SYNCHRONIZATION

Definition

- It is the coordination or understanding between two entities.
- In multithreading, it is usually used between threads.

Important Terms in Multithreading Synchronization

Resource Sharing

- It's when more than one thread accesses the same resource like a file, network connection, data objects, etc.
- If there's an object in the heap, it can be accessed by multiple threads.
- That object then becomes the shared resource.

Critical Section

- The lines of code in a thread which are accessing the shared resource/object is the critical section.
- It's said to be a piece of code which is critical.

Mutual Exclusion

- Mutual exclusion states that the accessing of one thread prevents the accessing of another.

Locking/Mutex

- Mutex is the variable used for locking the threads.
- If the mutex variable is set to zero, it is free or not occupied.
- When one thread's time period is finished, another thread cannot access the object as the mutex will not remain zero.
- Thread two can access the object if and only if the mutex is zero again.

- For every shared resource, there should be a lock applied by the thread itself.
- Here, the threads are responsible for mutual exclusion.
- Mutex was not useful as the threads would overlap each other if the mutex was not being locked by the first one.

Semaphore

- Semaphore was like an operating system (before the introduction of Java) to control the coordination of threads so they should not overlap.
- It was supported by the UNIX operating system.
- The semaphore creates a scenario where the thread occupying the object would signal the others after its work is finished.
- It creates a block queue where the upcoming thread is to be in a waiting state.
- The methods used here are `wait()` and `signal()`.
- Here, the operating system handles the mutual exclusion.

Monitor

- Here, the object itself takes responsibility for mutual exclusion.
- It can be achieved using object orientation.
- The complete mechanism is inside the object itself.
- The read and write method, the data, and the block queue belong to the shared object itself as it can be accessed by any of the threads at a particular time.
- Java ensures that only one thread is accessed at a time.

Example Program:

```
class MyData {
    void display(String str) {
        synchronized(this) {
            for(int i = 0; i < str.length(); i++) {
                System.out.print(str.charAt(i));
            }
        }
    }
}

class MyThread1 extends Thread {
    MyData d;
    MyThread1(MyData dat) { d = dat; }
    public void run() {
        d.display("Hello World");
    }
}

class MyThread2 extends Thread {
    MyData data;
    MyThread2(MyData dat) { data = dat; }
    public void run() {
        data.display("Welcome");
    }
}

class Test {
    public static void main(String[] args) {
        MyData d = new MyData();
        MyThread1 t1 = new MyThread1(d);
        MyThread2 t2 = new MyThread2(d);
        t1.start();
        t2.start();
    }
}
```

- The classes MyThread1 and MyThread2 access the data from the display class, which is the shared object.
- The synchronized method is used to execute the classes separately in the above example.

- The synchronized method prints the data separately from both threads; in other words, it prevents the data from overlapping.
- The for loop inside the synchronized block prints one character at a time.
- Another method for synchronization is to write the `synchronized` keyword at the method signature of the display class, so the whole method is synchronized.

Inter-thread Communication

- It's the communication between synchronized threads.
- It's the communication between a single producer thread and a consumer thread.
- Inter-thread communication refers to the synchronization between the producer thread and the consumer thread to access the write and read methods simultaneously.
- To achieve communication, `flag=true` and `flag=false` are used.

Race Condition

- Here, there is a single producer thread and multiple consumer threads.
- All consumers do not execute at once; they do so in a round-robin fashion.
- When the count is zero, it's the producer's turn.
- When the count is not zero, it's the consumers' turn.
- Since there is more than one consumer, any of the consumers can access it.
- The `notify()` method can open any thread here as they may not be in an order.
- The race condition states that if one thread is accessing the shared resource and all others are blocked, then once the thread has finished working, it will inform the others, and any of the threads may access the object, just like in a race.
- The count method is used to control the race.
- The race condition can be avoided by inter-thread communication.

Example Program:

```
class MyData {
    int value = 0;
    boolean flag = true;

    synchronized void set(int v) {
        while(!flag) {
            try { wait(); } catch(InterruptedException e) {}
        }
        value = v;
        flag = false;
        notify();
    }

    synchronized int get() {
        int x = 0;
        while(flag) {
            try { wait(); } catch(InterruptedException e) {}
        }
        x = value;
        flag = true;
        notify();
        return x;
    }
}

class Producer extends Thread {
    MyData d;
    Producer(MyData dat) { d = dat; }

    public void run() {
        int i = 1;
        while(true) {
            d.set(i);
            System.out.println("Producer: " + i);
            i++;
        }
    }
}

class Consumer extends Thread {
    MyData d;
    // Consumer class implementation...
}
```

In the above example:

- The inter-thread communication part is achieved by the programmer.
- The producer will be writing to the shared object, and the consumer will be reading from the shared object.
- The shared object has three things: the data or value given, the set method to write the data, and the get method to read the data.
- As the set and get methods are synchronized, only one thread is allowed to enter at a time.
- If the flag is true, it's the producer's turn, and if the flag is false, it's the consumer's turn.
- `wait()` and `notify()` are used to bring in the communication between the threads.