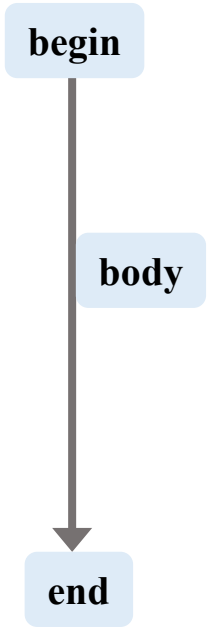
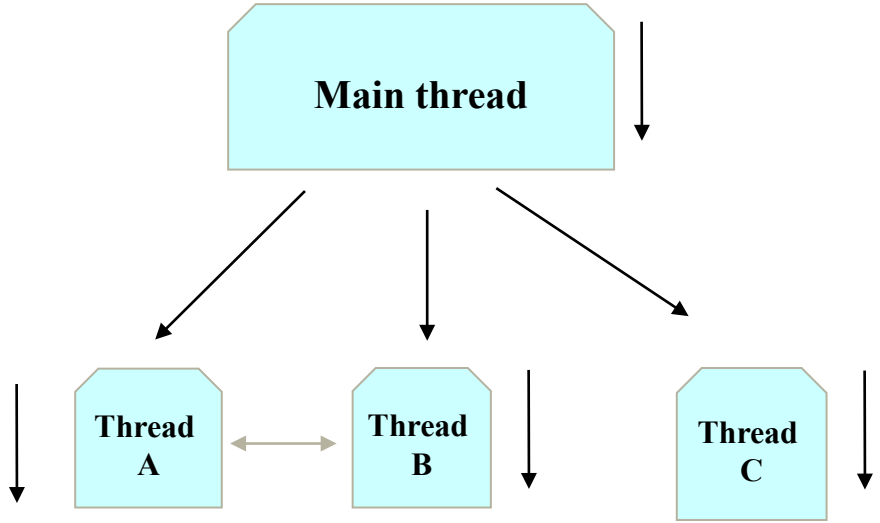
What is Multithreading?

A single threaded program

```
class ABC
{
    …
    public void main(..)
    {
        …
        …
    }
}
```



