Multithreading

By Rahul Barve

Objectives

- Understand Multitasking
- Introduction to Multithreading
- Need for Multithreading
- Implementing Multithreading
- Thread Life Cycle
- Thread Priorities
- Understanding Thread Methods
- Synchronization
- Inter-thread Communication

Multitasking

By Rahul Barve

Multitasking

- Multitasking is a process that involves multiple tasks running simultaneously.
- It is a generic terminology which has 2 forms: Multiprocessing and Multithreading.

Multiprocessing

By Rahul Barve

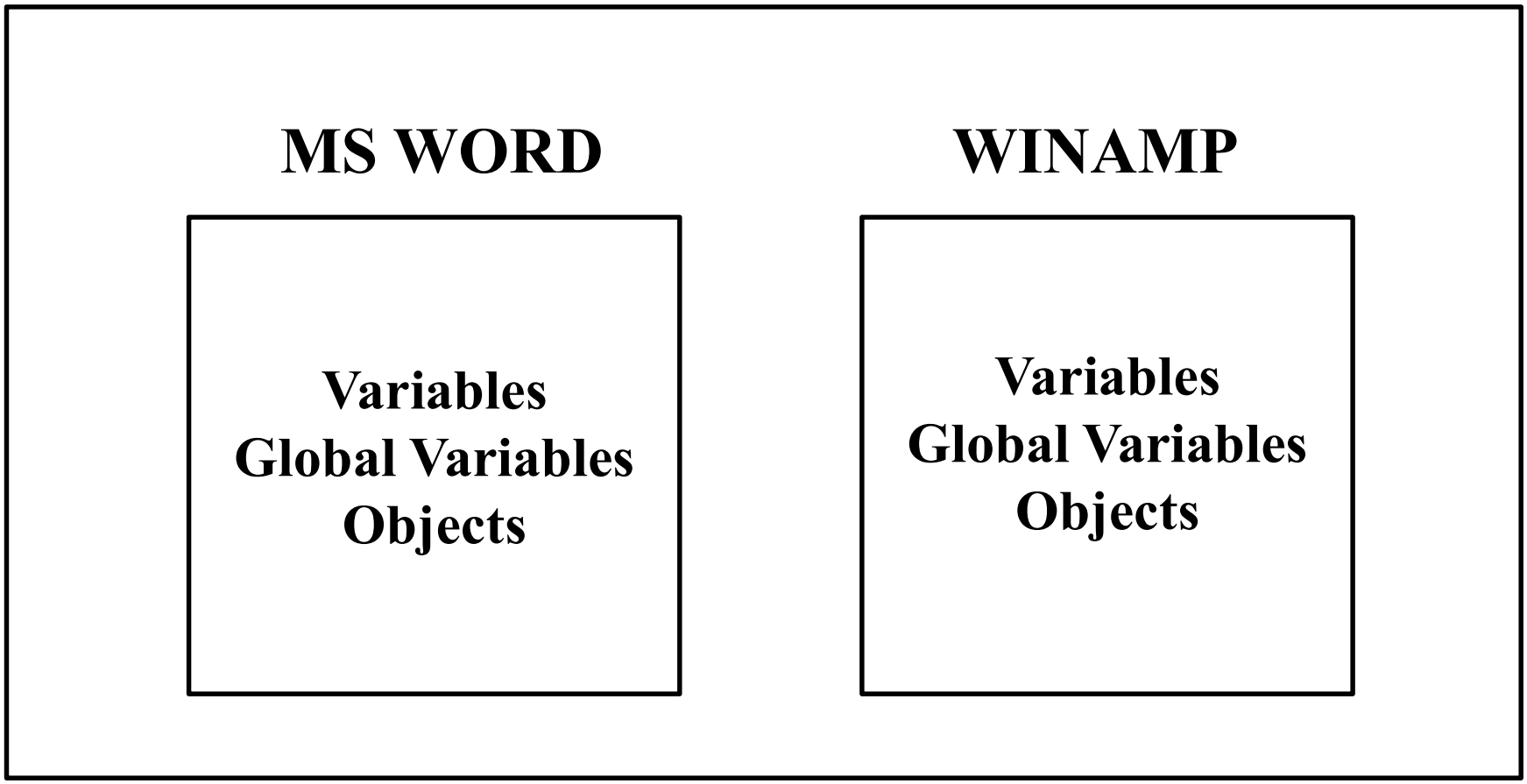
Multiprocessing

- Program is a set of instructions and a Process is a ‘Running Instance of a Program’.
- CPU is shared between 2 processes.

