Thread Lifecycle:

3. Blocked/Waiting:

A thread enters this state when it is waiting for a resource or another thread to complete.

Step1:

```java
package blockedWaiting;

public class SharedResource {
    synchronized void doWork(){
        try{
            Thread.sleep(2000);  // Simulating some work
        }catch (InterruptedException e){
            System.out.println(e);
        }
    }
}
```

- The SharedResource class has a method doWork(), which is synchronized(only one thread can access this method at a time).

- Inside the method, the current running thread sleeps for 2 seconds to simulate some work.

Step2:

```java
package blockedWaiting;

public class MyThread extends Thread{
    SharedResource resource;

    MyThread(SharedResource resource){
        this.resource = resource;
    }

    public void run() {
        resource.doWork();
    }

}
```

- Mythread extends the thread class
- It takes SharedResource object as parameter in its constructor.
- In run() method, the thread calls the doWork() method on the SharedResource object.

```java
package blockedWaiting;

public class Main {
    public static void main(String[] args) {
        SharedResource resource = new SharedResource();
        MyThread t1 = new MyThread(resource);
        MyThread t2 = new MyThread(resource);

        t1.start();
        t2.start();

        try {
            Thread.sleep(100); // waits for threads to start
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        System.out.println("Thread t1 State: "+
t1.getState());

        System.out.println("Thread t2 State: "+
t2.getState());
    }
}
```

- In main method
- A shared resource object is created.
- Two threads t1 and t2 are created, both are sharing same SharedResource object.
- Both threads are started using start() method.
- The main thread sleeps for 100 milliseconds to ensure t1 and t2 have started executing.
- The states of t1 and t2 are printed.

o/p:

```
Thread t1 State: BLOCKED

Thread t2 State: TIMED_WAITING
```

1. Synchronization:
   - It ensures the thread safety.
2. States:
   - TIMED_WAITING: a thread is waiting for specific amount of time(using sleep())
   - BLOCKED: A thread is waiting to acquire a lock held by another thread.

t1:

Starts -> acquire the lock -> enters doWork() -> sleeps for 2 seconds

t2

Starts -> Tries to acquire lock -> Lock is held by t1 -> BLOCKED

Main Thread

Sleeps for 100ms -> Checks states of t1 and t2 -> print their states

4. Timed Waiting:

5. Terminated:

Complete Example of lifecycle:

```java
package threadLifecycle;

public class MyThread extends Thread{

    public void run(){
        try {
            System.out.println("Thread is running!!!!!!");
            Thread.sleep(1000); // Timed_Waiting State
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}
```

```java
package threadLifecycle;



public class Main {
    public static void main(String[] args) throws
InterruptedException{
        MyThread t1 = new MyThread();
        System.out.println("Thread state after creation: "+
t1.getState());

        t1.start();
        System.out.println("Thread after start(): "+
t1.getState());

        Thread.sleep(100); // Wait for t1 to enter the
timed_waiting state
        System.out.println("Thread state during sleep(): "+
t1.getState());

        t1.join(); // wait for t1 to finish
        System.out.println("Thread state after completion:
"+t1.getState());
    }
}
```

O/P:

```
Thread state after creation: NEW

Thread after start(): RUNNABLE

Thread is running!!!!!!

Thread state during sleep(): TIMED_WAITING

Thread state after completion: TERMINATED
```