

Name: Vaishali Jadhav

Phn no: 7030674246

Assignment no-8

Q1.Create a Deadlock class to demonstrate deadlock in multithreading environment

```
package vaishali;

public class Assignment8 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        final String resource1 = "vaishali adhav";
        final String resource2 = "rupali yadav";
        Thread t1 = new Thread() {
            public void run() {
                synchronized (resource1) {
                    System.out.println("Thread 1: locked resource 1");

                    try { Thread.sleep(100);} catch (Exception e) {}

                    synchronized (resource2) {
                        System.out.println("Thread 1: locked resource 2");
                    }
                }
            }
        };
        Thread t2 = new Thread() {
            public void run() {
                synchronized (resource2) {
                    System.out.println("Thread 2: locked resource 2");

                    try { Thread.sleep(100);} catch (Exception e) {}

                    synchronized (resource1) {
                        System.out.println("Thread 2: locked resource 1");
                    }
                }
            }
        };
        t1.start();
        t2.start();
    }

    public void startThreads() {
        // TODO Auto-generated method stub
    }
}
```

Output:

Thread 1: locked resource 1
Thread 2: locked resource 2

Q2. Implement wait, notify, notify All methods.

```
package vaishali;

public class Assignment_8 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        {
            //Assignment_8.Customer();
            final Customer c=new Customer();
            new Thread()
            {
                public void run()
                {c.withdraw(15000);}
            }.start();

            new Thread()
            {
                public void run()
                {c.deposit(10000);}
            }.start();
        }
    }
}

class Customer
{
    int amount = 60000;

    synchronized void withdraw(int amount)
    {
        System.out.println("going to withdraw...");

        if (this.amount < amount)
        {
            System.out.println("Less balance; waiting for
deposit...");

            try
            {
                wait();
            }
            catch (Exception e)
            {}
        }
        this.amount -= amount;
        System.out.println("withdraw completed...");
    }

    synchronized void deposit(int amount) {
        System.out.println("going to deposit...");
        this.amount += amount;
        System.out.println("deposit completed... ");
        notify();
    }
}
```

```

    }
}

```

Output:

```

going to withdraw...
withdraw completed...
going to deposit...
deposit completed...

```

Q3.Demonstrate how to share threadLocal data between multiple threads

```

package vaishali;

public class assign_8 {
    public static class MyRunnable implements Runnable
    {
        private ThreadLocal<Integer> threadLocal =
            new ThreadLocal<Integer>();
        @Override
        public void run() {
            threadLocal.set( (int) (Math.random() * 100) );
            try
            {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
            }
            System.out.println(threadLocal.get());
        }
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MyRunnable runnableInstance = new MyRunnable();

        Thread t1 = new Thread(runnableInstance);
        Thread t2 = new Thread(runnableInstance);
        t1.start();
        t2.start();
    }
}

```

Output:

```

7
7

```

Q4. Creates multiple threads in anonymous inner classes.

```

package vaishali;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

```

```

public class Assi8 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new Assignment8().startThreads();
    }

    private void startThreads()
    {
        ExecutorService taskList
            = Executors.newFixedThreadPool(2);

        taskList.execute(new InnerClass(1));
        taskList.execute(new InnerClass(2));
        taskList.execute(new InnerClass(3));
        taskList.execute(new InnerClass(4));
        taskList.execute(new InnerClass(5));
        taskList.shutdown();
    }

    private void pause(double seconds)
    {
        try {
            Thread.sleep(Math.round(1000.0 * seconds));
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    // Inner Class
    public class InnerClass implements Runnable {

```

```
private int loopLimit;
InnerClass(int loopLimit)
{
    this.loopLimit = loopLimit;
}

public void run()
{
    for (int i = 0; i < loopLimit; i++) {
        System.out.println(
            Thread.currentThread().getName()
            + " Counter: " + i);
        pause(Math.random());
    }
}
}

}
```