Multiprocessing

- In Multiprocessing, OS implements Context Switching mechanism.
- Program's entire context including variables, global variables, objects etc., is stored separately.

Multiprocessing



Multithreading

By Rahul Barve

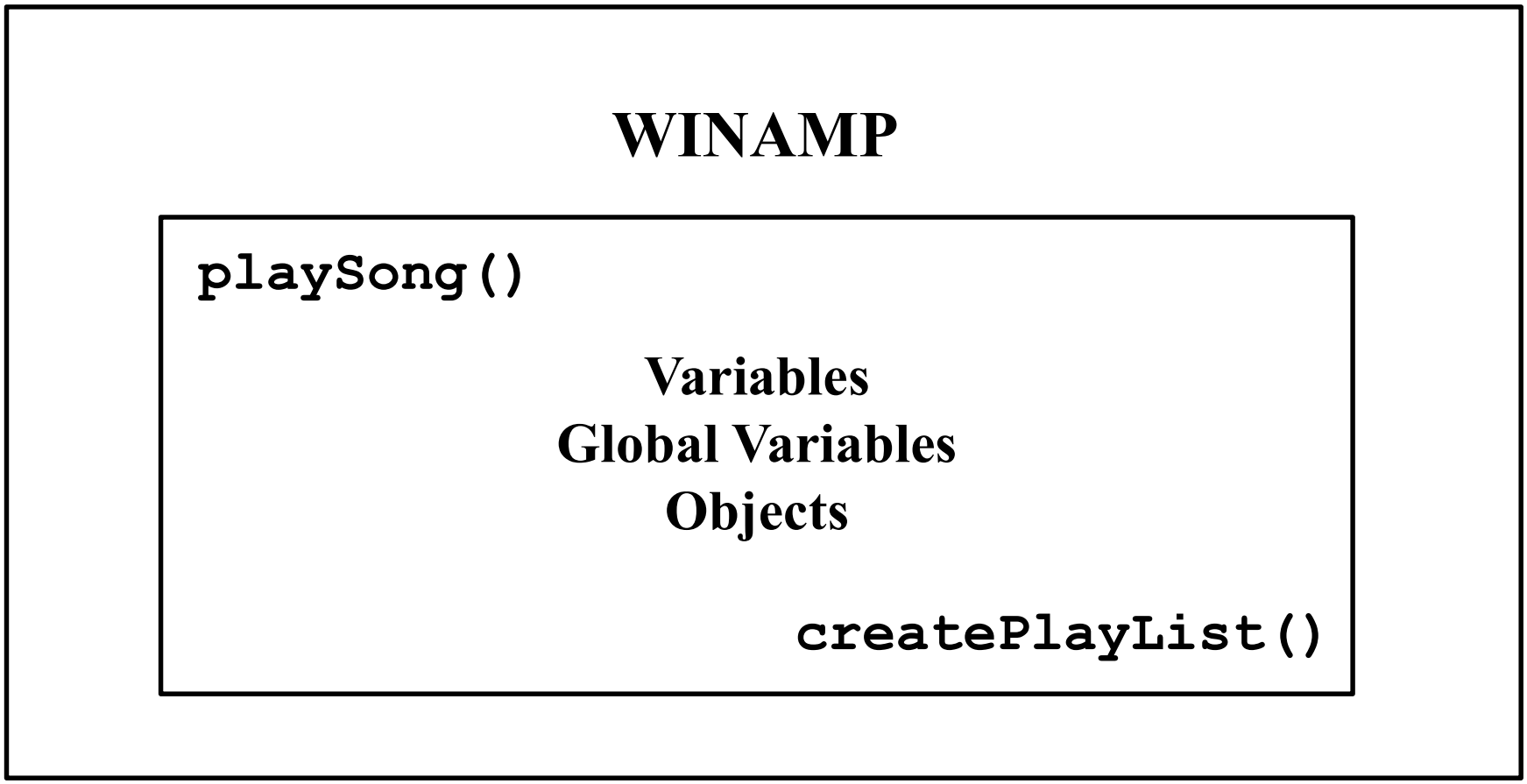
Multithreading

- Different tasks within a main task execute simultaneously.
- Multithreading implements the idea of Multitasking by taking it one level lower.

Multithreading

- Each sub task within an application is called as a Thread.
- Since multiple threads run within a same application, may share the data.

Multithreading



Multithreading

- A Thread is an entity within a Process.
- It defines the path of execution.

