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

```
package yasin;
public class Assignment_8 {
            public static void main(String[] args) {
              final String resource1 = "Yasin Tamboli";
              final String resource2 = "Tushar Dharekar";
              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();
            }

}
```

Output:

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

2 Implement wait , notify and notifyAll methods.

```
package yasin;
public class Assignment_8 {
        public static void main(String args[])
        {
                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...
```

3. Demonstrate how to share ThreadLocal data between multiple threads

```
package yasin;
public class Assignment_8 {
            public static class MyRunnable implements Runnable
              {
                 private ThreadLocal<Integer> threadLocal =
                       new ThreadLocal<Integer>();
                 @Override
                 public void run() {
                    threadLocal.set( (int) (Math.random() * 10D) );
                    try
                    {
                       Thread.sleep(1000);
                    } catch (InterruptedException e) {
                    }
                    System.out.println(threadLocal.get());
                 }
              }
              public static void main(String[] args)
              {
                 MyRunnable runnableInstance = new MyRunnable();

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

Output:

```
2
```

3

4. Create multiple threads using anonymous inner classes

```
package yasin;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
public class Assignment_8 {
public static void main(String[] args)
        {
                new  Assignment_6().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());
                        }
                }
        }
    }
```

Output:

```
pool-1-thread-1 Counter: 0
pool-1-thread-2 Counter: 0
pool-1-thread-2 Counter: 1
pool-1-thread-1 Counter: 0
pool-1-thread-2 Counter: 0
pool-1-thread-1 Counter: 1
pool-1-thread-1 Counter: 2
pool-1-thread-2 Counter: 1
pool-1-thread-2 Counter: 2
pool-1-thread-2 Counter: 3
pool-1-thread-1 Counter: 0
pool-1-thread-1 Counter: 1
pool-1-thread-1 Counter: 2
pool-1-thread-1 Counter: 3
pool-1-thread-1 Counter: 4
```