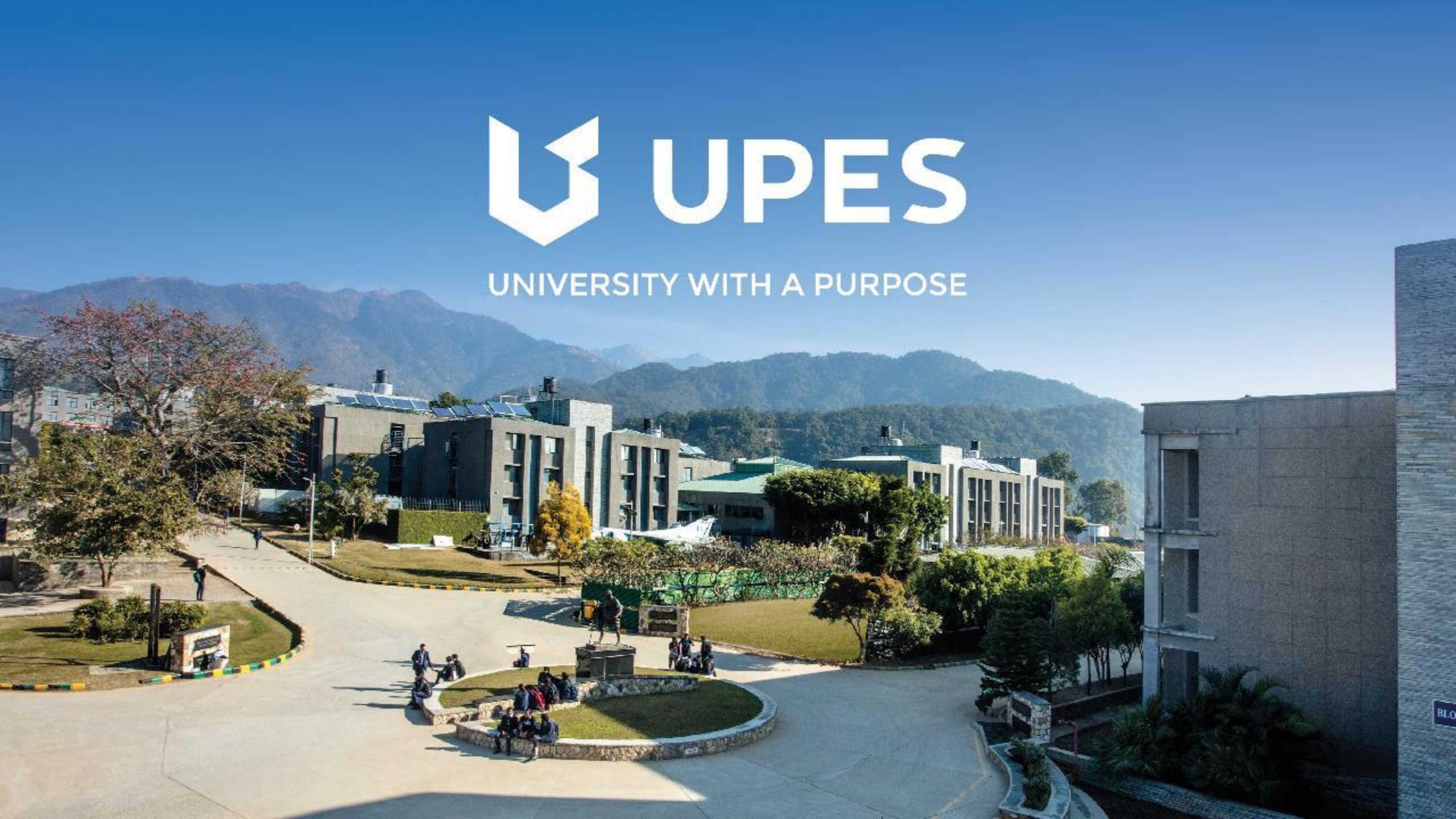

UPES
UNIVERSITY WITH A PURPOSE

Thread Synchronization

Thread States

A thread can be in one of the following states:

NEW: A thread that has not yet started is in this state.

RUNNABLE: A thread executing in the Java virtual machine is in this state.

