

# **JAVA @17**

## Java Thread – Part I

# Objective

After completing this session you will be able to understand,

- What is Multithreading?
- Life cycle of a Thread
- Creating Thread
- Thread Scheduler
- Thread Interruption techniques
- Thread Priority

# Process vs Thread

Before we learnt about threads let's first understand the difference between a thread and a process.

**Process** are executables which run in separate memory space.

**Threads** are small process which run in shared memory space within a process.

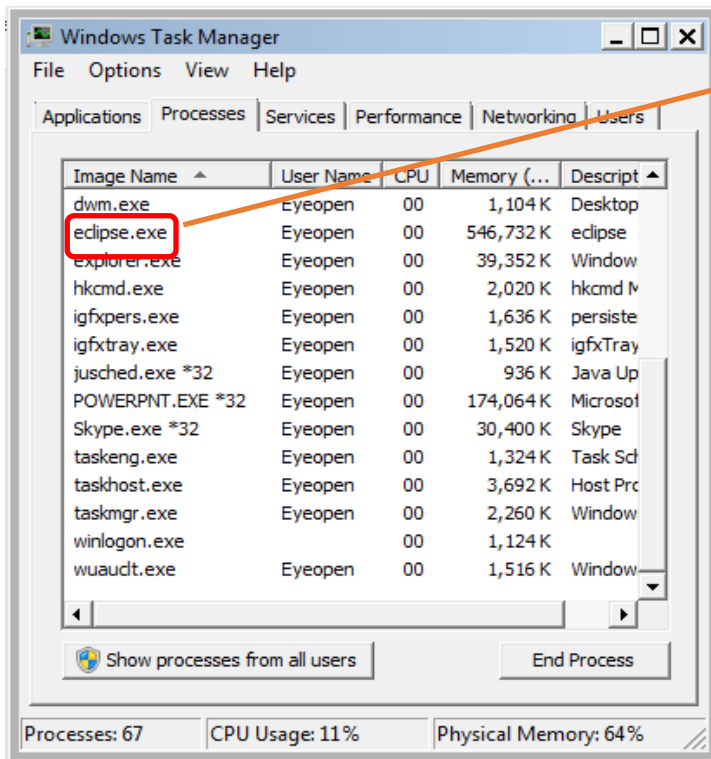
***So in a nutshell Process is a container for Threads, all thread runs inside the process.***



**Still confused let's look at an example to understand it better.**

# Process

Lets consider Eclipse IDE application to understand it better, what happens when you start an eclipse application.