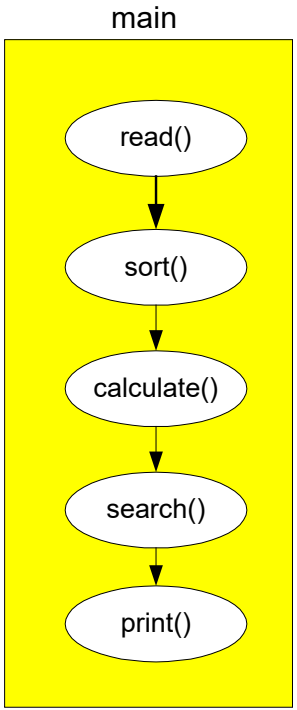
A multithreaded program

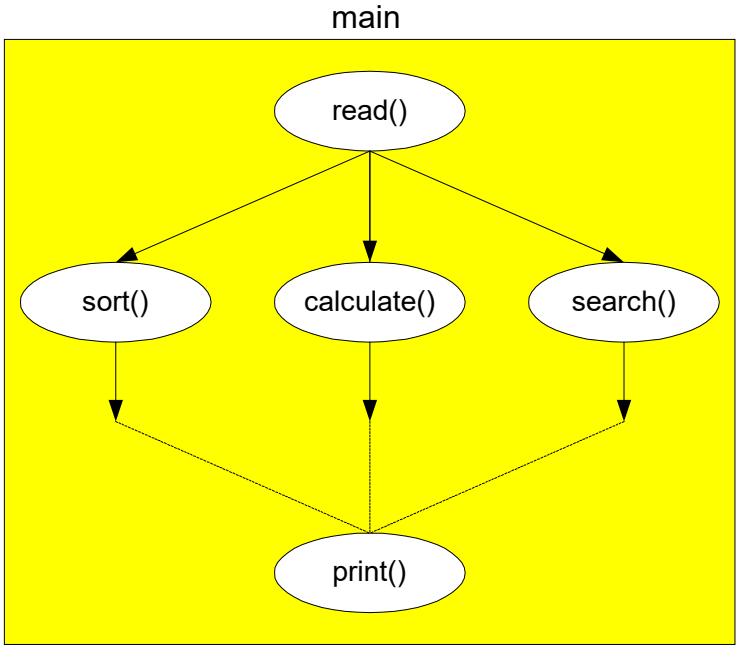


Threads may switch or exchange data/ results among them

A multithreaded program

```
public class X {
   main () {
      . read()
          { ...};
      . sort()
          {...};
      . calculate()
          {...};
      . search()
          {...};
      . print()
          {...};
}
```





The concept

Multiple tasks in computer

- Draw and display images on screen
- Check keyboard and mouse input
- Send and receive data on network
- Read and write files to disk
- Perform useful computation (editor, browser, game)

How does computer do everything at once?

- Multitasking
- Multiprocessing

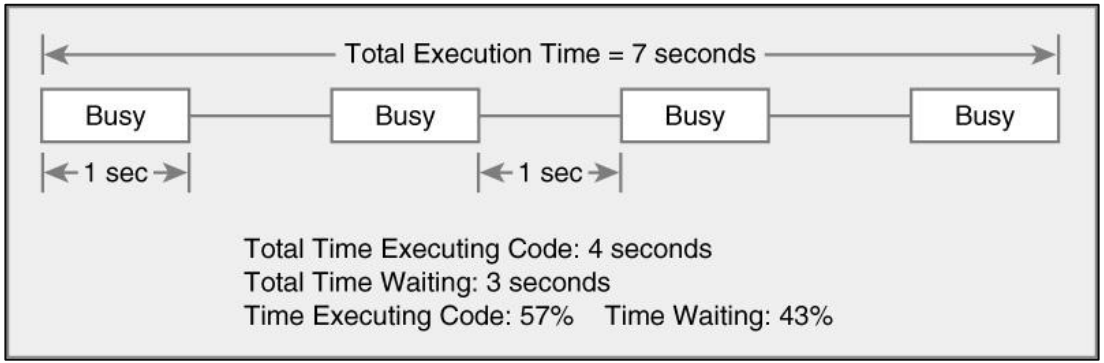
Multitasking (time-sharing)

- Approach
  - ➢Computer does some work on a task
  - ➢Computer then quickly switch to next task
  - ➢Tasks managed by operating system (scheduler)

- Computer seems to work on tasks concurrently
- Can improve performance by reducing waiting

Multitasking can improve performance

Single task →



Single task →

Two tasks →

Two tasks →

Multiprocessing (multi-threading)

- Multiple processing units (multiprocessor).
- Computer works on several tasks in parallel.
- Performance can be improved.

Dual-core AMD Athlon X2

32 processor Pentium Xeon

4096 processor Cray X1

Perform multiple tasks using…

Process

- Definition – executable program loaded in memory
- Has own address space : Variables and data structures (in memory)
- Each process may execute a different program
- Communicate via operating system, files, network
- May contain multiple threads

Thread

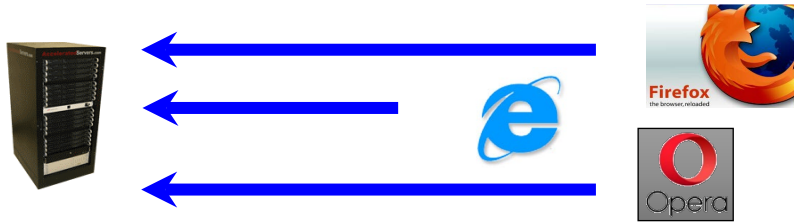
- Definition – sequentially executed stream of instructions
- Shares address space with other threads
- Has own execution context : Program counter, call stack (local variables)
- Communicate via shared access to data
- Multiple threads in process execute same program
- Also, known as "*lightweight process*"

Why Multithreading?

Motivation for multithreading

1. Captures logical structure of problem
   - ➢ May have concurrent interacting components
   - ➢ Can handle each component using separate thread
   - ➢ Simplifies programming for problem

Example



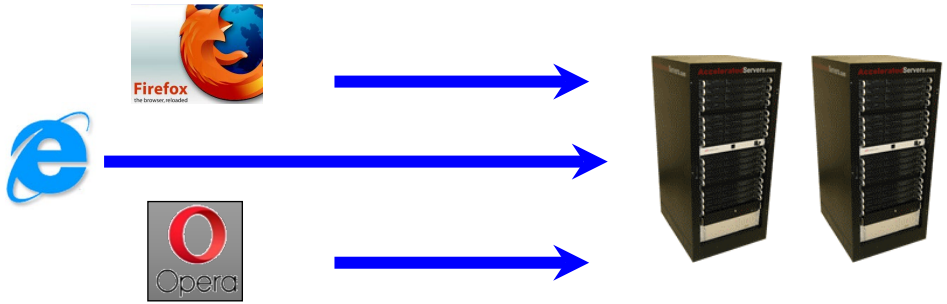
Web server uses threads to handle …

Multiple simultaneous web browser requests

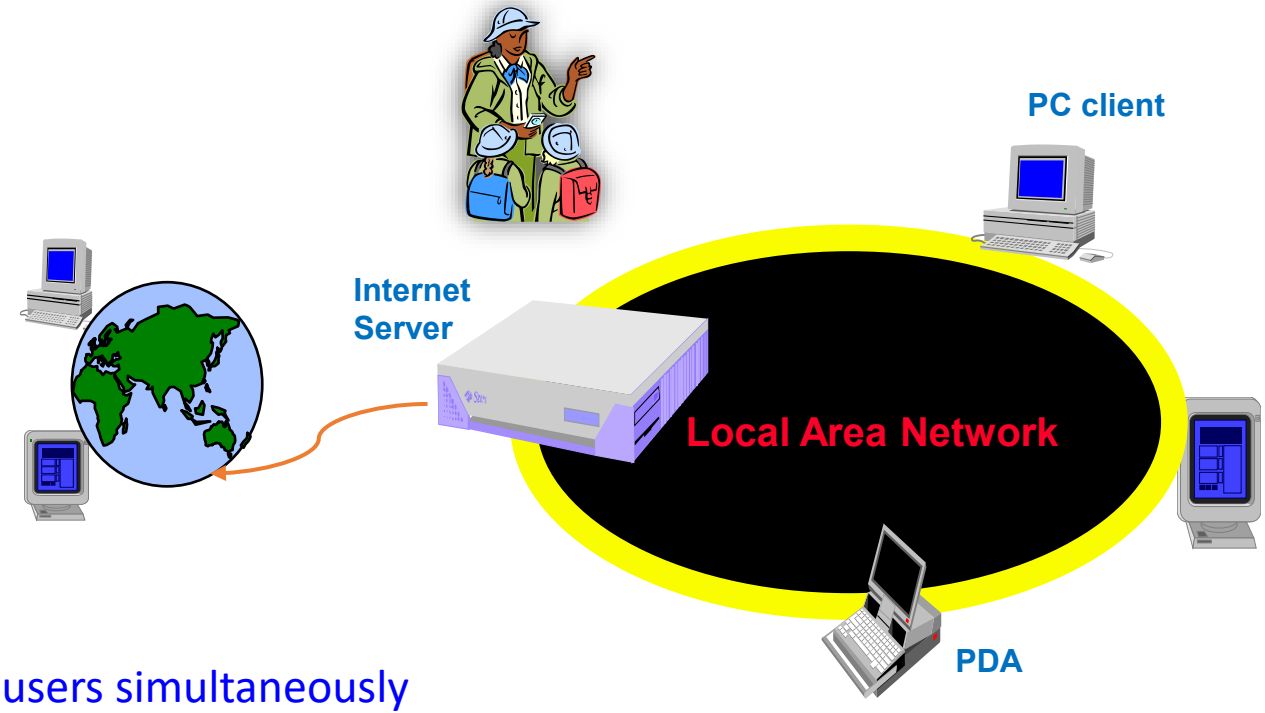
Motivation for multithreading

2. Better utilize hardware resources
   - When a thread is delayed, compute other threads
   - Given extra hardware, compute threads in parallel
   - Reduce overall execution time

