# Advanced Programming Language

## Assignment 7

Chinmay Dorge
LCS2020022

Q1)" Synchronization in Java is the capability to control the access of multiple threads to any
shared resource." Write a java program demonstrating thread synchronization using

1)Synchronization method
2)Synchronization block

Code: Q1.java

```
public class Q1 {
    Integer i = 0;
    synchronized int count1() {
        i++;
        return i;
    }

    int count2 () {
        synchronized (i) {
            i += 2;
        }
        return i;
    }

    public static void main(String[] args) {
        var obj = new Q1();
        Runnable r1 = new Runnable() {
            public void run(){
                System.out.println(obj.count1());
            }
        };
        Runnable r2 = new Runnable() {
            public void run(){
                System.out.println(obj.count2());
            }
        };
        Thread a = new Thread(r1);
        Thread b = new Thread(r1);
        Thread c = new Thread(r2);

        try {
            a.start();
            Thread.sleep(20);
            b.start();
            Thread.sleep(20);
            c.start();
            Thread.sleep(20);
            a.join();
            b.join();
```

```
            c.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q1> javac Q1.java
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q1> java Q1
1
2
4
```

Q2) Write a java program to demonstrate the use of static synchronized method. Create 3 threads say t1, t2 and t3 respectively and make use of static synchronization.

Code: Q2.java

```
class printNum
{
    public static synchronized void func(int n,int m) {
        for(int i=n;i<=m;i++){
            System.out.print(i + " ");
            try {
                Thread.sleep(500);
            }
            catch(Exception e){
                System.out.println(e);
            }
        }
        System.out.println();
    }
}
class t1 extends Thread {
    @Override
    public void run() {
        printNum.func(1,10);
    }
}
class t3 extends Thread {
    @Override
    public void run() {
        printNum.func(21,30);
    }
}
class t2 extends Thread {
    @Override
    public void run() {
        printNum.func(11,20);
    }
}
public class Q2 {
    public static void main(String[] args) {
        t1 t1 = new t1();
        t2 t2 = new t2();
```

```
        t3 t3 = new t3();
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output:

```
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q2> javac Q2.java
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q2> java Q2
11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10
21 22 23 24 25 26 27 28 29 30
```

Q3)" Deadlock in Java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. "Implement a java program that goes into deadlock by creating 4 threads t1, t2, t3 and t4. You should make use put print statements at appropriate positions to make it evident that your program has entered deadlock.

Code: Q3.java

```
public class Q3{
    static Object Lock_1 = new Object(), Lock_2 = new Object();
    public static void main(String[] args) {
        Thread t1 = new Thread(new Runnable() {
            public void run(){
                try{
                    System.out.println("t1 locking Lock_1.");
                    synchronized(Lock_1){
                        System.out.println("t1 has locked Lock_1");
                        Thread.sleep(1000);
                        System.out.println("t1 trying to lock Lock_2");
                        synchronized(Lock_2){
                            System.out.println("t1 has acquired both Lock_1
and Lock_2 (We will never reach here)");
                        }
                    }
                }catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
        });
        Thread t2 = new Thread(new Runnable() {
            public void run(){
                try{
                    System.out.println("t2 locking Lock_2.");
                    synchronized(Lock_2){
                        System.out.println("t2 has locked Lock_2");
                        Thread.sleep(1000);
                        System.out.println("t2 trying to lock Lock_1");
```

```
                        synchronized(Lock_1){
                            System.out.println("t1 has acquired both Lock_1
and Lock_2");
                        }
                    }
                }catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
        });
        Thread t3 = new Thread(new Runnable() {
            public void run(){
                try{
                    System.out.println("t3 trying to lock Lock_1");
                    Thread.sleep(1000);
                    synchronized(Lock_1){
                        System.out.println("t3 has locked Lock_1 (We will
never reach here)");
                    }
                }catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
        });
        Thread t4 = new Thread(new Runnable() {
            public void run(){
                try{
                    System.out.println("t4 trying to lock Lock_2.");
                    Thread.sleep(1000);
                    synchronized(Lock_2){
                        System.out.println("t4 has locked Lock_2 (We will
never reach here)");
                    }
                }catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
        });

        t1.start();
        t2.start();
        t3.start();
        t4.start();

        try{
            Thread.sleep(2000);
            if(t1.isAlive()) {
                System.out.println("t1 is in deadlock.");
            }
            if(t2.isAlive()) {
                System.out.println("t2 is in deadlock.");
            }
            if(t3.isAlive()) {
                System.out.println("t3 is in deadlock.");
            }
            if(t4.isAlive()) {
                System.out.println("t4 is in deadlock.");
            }
        } catch(InterruptedException e){
            e.printStackTrace();
        }
```

```
    }
}
```

Output:

```
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q3> java Q3
t2 locking Lock_2.
t1 locking Lock_1.
t4 trying to lock Lock_2.
t2 has locked Lock_2
t3 trying to lock Lock_1
t1 has locked Lock_1
t1 trying to lock Lock_2
t2 trying to lock Lock_1
t1 is in deadlock.
t2 is in deadlock.
t3 is in deadlock.
t4 is in deadlock.
```

Q4) Inter-thread communication or Co-operation is all about allowing synchronized threads to communicate with each other. Write a java program using three threads t1, t2 and t3 using Thread classes in separate files. Make use of them in your main class. Using the demonstrate the working of

1)wait()
2)notify()
3)notifyAll()

Code: FirstThread.java

```java
package Threads;
public class FirstThread extends Thread {
    public Object obj;
    public FirstThread(Object obj){
        this.obj = obj;
    }
    public void run(){
        synchronized (obj){
            System.out.println("Executing FirstThread");
            System.out.println("FirstThread will wait for SecondThread");
            try {
                obj.wait();
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            System.out.println("FirstThread completed");
            obj.notifyAll();
        }
```

```
    }
}
```

SecondThread.java

```
package Threads;
public class SecondThread extends Thread {
    public Object obj;
    public SecondThread(Object obj){
        this.obj = obj;
    }
    public void run(){
        synchronized (obj){
            System.out.println("Executing Second Thread");
            System.out.println("SecondThread will wait for ThirdThread");
            try{
                obj.wait();
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            System.out.println("SecondThread completed");
            obj.notify();
        }
    }
}
```

ThirdThread.java

```
package Threads;
public class ThirdThread extends Thread{
    public Object obj;
    public ThirdThread(Object obj){
        this.obj = obj;
    }
    public void run(){
        synchronized (obj){
            System.out.println("Executing Third Thread");
            obj.notify();
            System.out.println("ThirdThread completed");
        }
    }
}
```

Q4.java

```
import Threads.FirstThread;
import Threads.SecondThread;
import Threads.ThirdThread;
public class Q4 {
    public static void main(String[] args) {
        Object obj = new Object();
        FirstThread thread1 = new FirstThread(obj);
        SecondThread thread2 = new SecondThread(obj);
        ThirdThread thread3 = new ThirdThread(obj);

        thread1.start();
        thread2.start();
        thread3.start();

        try {
```

```
            thread3.join();
            Thread.sleep(100);
            thread1.interrupt();
            thread2.interrupt();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q4> javac Q4.java
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q4> java Q4
Executing FirstThread
FirstThread will wait for SecondThread
Executing Second Thread
SecondThread will wait for ThirdThread
Executing Third Thread
ThirdThread completed
FirstThread completed
SecondThread completed
```

Q5) The 3 methods provided by the Thread class for interrupting a thread

- public void interrupt()
- public static Boolean interrupted()
- public Boolean isInterrupted()

Write a java program that demonstrates the use of above three methods. Create four threads t1, t2, t3 and t4 and demonstrate. Also, handle the exception, when interrupt is called when threads when sleep() or wait() are already called on them.

Code: Q5.java

```
public class Q5 {
    public static void main(String[] args) {
        Thread t1 = new Thread(() -> {
            try {
                Thread.sleep(1000000);
            } catch (InterruptedException e) {
                System.out.println("interrupted first thread");
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    synchronized (this) {
                        this.wait();
                    }
```

```
                } catch (InterruptedException e) {
                    System.out.println("interrupted second thread");
                }
            }
        });
        Thread t3 = new Thread(() -> {
            try {
                Thread.sleep(100000);
            } catch (InterruptedException e) {
                System.out.println("interrupted returned " +
Thread.interrupted());
            }
        });
        Thread t4 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    synchronized (this) {
                        this.wait();
                    }
                } catch (InterruptedException e) {
                    System.out.println("interrupted returned " +
Thread.interrupted());
                }
            }
        });

        t1.start();
        t2.start();
        t3.start();
        t4.start();

        System.out.println("interrupting first thread:");
        t1.interrupt();
        System.out.println("t1.isInterrupted() = " + t1.isInterrupted());
        System.out.println();
        System.out.println("interrupting second thread:");
        t2.interrupt();
        System.out.println("t2.isInterrupted() = " + t2.isInterrupted());
        System.out.println();
        System.out.println("interrupting third thread:");
        t3.interrupt();
        System.out.println("t3.isInterrupted() = " + t3.isInterrupted());
        System.out.println();
        System.out.println("interrupting fourth thread:");
        t4.interrupt();
        System.out.println("t4.isInterrupted() = " + t4.isInterrupted());

    }
}
```

Output:

```
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q5> javac Q5.java
PS C:\Users\hp\Desktop\CLG\Third Sem\APL (assignments)\Assignment-7\Q5> java Q5
interrupting first thread:
interrupted first thread
t1.isInterrupted() = true

interrupting second thread:
interrupted second thread
t2.isInterrupted() = true

interrupting third thread:
t3.isInterrupted() = true

interrupting fourth thread:
interrupted returned false
t4.isInterrupted() = true
interrupted returned false
```

Note: The code has been sent with the zip file and is also available on GitHub. Repo link https://github.com/Chinmay-Dorge/Advanced-Programming-Assignments