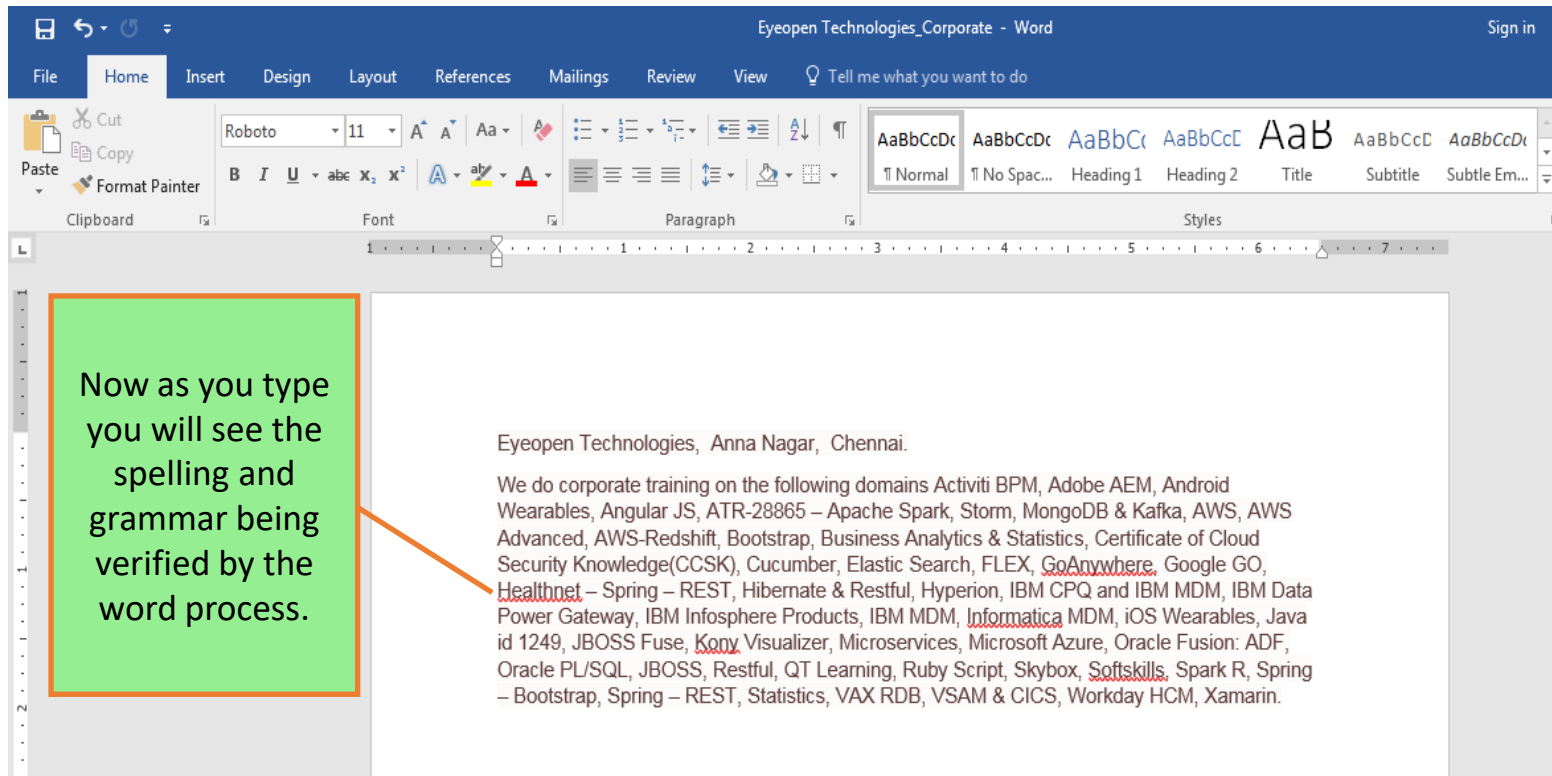
A process for eclipse ide application will be started.

Now lets see what happens when the user starts using word application.

# Thread

The spell check has been implemented as a *thread* within the word.exe process which runs continuously and verifies what you type.

Word.exe is the process and spell check is a thread running inside the process.



# Real time scenario

**Ram** was developing an application where he has a requirement where user can register his profile in the application, assume registration has three steps

**Validate user details** – Takes 3 seconds for execution for each user.

**Validate user Citizenship** - Takes 4 seconds for execution for each user.

Now customer wants to complete the registration process is less than 5 seconds.



Guess how **Mr. Ram** would have achieved it?

He used **Multi Threading**.

Lets see what it is and how it can be implemented in this session.

# Multitasking

## What is Multitasking?

- Refers to a computer's ability to perform multiple jobs concurrently
- More than one program are running concurrently,

Example: In windows you can run Word, power point , media player at the same time. Yu will working on word and in parallel media player might play some music.

# Multithreading

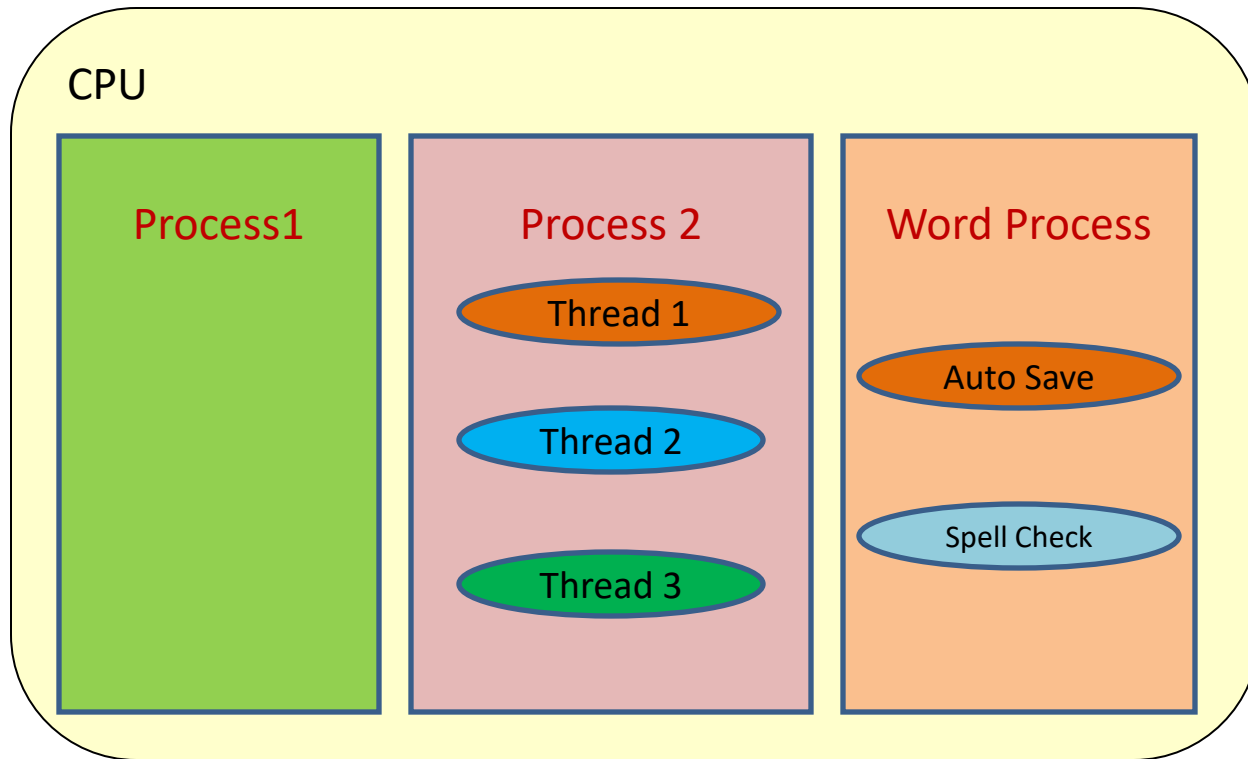
## What is Multithreading?

- A thread is a single sequence of execution within a program/process.
- This refers to multiple threads of control within a single program.
- Each program can run multiple threads of control within it.

Example: Microsoft word process having multiple threads like spell check, auto save etc.



# System Illustration



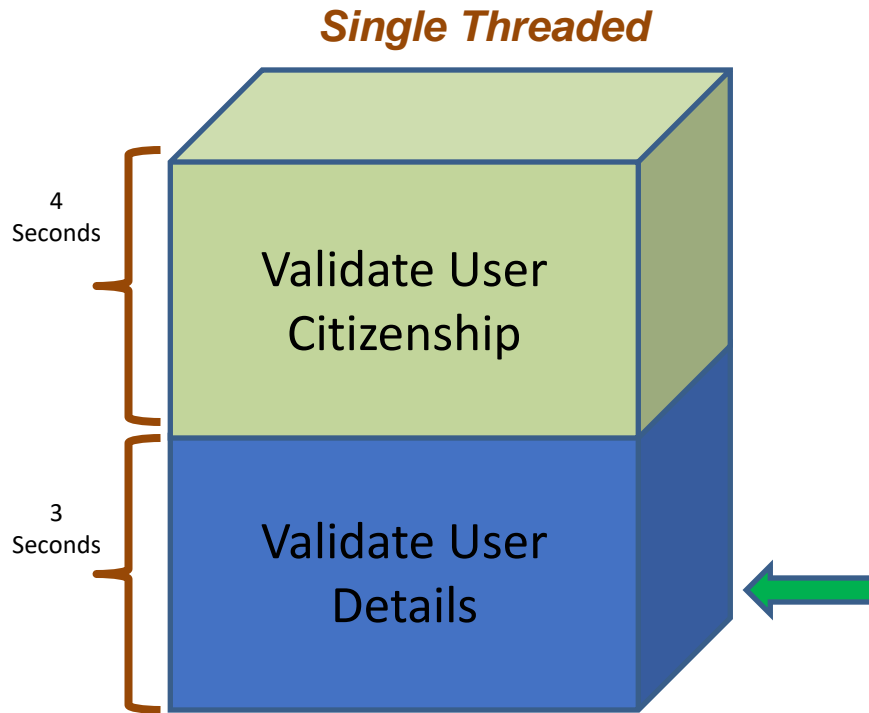
The CPU is running three processes. Process 2 in turn has three threads running inside it.

# Benefits of Multithreading

- To reduce response (execution) time of a process.
- Support Parallel Operation of Functions.
- Increase System Efficiency.
- Requires less overheads compared to Multitasking.

# How Ram Solved the Problem?

Ram implemented two threads for processing the **Validate user details** and **Validate user Citizenship**.



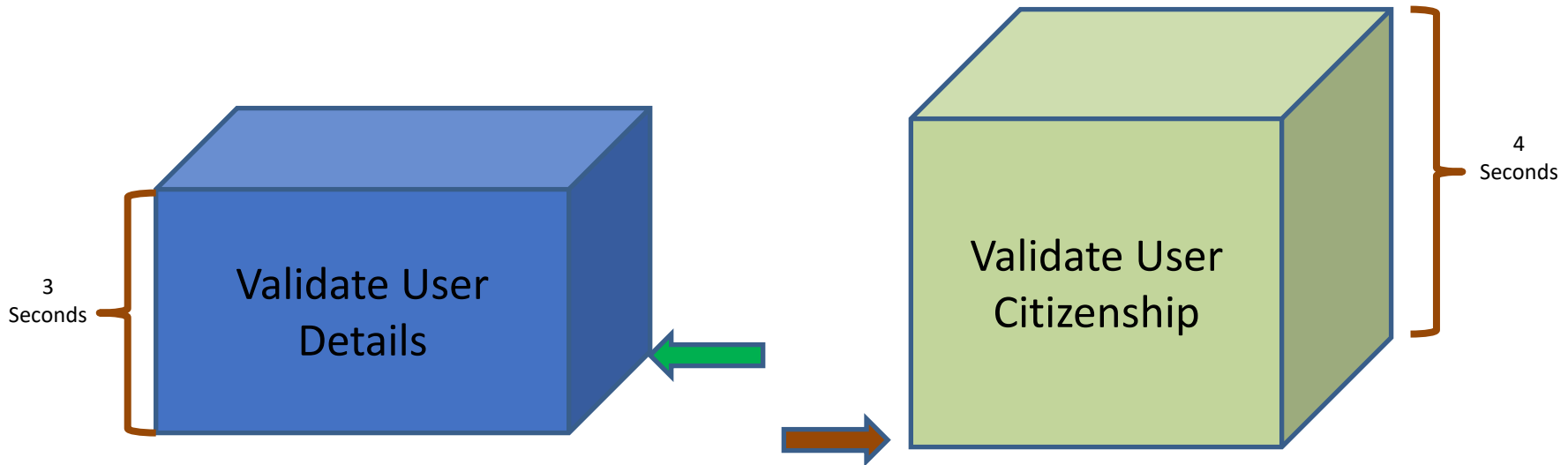
The thread takes 7 seconds for completing the registration process.

Let look how Ram implemented it.

# How Ram Solved the Problem?

Ram implemented two threads one thread for each method **Validate user details** and **Validate user Citizenship**.

*Multi Threaded*



The process will be completed in 4 seconds.  
Since both the threads works in parallel.

# What is an Application Thread?

*When we execute an application,*

- 1. The JVM creates a thread (T1) object which invokes the main() method and starts the application.*
- 2. The thread executes the statements of the program one by one (or) starts other threads.*
- 3. After executing all the statements, the method returns and the thread dies.*

The thread **T1** which is responsible for starting the application by invoking the main method is called “**Application Thread**”.

# Ways of Implementing Threads

**Method 1** : Extend *Thread* Class

**Method 2** : Implement *Runnable* Interface

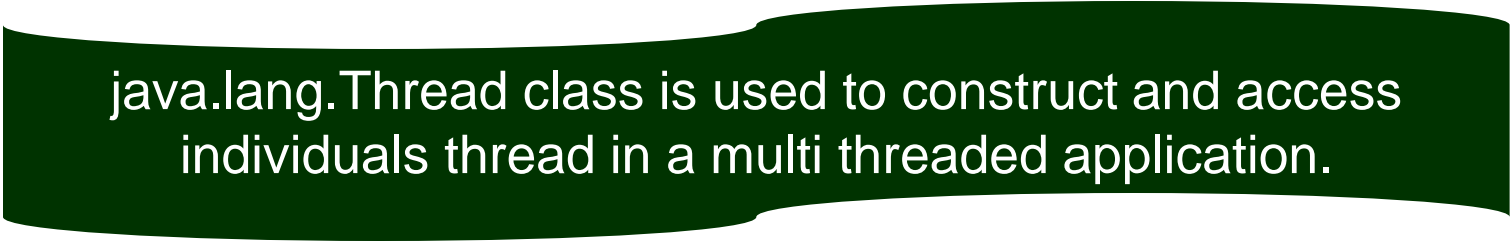
# Using Thread in Java



How To Use  
Threads in  
Java



Using Thread  
class



java.lang.Thread class is used to construct and access individuals thread in a multi threaded application.

# Using Thread in Java

```
public class Thread extends Object implements Runnable {  
    public Thread();  
    public Thread(String name); // Thread name  
    public Thread(Runnable r); // Thread  $\Rightarrow$  r.run()  
    public Thread(Runnable r, String name);  
    public void run();  
    public void start();// begin thread execution  
}
```

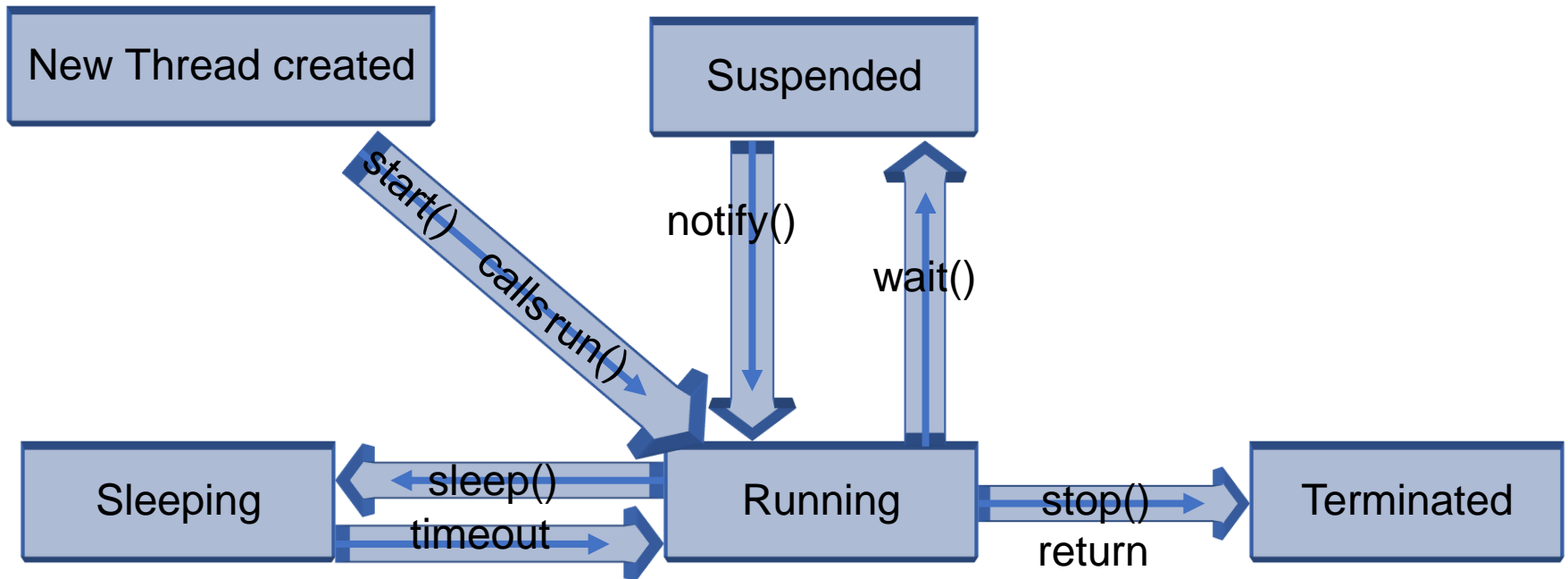


# Thread class methods

Method	Description
<b>void run()</b>	<ul style="list-style-type: none"><li>• The thread logic should be implemented in this method.</li><li>• This method should be overridden in all the Thread class.</li></ul>
<b>void start()</b>	Creates a new thread and invokes run method of the thread.
<b>getName()</b>	Returns the thread's name.
<b>int getPriority()</b>	Returns the thread's priority
<b>boolean isAlive()</b>	Tests if the thread is alive.