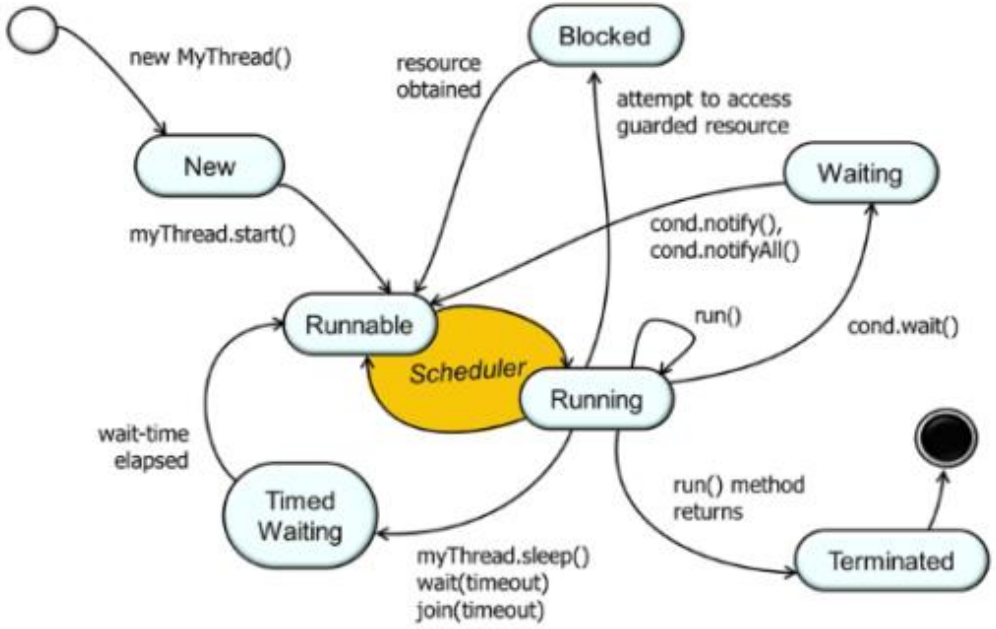
BLOCKED: A thread that is blocked waiting for a monitor lock is in this state.

WAITING: A thread that is waiting indefinitely for another thread to perform a particular action is in this state.

TIMED_WAITING: A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.

TERMINATED: A thread that has exited is in this state.

- A thread can be in only one state at a given point in time. These states are virtual machine states which do not reflect any operating system thread states.



Source:
https://bitstechnotes.wordpress.com/2017/12/16/java-thread-state-diagram/

Thread Synchronization

```
import java.util.Scanner;
class Account {
private int bal;
public Account(int bal) {
this.bal=bal;                    }

public boolean isSufficientBalance(int w) {
if (bal>w)
          return (true);
else
          return (false);                    }

public void withdraw(int amt) {
          bal=bal-amt;
          System.out.println("Withdrawal  amount  is:
" +amt);
          System.out.println("Current   balance   is:
"+bal);   }
}

class Customer implements Runnable {
private String name;
private Account account;

public Customer(Account account, String n) {
this.account=account;
name=n;                                          }
public void run() {
Scanner sc=new Scanner(System.in);
synchronized (account) { // To synchronize threads
for shared resource

System.out.println(name+"Enter amount to withdraw");
int amt=sc.nextInt();

if (account.isSufficientBalance(amt))
{
          System.out.println(name);
          account.withdraw(amt);
}
else
{
          System.out.println("Insufficient
balance");
}
}
}
}
public class ThreadSync{
public static void main(String[] args){
Account a1= new Account(1000);
Customer c1= new Customer(a1, "Ram");
Customer c2=new Customer(a1, "Shyam");
Thread t1=new Thread(c1);
Thread t2= new Thread(c2);
t1.start();
t2.start();
}   }
```

Inter-thread communication

- The synchronized locking mechanism suffices for keeping threads from interfering with each other, but you also need a way to communicate between threads.
- For this purpose, the **wait** method lets one thread wait until some condition occurs, and the notification methods **notifyAll** and **notify** tell waiting threads that something has occurred that might satisfy that condition.

wait() method: Causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

1. public final void wait(): waits until object is notified.
2. public final void wait(long timeout): waits for the specified amount of time.

Inter-thread communication contd..

`notify()` method: Wakes up a single thread that is waiting. If any thread is waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation.

Syntax:

```
public final void notify()
```

`notifyAll()`: Wakes up all threads that are waiting.

Syntax:

```
public final void notifyAll()
```

Inter-thread communication contd..

`sleep()`: This method causes the currently executing thread to sleep for the specified number of milliseconds.

// sleep for the specified number of milliseconds

```
public static void sleep(long millis) throws InterruptedException
```

//sleep for the specified number of milliseconds plus nano seconds

```
public static void sleep(long millis, int nanos)
                            throws InterruptedException
```

Inter-thread communication contd..

