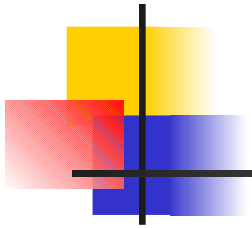
Multithreading

Introduction

Java provides built-in support for *multithreaded programming*.

A multithreaded program contains two or more parts that can run concurrently.

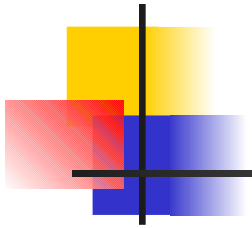
Each part of such a program is called a thread, and each thread defines a separate path of execution.



A multithreading is a specialized form of multitasking

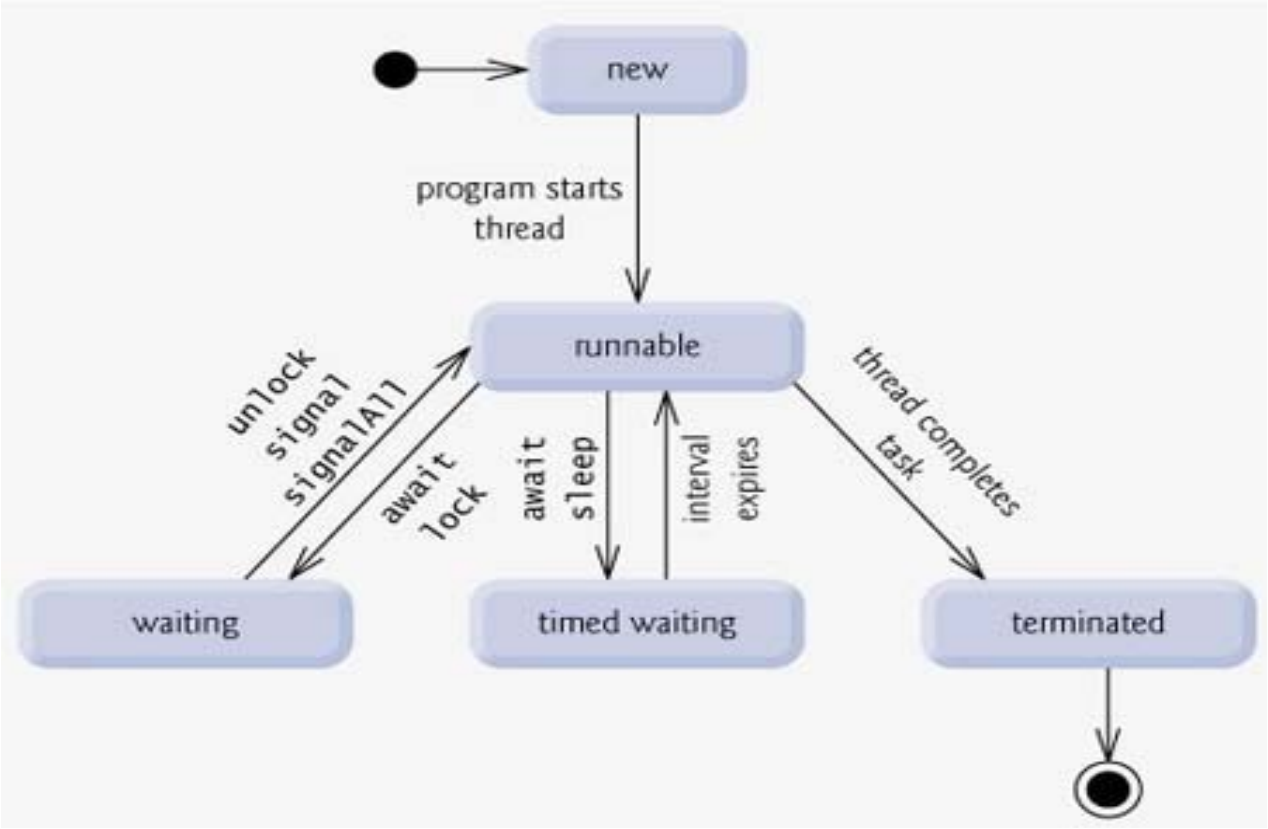
process: A process consists of the memory space allocated by the operating system that can contain one or more threads.

A thread cannot exist on its own; it must be a part of a process.



Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum.

Life Cycle of a Thread



- **New:** A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.
- **Runnable:** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

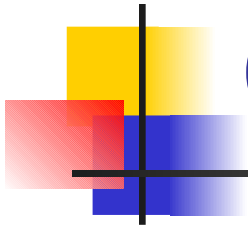
- **Waiting:** Sometimes a thread transitions to the waiting state while the thread waits for another thread to perform a task.A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
- **Timed waiting:** A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

- **Terminated:** A runnable thread enters the terminated state when it completes its task or otherwise terminates.