Process Vs. Thread

- Each process has a complete set of its own variables.
- It takes more overhead to launch a new process.
- Inter-process communication is a heavyweight activity.
- Threads live within a single process, thus may share the same data.
- It takes much less overhead to launch a new thread.
- Inter-thread communication is a lightweight activity.

Implementing Multithreading

By Rahul Barve

Implementing Multithreading

- In order to implement multithreading Java provides an API from `java.lang` package:
- A `Thread` class and a `Runnable` interface.

Implementing Multithreading

- To implement multithreading, it is necessary to create a class that is known as a Thread Implementation Class.
- It must either extend `Thread` or implement `Runnable`.

Implementing Multithreading

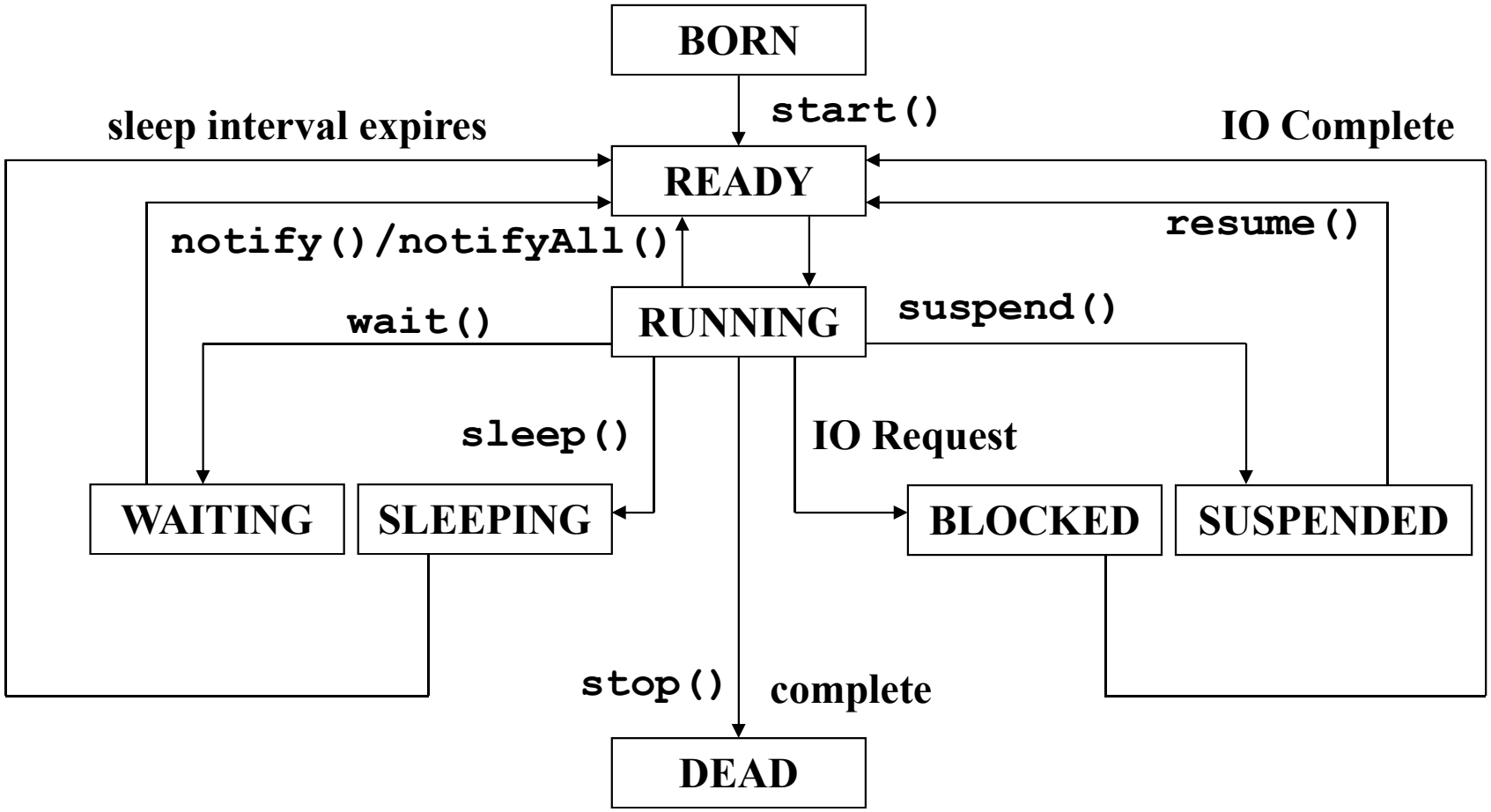
```
public class MyThread extends Thread {
     public void run(){….}
}

public class MyThread implements Runnable {
     public void run(){….}
}
```

Implementing Multithreading

- All thread implementation classes define a `run()` method which provides a logic for the thread or acts as a gateway to the logic.

