1. Write a Java program that illustrates the use of the yield( ), start( ),stop( ),run ( ), sleep( ), wait( ) and isAlive( ) methods.

2. Write multithreaded program that continuously prints the strings “ping” and “PONG” in the console at random distances at intervals of one second. Use two threads one for “ping” and another for “PONG”.

3. Write a program to sort a list of strings using multithreading. Create one of the threads that take a string as input from the user, another thread that sorts the strings and finally another thread that displays the sorted list of strings. Each of the input, sort, and display methods is to be synchronized.

4. Inherit a class from Thread and override the run( ) method. Inside run( ), print a message, and then call sleep( ). Repeat this three times, then return from run( ). Put a start-up message in the constructor and override finalize( ) to print a shut-down message. Make a separate thread class that calls System.gc( ) and System.runFinalization( ) inside run( ), printing a message as it does so. Make several thread objects of both types and run them to see what happens.

5. Create two Thread subclasses, one with a run( ) that starts up, captures the reference of the second Thread object and then calls wait( ). The other class’ run( ) should call notifyAll( ) for the first thread after some number of seconds have passed, so the first thread can print a message.

6. Suppose that two threads “t1” and “t2” access a shared integer “x”. Thread “t1” indefinitely increases “x” and “t2” indefinitely prints the value of “x”. That is the threads run in an infinite loop. However, thread “t1” must not increase “x” till that value of “x” is printed by “t2” and “t2” must not print the same value of “x” more than once.

7. Define the classes for implementing “t1” and “t2”. Write appropriate methods for accomplishing the above.

8. Write a program, which will create a deadlock.