**Advanced Java: Multi-threading Part 4 -- Multiple Locks; Using Synchronized Code Blocks**

**2nd class – secondclass.java:**

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Random;

**public** **class** secondclass{

**private** Random random=**new** Random();

**private** Object lock1 = **new** Object();

**private** Object lock2 = **new** Object();

**private** List<Integer> list1 = **new** ArrayList<Integer>();

**private** List<Integer> list2 = **new** ArrayList<Integer>();

**public** **void** stageOne(){

**synchronized** (lock1){

**try** {

Thread.*sleep*(1);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

list1.add(random.nextInt(100));

}

}

**public** **void** stageTwo(){

**synchronized** (lock2){

**try** {

Thread.*sleep*(1);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

list2.add(random.nextInt(100));

}

}

**public** **void** process(){

**for**(**int** i=0;i<1000;i++){

stageOne();

stageTwo();

}

}

**public** **void** main(){

System.*out*.println("Starting ...");

**long** start = System.*currentTimeMillis*();

Thread t1 = **new** Thread(**new** Runnable(){

**public** **void** run() {

process();

}

});

Thread t2 = **new** Thread(**new** Runnable(){

**public** **void** run() {

process();

}

});

t1.start();

t2.start();

**try** {

t1.join();

t2.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**long** end = System.*currentTimeMillis*();

System.*out*.println("Time taken: "+(end-start));

System.*out*.println("List1: "+list1.size()+"List2:"

+list2.size());

}

}

**1st class – apples.java:**

**class** apples {

**public** **static** **void** main(String args[]){

**new** secondclass().main();

}

}

\_\_ -Remarkable codes

**Result:**

Starting ...

Time taken: 3183

List1: 2000 List2: 2000

**Explanation**: