CAP615
PROGRAMMING IN JAVA

Unit-3

Created By:
Kumar Vishal
(SCA), LPU

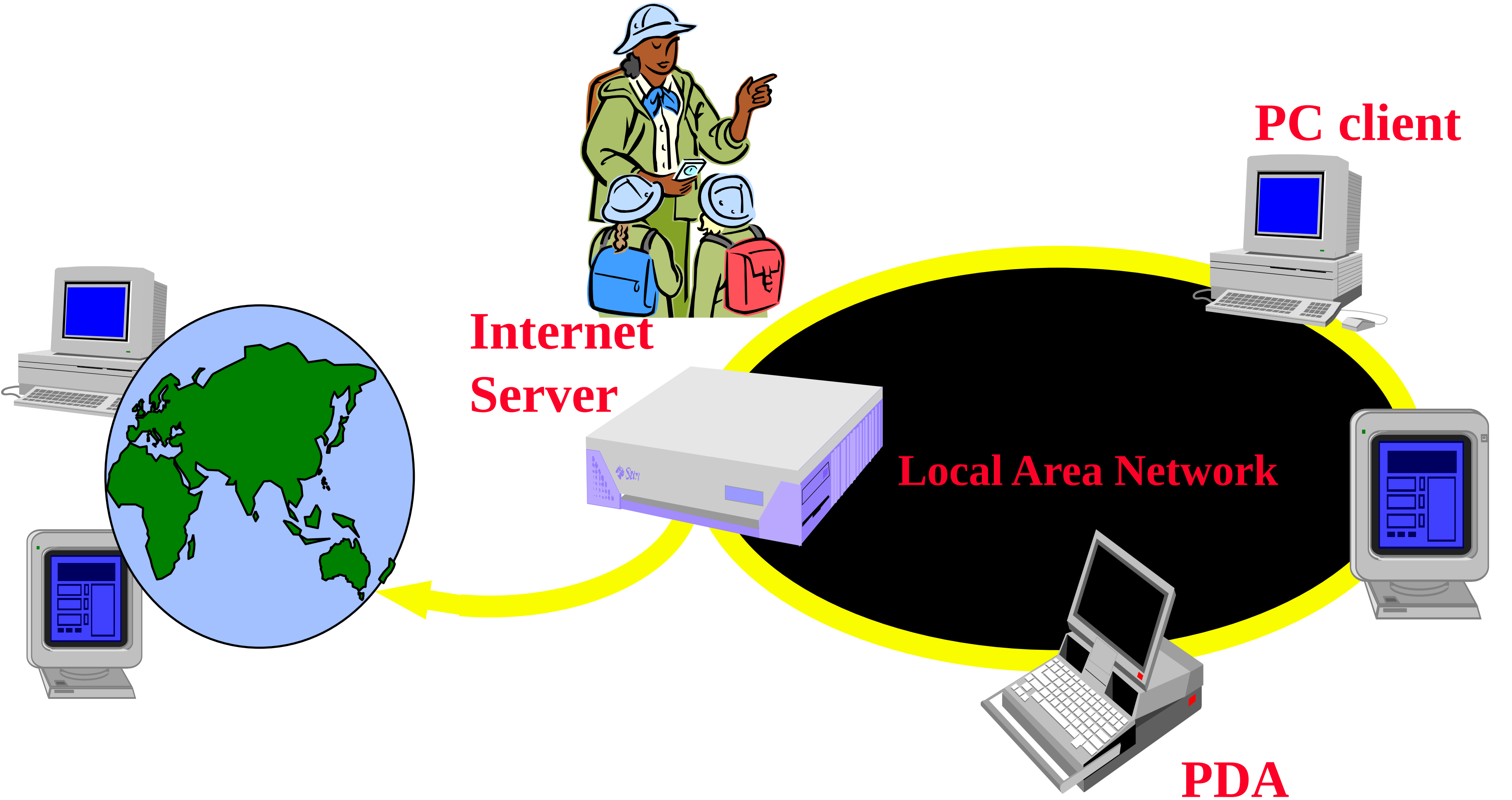
Topics Covered….

- ✓implementing multithreading,
- ✓life cycle of a thread,
- ✓thread communication,
- ✓suspending, resuming,
- ✓deadlock and stopping threads,
- ✓thread synchronization,
- ✓handling exceptions during multithreading

Multithreading in java

- ✓ Multithreading in java is a process of executing multiple threads simultaneously.
- ✓ Multithreading is used to achieve multitasking.
- ✓ Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.
- ✓ It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- ✓ You can perform many operations together, so it saves time.

Web/Internet Applications: Serving Many Users Simultaneously



Creating threads in Java

- Create a class that extends the Thread class
- Create a class that implements the Runnable interface

Create a class by extending Thread class and override run() method:

```
class MyThread extends Thread
{
    public void run()
   {
      // thread body of execution
   }
}
```

Create a thread:

```
MyThread thr1 = new MyThread();
```

Start Execution of threads:

```
thr1.start();
```

Create and Execute:

```
new MyThread().start();
```

Create a class that implements the interface Runnable and override run() method:

```
class MyThread implements Runnable
{
 .....
 public void run()
 {
   // thread body of execution
 }
}
```

Creating Object:

```
MyThread myObject = new MyThread();
```

Creating Thread Object:

```
Thread thr1 = new Thread( myObject );
```

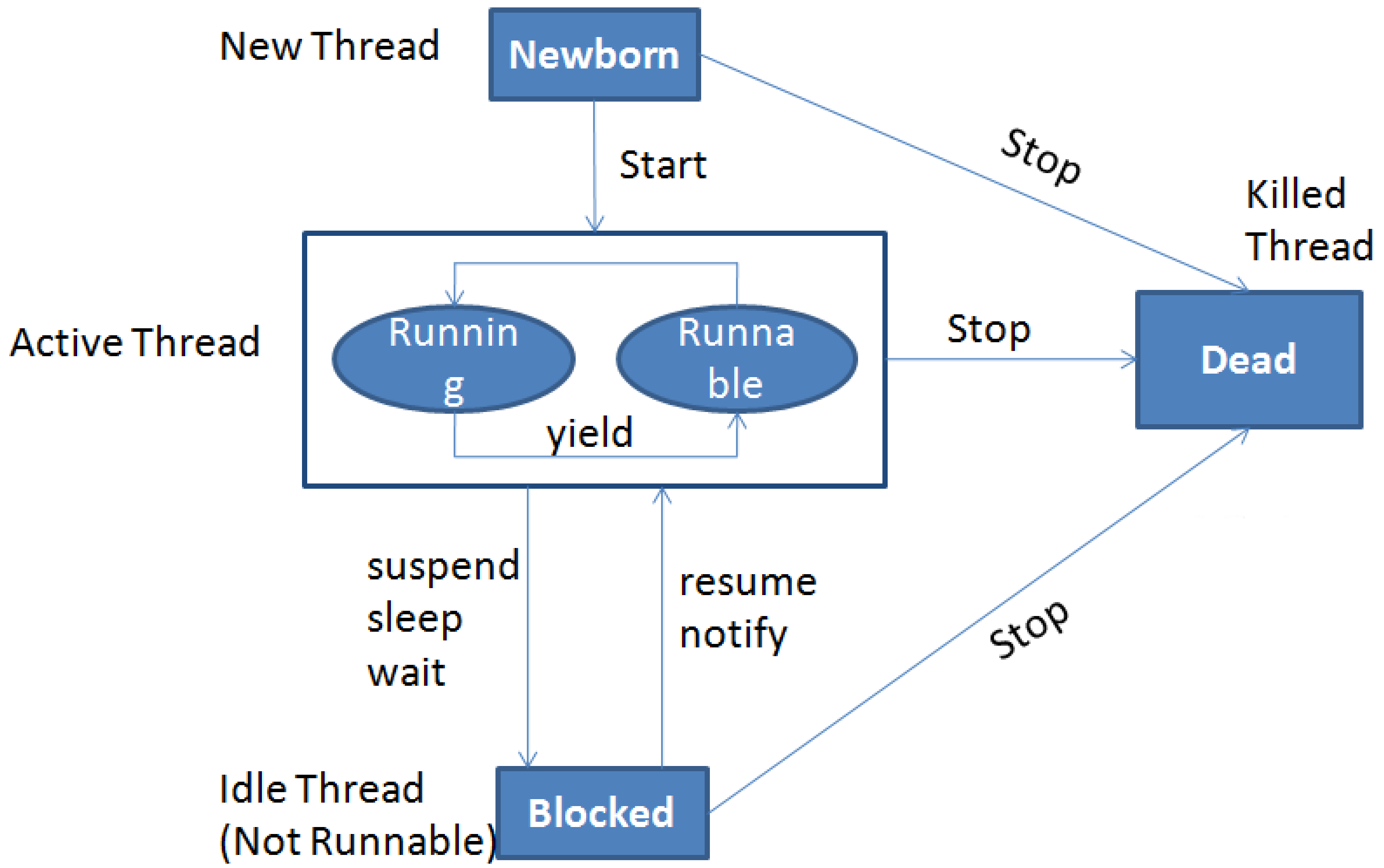
Start Execution:

```
thr1.start();
```

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

Basically there are four state:

- New Born
- Runnable/Running
- Non-Runnable/Blocked
- Terminated/Dead


New Thread
Newborn
Start
Stop
Killed Thread
Active Thread
Runnin g
Runna ble
yield
Stop
Dead
suspend
sleep
wait
resume
notify
Stop
Idle Thread
(Not Runnable)
Blocked

1 Newborn State

- When we create a thread it will be in Newborn State.
- The thread is just created still its not running.
- We can move it to running mode by invoking the start() method and it can be killed by using stop() method.

2 Runnable State

- It means that thread is now ready for running and its waiting to give control.
- We can move control to another thread by yield() method.
- A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready run at any instant of time. It is the responsibility of the **thread scheduler** to give the thread, time to run.

3 Running State

- It means thread is in its execution mode becaause the control of cpu is given to that particular thread.
- It can be move in three different situation from running mode.

4 Blocked State

- A thread is called in Blocked State when it is not allowed to entering in Runnable State or Running State.
- It happens when thread is in waiting mode, suspended or in sleeping mode.

5 Dead State

- When a thread is completed executing its run() method the life cycle of that particular thread is end.
- We can kill thread by invoking stop() method for that particular thread and send it to be in Dead State.

thread synchronization

- In multithread program where multiple threads try to access the same resources then problem occurs..
- To overcome this concept of thread synchronization came.
- problem without Synchronization
  - Example

Interthread communication

- Interthread communication is important when you develop an application where two or more threads exchange some information.
- There are three methods which makes thread communication possible:

wait(), notify() and notifyAll()

--All these methods belong to object class as final

--They must be used within a synchronized block only.

- **wait()-**It tells the calling thread to release the lock and go to sleep until some other thread enters the same monitor and calls notify().
- **notify()-**It wakes up one single thread that called wait() on the same object.
- **notifyAll()-**It wakes up all the threads that called wait() on the same object.

Example

Thread Priority

- In Java, each thread is assigned priority, which affects the order in which it is scheduled for running. The threads so far had same default priority (NORM_PRIORITY) and they are served using FCFS policy.
  - Java allows users to change priority:
    - Threadobject.setPriority(intNumber)
      - MIN_PRIORITY = 1
      - NORM_PRIORITY=5
      - MAX_PRIORITY=10

Thread Priority Example

```
class A extends Thread
{
      public void run()
       {
          System.out.println("Thread A started");
          for(int i=1;i<=4;i++)
           {
                System.out.println("\t From ThreadA: i= "+i);
           }
            System.out.println("Exit from A");
       }
}
class B extends Thread
{
      public void run()
       {
          System.out.println("Thread B started");
          for(int j=1;j<=4;j++)
           {
                System.out.println("\t From ThreadB: j= "+j);
           }
            System.out.println("Exit from B");
       }
}
```

Thread Priority Example

```
class C extends Thread
{
      public void run()
        {
            System.out.println("Thread C started");
            for(int k=1;k<=4;k++)
              {
                   System.out.println("\t From ThreadC: k= "+k);
              }
               System.out.println("Exit from C");
        }
}
class ThreadPriority
{
         public static void main(String args[])
          {
                    A threadA=new A();
                    B threadB=new B();
                    C threadC=new C();
                  threadC.setPriority(Thread.MAX_PRIORITY);
                  threadB.setPriority(threadA.getPriority()+1);
                  threadA.setPriority(Thread.MIN_PRIORITY);
                  System.out.println("Started Thread A");
                   threadA.start();
                  System.out.println("Started Thread B");
                   threadB.start();
                  System.out.println("Started Thread C");
                   threadC.start();
                   System.out.println("End of main thread");
          }
}
```

Exception Handling where exceptions may occur

- int a=50/0;//ArithmeticException
- String s=null; System.out.println(s.length());//NullPointerException

String s="abc";

int i=Integer.parseInt(s);//NumberFormatException

- int a[]=new int[5]; a[10]=50; //ArrayIndexOutOfBoundsException

Five keywords used in Exception handling:

- try
- catch
- finally
- throw
- throws

finally block

- The finally block is a block that is always executed. It is mainly used to perform some important tasks such as closing connection, stream etc.
- **Rule:** For each try block there can be zero or more catch blocks, but only one finally block.

Throw/throws keyword

- If a method does not handle a checked exception, the method must declare it using the **throws** keyword. The throws keyword appears at the end of a method's signature.
- The throw keyword is used to explictily throw an exception. We can throw either checked or uncheked exception. The throw keyword is mainly used to throw custom exception.

_____controlled the life cycle of a thread?

A. main()

B. JDK

C. JRE

D. JVM

Constructors of Thread class?

A. Thread()

B. Thread(String name)

C. Thread(Runnable r)

D. Above All

Select which is not method of Thread class?

A. public void run()

B. public void start()

C. public void join()

D. public void getThread()

Thread start() method internally calls _____ method?

A. get()
B. execute()
C. run()
D. None