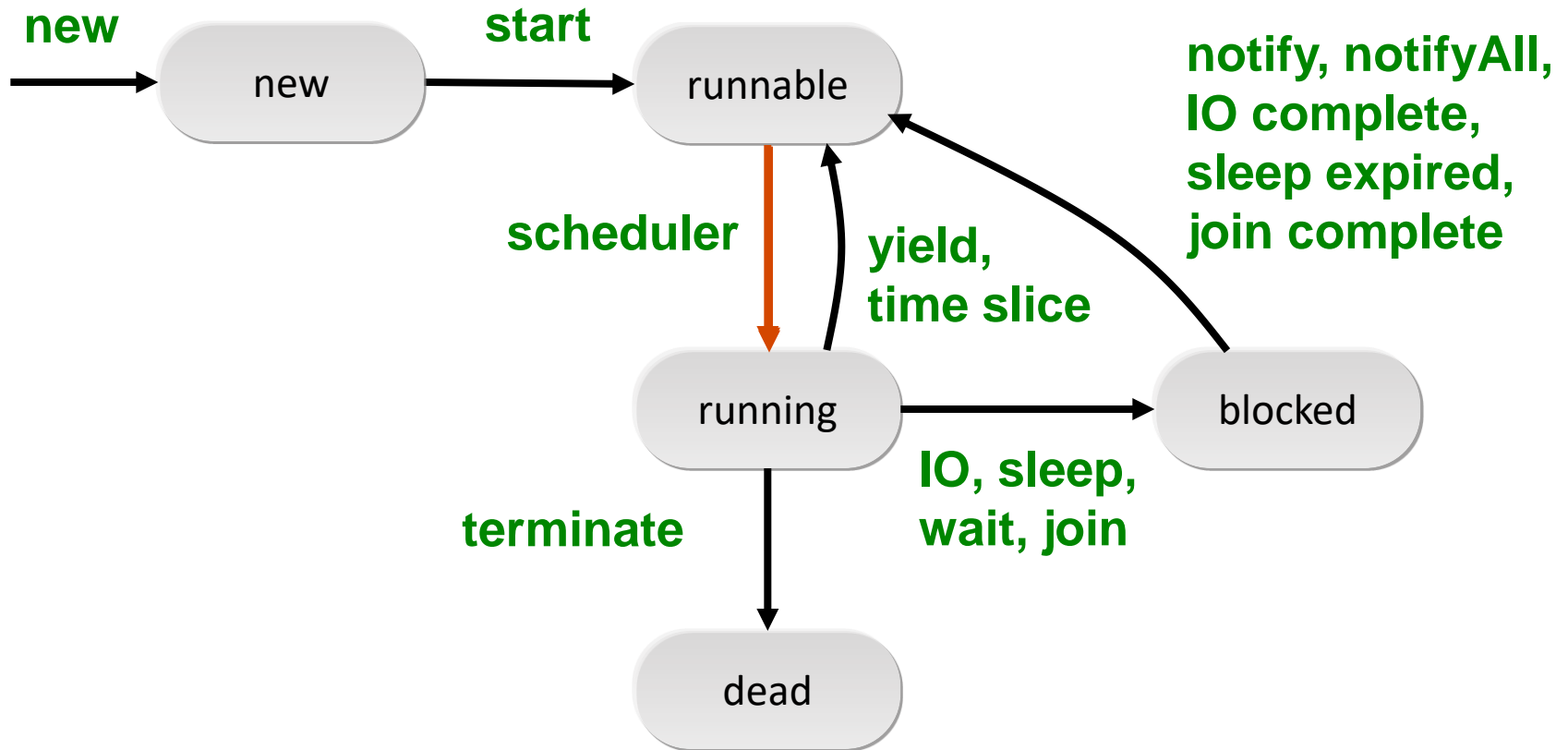
Method	Description
<b>Thread currentThread()</b>	Return a reference to the currently executing thread object
<b>setName(String name)</b>	Sets a name for the thread to be equal to the argument name.
<b>setPriority (int newPriority)</b>	Changes the priority of the thread
<b>static void sleep(long millis)</b>	Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
<b>void yield()</b>	Causes the currently executing thread object to temporarily pause and allow other threads of same priority to execute

# Lifecycle of Thread



# Java thread states

## State Diagram



# Example – How to develop a thread?

In this demo we will learn how to

1. Create a Thread by extending the Thread class
2. Override the run method
3. Create a thread object and start the thread using the start method
4. How the following methods operates
  - a. getName()
  - b. setName()
  - c. setPriority()
  - d. getPriority()
  - e. sleep()

## **Components to be developed,**

1. ThreadEX – The Thread class should loop and print values 0...4.
2. ThreadExMain – The main class to execute the Threads

# Solution - Thread

```
public class ThreadEx extends Thread {  
    int i = 0;
```

Class ThreadEx extends Thread class and becomes a Thread class

```
    public ThreadEx(String name) {  
        this.setName(name);
```

Sets the name of the Thread using setName method()

```
    }
```

Overrides  
the run()  
method

Prints the name of the current  
Thread using the getName()  
method

```
    public void run() {  
        for (i = 0; i < 5; i++) {  
            System.out.println("Printing from " + Thread.currentThread().getName()  
                               + " the value of i :: " + i);  
            try {  
                Thread.sleep(300);  
            } catch (InterruptedException ex) {  
                ex.printStackTrace();  
            }  
        }  
    }  
}
```

Makes the Thread sleep for 300 milliseconds. Invoking the sleep may cause an InterruptedException to be thrown which should be handled.