Thread Priorities

Java priorities are in the range between

- MIN_PRIORITY (a constant of 1)
- MAX_PRIORITY (a constant of 10).
- NORM_PRIORITY (a constant of 5). (default)



Creating a Thread

Two methods:

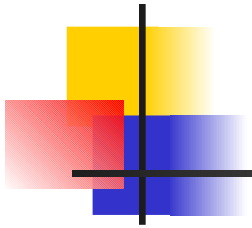
- implement the Runnable interface.
- extend the Thread class, itself

Create Thread by Implementing Runnable

- To implement Runnable, a class need only implement a single method called **run( )**, which is declared like this:

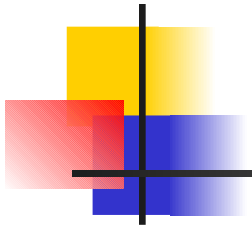
```
public void run( )
```

You will define the code that constitutes the new thread inside run() method



After you create a class that implements Runnable, you will instantiate an object of type Thread from within that class

Thread(Runnable threadOb, String threadName);



After the new thread is created, it will not start running until you call its **start( )** method, which is declared within Thread.

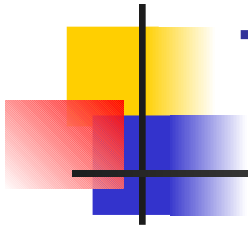
```
void start( );
```

```
// Create a new thread.
class NewThread implements Runnable
{
Thread t;
NewThread() { // Create a new, second thread t
   = new Thread(this, "Demo Thread");
   System.out.println("Child thread: " + t);
   t.start(); // Start the thread
 }
```

```
// This is the entry point for the second thread.
 public void run()
   {
   try
   {
     for(int i = 5; i > 0; i--)
        {
         System.out.println("Child Thread: " + i);
   // Let the thread sleep for a while.
      Thread.sleep(500);
        } }
    catch (InterruptedException e)
    {
       System.out.println("Child interrupted.");
 }
System.out.println("Exiting child thread.");
} }
```

```
class ThreadDemo {
 public static void main(String args[]) {
  new NewThread(); // create a new thread
  try {
        for(int i = 5; i > 0; i--)
         {
              System.out.println("Main Thread: " + i);
               Thread.sleep(1000);
         } }
       catch (InterruptedException e)
        {
          System.out.println("Main thread interrupted.");
        } System.out.println("Main thread exiting.");
 } }
```

```
Child thread: Thread[Demothread,5,main] Main Thread: 5
   Child Thread: 5
    Child Thread: 4
   Main Thread: 4
   Child Thread: 3
   Child Thread: 2
  Main Thread: 3
   Child Thread: 1
 Exiting child thread.
Main Thread: 2
 Main Thread: 1
 Main thread exiting.
```



Create Thread by Extending Thread

- The second way to create a thread is to create a new class that extends **Thread**, and then to create an instance of that class.
- Override run() method
- Call start() method

```
// Create a new thread.
class NewThread extends Thread
{
NewThread() { // Create a new, second thread
Super(“Demo thread”);
 System.out.println("Child thread: " + this);
start(); // Start the thread
}
```

Creating multiple threads

```
// Create multiple threads.
class NewThread implements Runnable
{
  String name;
Thread t;
NewThread(String threadname) {
t = new Thread(this, name);
   System.out.println("New thread: " + t);
   t.start(); // Start the thread
 }
```

```
// This is the entry point for the  thread.
 public void run()
   {
   try
   {
     for(int i = 5; i > 0; i--)
        {
         System.out.println("name: " + i);
   // Let the thread sleep for a while.
      Thread.sleep(1000);
        } }
    catch (InterruptedException e)
    {
       System.out.println(name +" interrupted.");
 }
System.out.println(name + "Exiting ");
} }
```

```
class MultiThreadDemo {
 public static void main(String args[]) {
  new NewThread(“one”); // start thread
 new NewThread(“two”);
 new NewThread(“three”);
  try {
        //wait for other threads to end
              Thread.sleep(10000);
         }
       catch (InterruptedException e)
        {
          System.out.println("Main thread interrupted.");
        } System.out.println("Main thread exiting.");
 } }
```

```
New thread : Thread[One,5,main]
New thread : Thread[One,5,main]
New thread : Thread[One,5,main]
One:5
Two:5
Three:5
One:4
….
One exiting
Two exiting
Three exiting
Main thread exiting
```