Life Cycle of Thread



Thread Priorities

By Rahul Barve

Thread Priorities

- Depending upon the task the thread is going to perform, it is possible to assign a priority to the thread.
- Priorities can be specified using `setPriority()` method within the range 1 to 10, being 1 as lowest and 10 as highest.

Thread Priorities

- The scheduler picks up the highest priority thread that is currently in the `READY` state.
- The default priority of a thread is 5.

Thread Methods

By Rahul Barve

Thread Methods

- `start()`
- `stop()`
- `yield()`
- `isAlive()`
- `sleep()`
- `suspend()`
- `resume()`
- `currentThread()`
- `join()`

`start()`

- A method that makes a request to OS for the creation of the Thread.
- Responsible for transitioning of the thread from `BORN` state to `READY` state.

`stop()`

- Forcefully kills the thread.
- Sends the thread into `DEAD` state.
- Declared as `deprecated`.

`yield()`

- A `static` method that causes currently executing thread to yield the control.
- If there are other runnable threads of which, priority is at least as high as this thread, they will be scheduled next.

`isAlive()`

- Returns `true` if a thread has started and not yet terminated.

`sleep()`

- A `static` method that sends a thread into the `SLEEPING` state.
- A thread remains into the `SLEEPING` state until the sleep time interval is over.

`suspend()` and `resume()`

- `suspend()`:
  - Sends a thread into the `SUSPENDED` state.
- `resume()`:
  - Brings the thread into the `READY` state.
- Both are declared as `deprecated`.

`currentThread()`

- A `static` method that returns a reference to the thread that is currently running.

`join()`

- Causes the parent thread to wait until the termination of the child thread on which it is invoked.

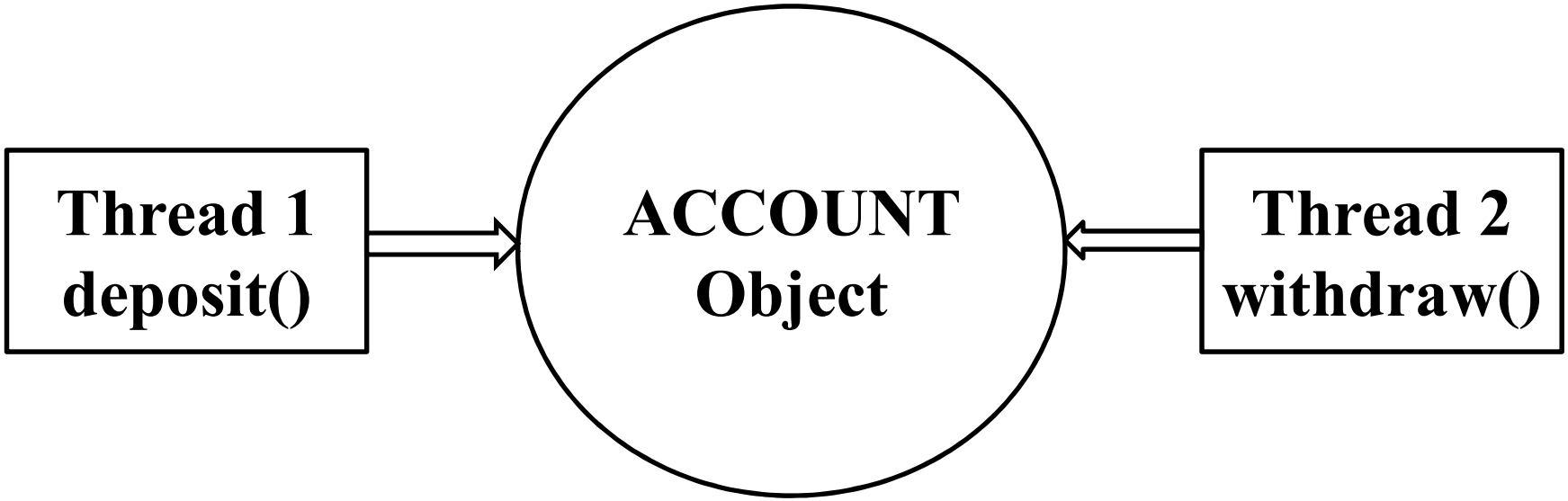
Synchronization

By Rahul Barve

Synchronization

- Many times in an application, two or multiple threads need to access the same object.
- This needs to ensure that the object will be modified by only one thread at a time, otherwise they will fall into race condition.