# Solution - Thread

```
public class ThreadExMain {  
    public static void main(String args[]) throws InterruptedException {  
        Thread t = Thread.currentThread();  
        System.out.println("The Current Thread is " + t.getName());  
        System.out.println("The Current Thread Priority is " + t.getPriority());  
        t.setName("myThread");  
        t.setPriority(6);  
        System.out.println("The Current Thread is " + t.getName());  
        System.out.println("The Current Thread Priority is " + t.getPriority());  
        ThreadEx ex1 = new ThreadEx("first");  
        ThreadEx ex2 = new ThreadEx("second");  
        System.out.println("The Thread Priority of ex1 is "  
            + ex1.getPriority());  
        System.out.println("The Thread Priority of ex2 is "  
            + ex2.getPriority());  
        ex1.start();  
        ex2.start();  
        for (int i = 0; i < 5; i++) {  
            System.out.println("Printing from : "  
                + Thread.currentThread().getName() + " :: " + i);  
            Thread.sleep(100);  
        }  
    }  
}
```

Gets the priority and name of the main thread

Sets the priority and name of the main thread

Creates two new Thread objects with name "first" and "second"

Starts the Thread by invoking the start () which will invoke the run() method of the Thread class

# Solution - Thread

- Execute the main class – ***ThreadExMain***
- The output will be something as shown below – Output may vary for each execution since Thread execution is based on the underlying operating system.

```
The Current Thread is main
The Current Thread Priority is 5
The Current Thread is myThread
The Current Thread Priority is 6
The Thread Priority of ex1 is 6
The Thread Priority of ex2 is 6
Printing from : myThread :: 0
Printing from first the value of i :: 0
Printing from second the value of i :: 0
Printing from : myThread :: 1
Printing from : myThread :: 2
Printing from first the value of i :: 1
Printing from second the value of i :: 1
Printing from : myThread :: 3
Printing from : myThread :: 4
Printing from second the value of i :: 2
Printing from first the value of i :: 2
Printing from second the value of i :: 3
Printing from first the value of i :: 3
Printing from second the value of i :: 4
Printing from first the value of i :: 4
```

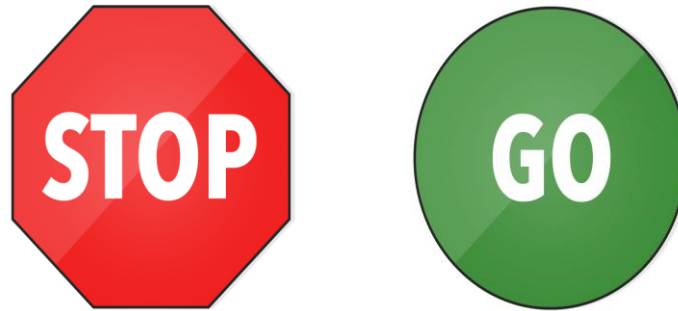


Different output  
displayed during  
different run

```
The Current Thread is main
The Current Thread Priority is 5
The Current Thread is myThread
The Current Thread Priority is 6
The Thread Priority of ex1 is 6
The Thread Priority of ex2 is 6
Printing from : myThread :: 0
Printing from first the value of i :: 0
Printing from second the value of i :: 0
Printing from : myThread :: 1
Printing from : myThread :: 2
Printing from second the value of i :: 1
Printing from first the value of i :: 1
Printing from : myThread :: 3
Printing from : myThread :: 4
Printing from second the value of i :: 2
Printing from first the value of i :: 2
Printing from first the value of i :: 3
Printing from second the value of i :: 3
Printing from first the value of i :: 4
Printing from second the value of i :: 4
```



# Time To Reflect



**Trainees to reflect the following topics before proceeding.**

- What is Multithreading?
- How can we implement threading in Java API?
- Explain about Thread Lifecycle methods in Java API?

Thank you

*You have successfully completed*  
**Thread-1**