSize) {
```

```
            wait(); // Wait if the buffer is full
        }
        buffer.add(item);
        System.out.println("Produced: " + item);
        notify(); // Notify waiting consumers
    }

    public synchronized int consume() throws InterruptedException {
        while (buffer.isEmpty()) {
            wait(); // Wait if the buffer is empty
        }
        int item = buffer.removeFirst();
        System.out.println("Consumed: " + item);
        notify(); // Notify waiting producers
        return item;
    }
}

class Producer implements Runnable {
    private Buffer buffer;

    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }


    public void run() {
        try {
            for (int i = 1; i <= 10; i++) {
```

```
            buffer.produce(i);
            Thread.sleep(1000); // Simulate production time
        }
      } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
      }
    }
}

class Consumer implements Runnable {
    private Buffer buffer;

    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 10; i++) {
                int item = buffer.consume();
                Thread.sleep(1500); // Simulate consumption time
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

OUTPUT

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
```

Q3) Write a java program in which thread sleep for 5 sec and change the name of thread.

CODE:

```
public class ThreadNameChangeExample {
    public static void main(String[] args) {
        Thread myThread = new Thread(new MyRunnable());

        myThread.start();

        try {
            Thread.sleep(5000); // Sleep for 5 seconds
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        myThread.setName("NewThreadName"); // Change the thread's name

        // Display the updated thread name
        System.out.println("Thread name after change: " +
myThread.getName());
    }
}

class MyRunnable implements Runnable {
    @Override
    public void run() {
        // Display the initial thread name
```

```
        System.out.println("Thread name before change: " +
Thread.currentThread().getName());
    }
}
```

OUTPUT:

```
Thread name before change: Thread-0
Thread name after change: NewThreadName
```

Q4) Write a java program in which thread sleep for 6 sec in the loop in reverse order from 5 to 1 and change the name of thread.

CODE:

```
public class ReverseSleepAndRenameThreads {
    public static void main(String[] args) {
        for (int i = 5; i >= 1; i--) {
            Thread thread = new Thread(new MyRunnable(i));
            thread.start();

            try {
                Thread.sleep(6000); // Sleep for 6 seconds
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

class MyRunnable implements Runnable {
    private int threadNumber;

    public MyRunnable(int threadNumber) {
        this.threadNumber = threadNumber;
    }

    @Override
    public void run() {
        // Display the initial thread name
```

```
        System.out.println("Thread " + threadNumber + " - Initial Name: " +
Thread.currentThread().getName());

        // Change the thread name
        Thread.currentThread().setName("NewThreadName" +
threadNumber);

        // Display the updated thread name
        System.out.println("Thread " + threadNumber + " - Updated Name: "
+ Thread.currentThread().getName());
    }
}
```

OUTPUT:

```
Thread 5 - Initial Name: Thread-5
Thread 5 - Updated Name: NewThreadName5
Thread 4 - Initial Name: Thread-4
Thread 4 - Updated Name: NewThreadName4
Thread 3 - Initial Name: Thread-3
Thread 3 - Updated Name: NewThreadName3
Thread 2 - Initial Name: Thread-2
Thread 2 - Updated Name: NewThreadName2
Thread 1 - Initial Name: Thread-1
Thread 1 - Updated Name: NewThreadName1
```

Q5) Write a java program for multithread in which user thread and thread started from main method invoked at a time each thread sleep for 1 sec.

CODE

```
public class MultiThreadExample {
    public static void main(String[] args) {
        // Create and start a user-defined thread
        Thread userThread = new MyUserThread();
        userThread.start();

        // Create and start a thread from the main method
        Thread mainThread = new Thread(new MyRunnable());
        mainThread.start();

        try {
            // Sleep for 1 second in the main thread to ensure that both threads run sequentially
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        System.out.println("Main method has completed.");
    }
}

class MyUserThread extends Thread {
    public void run() {
```

```
        System.out.println("User-defined thread is running.");
        try {
            Thread.sleep(1000); // Sleep for 1 second
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("User-defined thread has completed.");
    }
}

class MyRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("Thread from main method is running.");
        try {
            Thread.sleep(1000); // Sleep for 1 second
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("Thread from main method has completed.");
    }
}
```

OUTPUT:

```
User-defined thread is running.
User-defined thread has completed.
Thread from main method is running.
Thread from main method has completed.
Main method has completed.
```

Q6) Write a java program to solve printer synchronization problem in which all the jobs must be completed in order.

CODE

```
public class PrinterSyncProblem {
    private static final Object lock = new Object();
    private static int currentJob = 1;

    public static void main(String[] args) {
        Thread[] jobThreads = new Thread[5]; // Create 5 job threads

        for (int i = 0; i < jobThreads.length; i++) {
            jobThreads[i] = new Thread(new Job(i + 1));
            jobThreads[i].start();
        }
    }

    static class Job implements Runnable {
        private int jobId;

        public Job(int jobId) {
            this.jobId = jobId;
        }

        @Override
        public void run() {
            synchronized (lock) {
                while (jobId != currentJob) {
                    try {
                        lock.wait(); // Wait until it's this job's turn
```

```
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }

        // Print the job
        System.out.println("Job " + jobId + " is printing.");

        // Increment currentJob for the next job to print
        currentJob++;

        // Notify other threads waiting on the lock
        lock.notifyAll();
    }
  }
}
```


OUTPUT

```
Job 1 is printing.
Job 2 is printing.
Job 3 is printing.
Job 4 is printing.
Job 5 is printing.
```

Q7) Create a java program for the following

Use ThreadA to find number of digits present in the string k and store into variable dc, finally print the value of dc(output format: ThreadA:digitscount). Use ThreadB to find number of alphabetic present in the string k and store into variable cc, finally print the value of cc(output format:ThreadB:digitscount).

CODE:

```
public class ThreadCountExample {
    public static void main(String[] args) {
        String k = "Hello123World456";

        // Create and start ThreadA to count digits
        ThreadA threadA = new ThreadA(k);
        threadA.start();

        // Create and start ThreadB to count alphabetic characters
        ThreadB threadB = new ThreadB(k);
        threadB.start();

        try {
            // Wait for both threads to complete
            threadA.join();
            threadB.join();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        // Print the results
        System.out.println("ThreadA:" + threadA.getCount());
        System.out.println("ThreadB:" + threadB.getCount());
```

```
    }
}

class ThreadA extends Thread {
    private String k;
    private int dc; // Digits count

    public ThreadA(String k) {
        this.k = k;
    }
    public void run() {
        dc = countDigits(k);
    }

    public int getCount() {
        return dc;
    }

    private int countDigits(String s) {
        int count = 0;
        for (char c : s.toCharArray()) {
            if (Character.isDigit(c)) {
                count++;
            }
        }
        return count;
    }
}

class ThreadB extends Thread {
```

```
    private String k;
    private int cc; // Alphabetic characters count

    public ThreadB(String k) {
        this.k = k;
    }

    @Override
    public void run() {
        cc = countAlphabeticChars(k);
    }

    public int getCount() {
        return cc;
    }

    private int countAlphabeticChars(String s) {
        int count = 0;
        for (char c : s.toCharArray()) {
            if (Character.isLetter(c)) {
                count++;
            }
        }
        return count;
    }
}
```

OUTPUT:

```
ThreadA:6
ThreadB:10
```

Q8) Create two objects threadobj1 and threadobj2 for the class UserThreadPriority. Set the name of threadobj1 as “ThreadA” and threadobj2 as “ThreadB”. Get a String and a Character from the user and assign into UserThreadPriority class variable k and c respectively. Call the start() method for the thread objects threadobj1 and threadobj2.

CODE:

```
import java.util.Scanner;

public class UserThreadPriorityExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get a String from the user
        System.out.print("Enter a String: ");
        String inputString = scanner.nextLine();