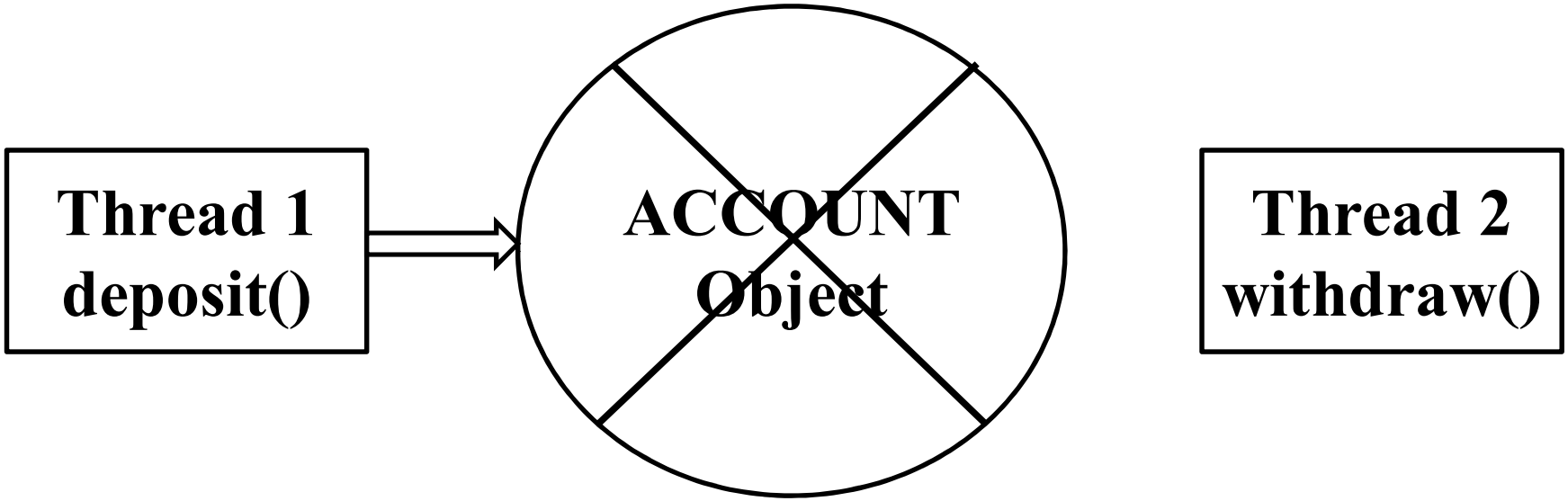
Synchronization



Synchronization

- Key to synchronization is the concept of monitor.
- A monitor is an object that is used as a mutually exclusive lock.
- Only one thread can own a monitor at a given time.

Synchronization



Synchronization

- When a thread acquires a lock, it is said to have entered the monitor.
- All other threads attempting to enter the locked monitor will get suspended, until the first thread exits the monitor.

Synchronization

- Thread synchronization is achieved in 2 ways:
  - Using synchronized methods.
  - Using synchronized blocks.

Synchronization

- Using synchronized methods.

```
public synchronized void m1(){
   //Some Code
}
```

Synchronization

- Using synchronized blocks.

```
synchronized(obj){
    //Some Code
}
```

Inter-Thread Communication

By Rahul Barve

Inter-Thread Communication

- Sometimes, in an application there is a need of two or multiple threads interacting with each other. It leads to inter-thread communication.

Inter-Thread Communication

- To implement inter-thread communication, there are 3 methods used:
  - `wait()`
  - `notify()`
  - `notifyAll()`

`wait()`

- Must be invoked within a synchronized context.
- When invoked, releases the lock and sends the currently running thread into the `WAITING` state.
- Gives a chance to other threads looking for the same lock.

notify() & notifyAll()

- `notify()`
  - Wakes up a single thread that is in `WAITING` state.
- `notifyAll()`
  - Wakes up all threads.

Lets Summarize

- Multitasking
- Multithreading
- Need for Multithreading
- Implementing Multithreading
- Thread Life Cycle
- Thread Priorities
- Thread Methods
- Synchronization
- Inter-thread Communication