Assignment – 2

write a program which implements threads with locks by using synchronized keyword ,

```
package internship;

class Table {
	synchronized void printTable(int n) {// synchronized method
		for (int i = 1; i <= 5; i++) {
			System.out.println(n * i);
			try {
				Thread.sleep(400);
			} catch (Exception e) {
				System.out.println(e);
			}
		}

	}
}

class MyThread1 extends Thread {
	Table t;

	MyThread1(Table t) {
		this.t = t;
	}

	public void run() {
		t.printTable(5);
	}

}

class MyThread2 extends Thread {
	Table t;

	MyThread2(Table t) {
		this.t = t;
	}

	public void run() {
		t.printTable(100);
	}
}

public class demo {
	public static void main(String args[]) {
		Table obj = new Table();// only one object
		MyThread1 t1 = new MyThread1(obj);
		MyThread2 t2 = new MyThread2(obj);
		t1.start();
		t2.start();
	}
}
```

write a program which elaborates the concept of producer consumer problem using wait() ,notify() & all required functionalities in it.

```
package Section3;

class Utility {
	int i;
	boolean bool = false;

	public synchronized void set(int i) throws InterruptedException {
		while (bool) {
			wait();
		}
		this.i = i;
		bool = true;
		System.out.println("Producer " + i);
		notify();
	}

	public synchronized void get() throws InterruptedException {
		while (!bool) {
			wait();
		}

		bool = false;
		System.out.println("Consumer " + i);
		notify();
	}
}

class Consumer implements Runnable {
	private Utility utility;

	public Consumer(Utility utility) {
		this.utility = utility;
		Thread consumer = new Thread(this, "Consumer");
		consumer.start();
	}

	public void run() {
		while (true) {
			try {
				utility.get();
				Thread.sleep(500);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}

}

class Producer implements Runnable {
	private Utility utility;

	public Producer(Utility utility) {
		this.utility = utility;
		Thread producer = new Thread(this, "Producer");
		producer.start();
	}

	public void run() {
		int i = 0;
		while (true) {
			try {
				utility.set(i++);
				Thread.sleep(1000);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}

}

public class ProducerConsumer {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Utility u = new Utility();
		new Producer(u);
		new Consumer(u);
	}
}
```

Write a program with data structure ,use atomic methods like get(),incrementAndGet(),decrementAndGet(),compareAndSet(),etc ,also use all other functionalities to make the program more responsive.

```
package Section3;

import java.util.concurrent.atomic.AtomicInteger;

public class Assignment3 {

	public static void main(String[] args) {
		AtomicInteger val = new AtomicInteger(4);

		System.out.println("\nGet\n" + val.get());

		System.out.println("Increment and get");
		System.out.println("Previous value: " + val);
		int res = val.incrementAndGet();
		System.out.println("Current value: " + res);

		System.out.println("\nDecrenent and get");
		System.out.println("Previous value: " + val);
		res = val.decrementAndGet();
		System.out.println("Current value: " + res);

		System.out.println("\nCompare And Set");
		System.out.println("\nWhen expected value match");
		val.compareAndSet(4, 77);
		System.out.println("the updated number: " + val);
		System.out.println("\nWhen expected value don't match");
		val.compareAndSet(4, 88);
		System.out.println("the updated number: " + val);

	}

}
```


TECHADEMY


TECHADEMY