        // Get a Character from the user
        System.out.print("Enter a Character: ");
        char inputChar = scanner.next().charAt(0);

        // Create threadobj1 and set its name to "ThreadA"
        Thread threadobj1 = new Thread(new
UserThreadPriority(inputString, inputChar));
        threadobj1.setName("ThreadA");

        // Create threadobj2 and set its name to "ThreadB"
        Thread threadobj2 = new Thread(new
UserThreadPriority(inputString, inputChar));
        threadobj2.setName("ThreadB");
```

```
        // Start both threads
        threadobj1.start();
        threadobj2.start();

        scanner.close();
    }
}

class UserThreadPriority implements Runnable {
    private String k; // String
    private char c;  // Character

    public UserThreadPriority(String k, char c) {
        this.k = k;
        this.c = c;
    }

    @Override
    public void run() {
        // Print the values of k and c
        System.out.println(Thread.currentThread().getName() + ": k = " + k +
", c = " + c);
    }
}
```


OUTPUT:

```
Enter a String: Hello
Enter a Character: X
ThreadA: k = Hello, c = X
ThreadB: k = Hello, c = X
```

Q9) Write java program using sleep() method of Thread class where the new class thread created from the previous class is implemented by using sleep method for 10,20,50,70,100 ns

CODE:

```
public class SleepingThreadExample {
    public static void main(String[] args) {
        // Create and start threads with different sleep durations
        Thread thread1 = new SleepingThread("Thread1", 10_000); // 10 ns
        Thread thread2 = new SleepingThread("Thread2", 20_000); // 20 ns
        Thread thread3 = new SleepingThread("Thread3", 50_000); // 50 ns
        Thread thread4 = new SleepingThread("Thread4", 70_000); // 70 ns
        Thread thread5 = new SleepingThread("Thread5", 100_000); // 100 ns

        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        thread5.start();
    }
}

class SleepingThread extends Thread {
    private long sleepDurationNs;

    public SleepingThread(String name, long sleepDurationNs) {
        super(name);
        this.sleepDurationNs = sleepDurationNs;
    }
```

```
    public void run() {
        try {
            System.out.println(getName() + " is sleeping for " +
sleepDurationNs + " ns.");
            Thread.sleep(0, (int) sleepDurationNs); // Sleep for the specified
nanoseconds
            System.out.println(getName() + " woke up.");
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

OUTPUT:

```
Thread1 is sleeping for 10000 ns.
Thread3 is sleeping for 50000 ns.
Thread2 is sleeping for 20000 ns.
Thread4 is sleeping for 70000 ns.
Thread5 is sleeping for 100000 ns.
Thread1 woke up.
Thread2 woke up.
Thread3 woke up.
Thread4 woke up.
Thread5 woke up.
```

Q10) Write a java Thread program using Thread Priority for 5 processes and execute the priority order among the process.

CODE:

```
public class ThreadPriorityExample {
    public static void main(String[] args) {
        // Create and start five threads with different priorities
        Thread thread1 = new PriorityThread("Process 1", Thread.MAX_PRIORITY); // Priority 10
        Thread thread2 = new PriorityThread("Process 2", Thread.NORM_PRIORITY); // Priority 5
        Thread thread3 = new PriorityThread("Process 3", Thread.MIN_PRIORITY); // Priority 1
        Thread thread4 = new PriorityThread("Process 4", Thread.NORM_PRIORITY); // Priority 5
        Thread thread5 = new PriorityThread("Process 5", Thread.MAX_PRIORITY); // Priority 10

        // Start the threads
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        thread5.start();
    }
}

class PriorityThread extends Thread {
    public PriorityThread(String name, int priority) {
        super(name);
        setPriority(priority);
```

```
    }
    public void run() {
        System.out.println(getName() + " is running with priority " +
getPriority());
    }
}
```

OUTPUT:

```
Process 1 is running with priority 10
Process 5 is running with priority 10
Process 2 is running with priority 5
Process 4 is running with priority 5
Process 3 is running with priority 1
```