Difference between wait() and sleep():

wait()	sleep()
wait() method releases the lock	sleep() method doesn't release the lock.
is the method of Object class	is the method of Thread class
is the non-static method	is the static method
should be notified by notify() or notifyAll() methods	after the specified amount of time, sleep is completed.

Inter-thread communication contd..

`join()`: method of a Thread instance is used to join the start of a thread's execution to end of other thread's execution such that a thread does not start running until another thread ends.

- If `join()` is called on a Thread instance, the currently running thread will block until the Thread instance has finished executing.
- The `join()` method waits at most this much milliseconds for this thread to die. A timeout of 0 means to wait forever.

Syntax:

// waits for this thread to die.

```
public final void join() throws InterruptedException
```

// waits at most this much milliseconds for this thread to die

```
public final void join(long millis) throws InterruptedException
```

// waits at most milliseconds plus nanoseconds for this thread to die.

```
The java.lang.Thread.join(long millis, int nanos)
```

Inter-thread communication contd..

Example:

```
import java.lang.*;

public class JoinDemo implements Runnable {
    public void run() {
        Thread t = Thread.currentThread();
        System.out.println("Current thread: "+ t.getName());

        // checks if current thread is alive
        System.out.println("Is Alive? "+ t.isAlive());
    }

    public static void main(String args[]) throws Exception {
        Thread t = new Thread(new JoinDemo());
        t.start();

        // Waits for 1000ms this thread to die.
        t.join(1000);

        System.out.println("\nJoining after 1000‟ + " mili seconds: \n");
        System.out.println("Current thread: " + t.getName());

        // Checks if this thread is alive
        System.out.println("Is alive? " + t.isAlive());
    }
}
```

Inter-thread communication contd..

Output:

```
Current thread: Thread-0
Is Alive? true

Joining after 1000 mili seconds:

Current thread: Thread-0
Is alive? False
```

Note:

1. If any executing thread t1 calls join() on t2 i.e; t2.join() immediately t1 will enter into waiting state until t2 completes its execution.
2. Giving a timeout within join(), will make the join() effect to be nullified after the specific timeout.

Inter-thread communication contd..

Example:

```
class Customer{
int amount=10000;
synchronized void withdraw(int amount){
System.out.println("going to withdraw...");

if(this.amount<amount){
System.out.println("Less balance; waiting for dep
osit...");
try{wait();}catch(Exception e){}
}
this.amount-=amount;
System.out.println("withdraw completed...");
}
synchronized void deposit(int amount){
System.out.println("going to deposit...");
this.amount+=amount;
System.out.println("deposit completed... ");
notify();
}  }
class Test{
public static void main(String args[]){
final Customer c=new Customer();
new Thread(){
public void run(){c.withdraw(15000);}
}.start();
new Thread(){
public void run(){c.deposit(10000);}
}.start();  }}
```

Inter-thread communication contd..

```
public class WaitNotifyTest {
   private static final long SLEEP_INTERVAL =
3000;
   private boolean running = true;
   private Thread thread;
   public void start() {
      print("Inside start() method");
      thread = new Thread(new Runnable() {
         @Override
         public void run() {
            print("Inside run() method");
            try {
               Thread.sleep(SLEEP_INTERVAL);
            } catch(InterruptedException e) {

Thread.currentThread().interrupt();
            }
            synchronized(WaitNotifyTest.this)
{
               running = false;
               WaitNotifyTest.this.notify();
            }
         }
      });
      thread.start();
   }
public void join() throws InterruptedException {
      print("Inside join() method");
      synchronized(this) {
         while(running) {
            print("Waiting for the peer thread to
finish.");
            wait(); //waiting, not running
         }
         print("Peer thread finished.");
      }
   }
   private void print(String s) {
      System.out.println(s);
   }
   public static void main(String[] args) throws
InterruptedException {
      WaitNotifyTest test = new WaitNotifyTest();
      test.start();
      test.join();
   }
}
```

Inter-thread communication contd..

Example:

```
import java.lang.*;
 public class SleepDemo implements Runnable {
    Thread t;
    public void run() {
        for (int i = 0; i < 4; i++) {
            System.out.println(Thread.currentThread().getName()+ "  " + i);
            try {
                // thread to sleep for 1000 milliseconds
                Thread.sleep(1000);
            }
              catch (Exception e) {
                System.out.println(e);
            }
        }
    }
     public static void main(String[] args) throws Exception {
        Thread t = new Thread(new SleepDemo());
         // call run() function
        t.start();
         Thread t2 = new Thread(new SleepDemo());
         // call run() function
        t2.start();
    }
}
```

THANK YOU