Example

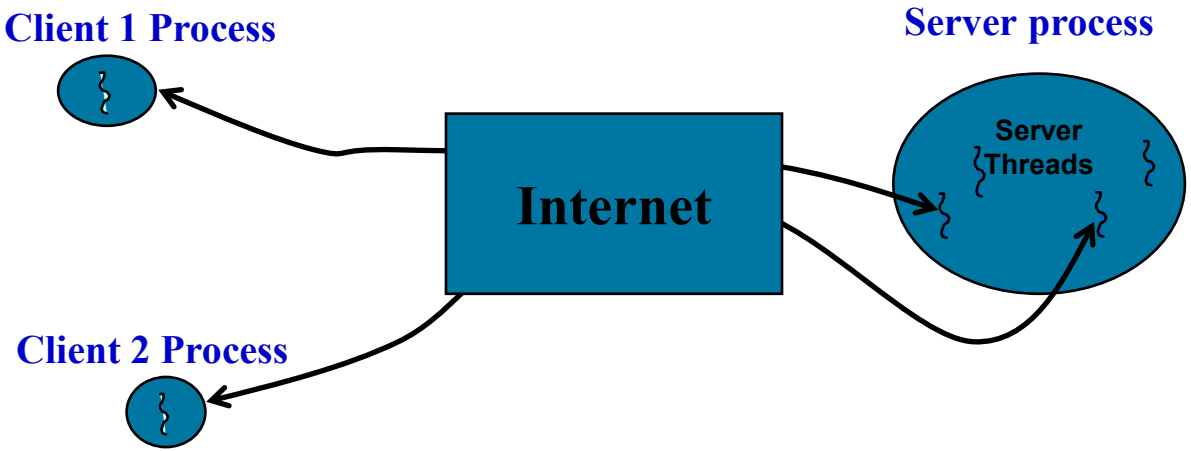


Web/ Internet applications



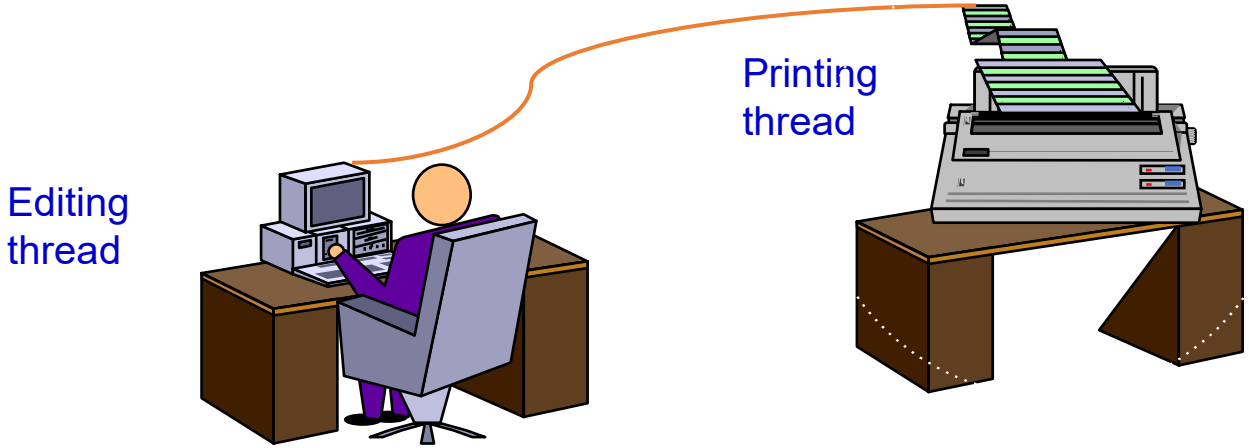
Serving many users simultaneously

Multithreaded server



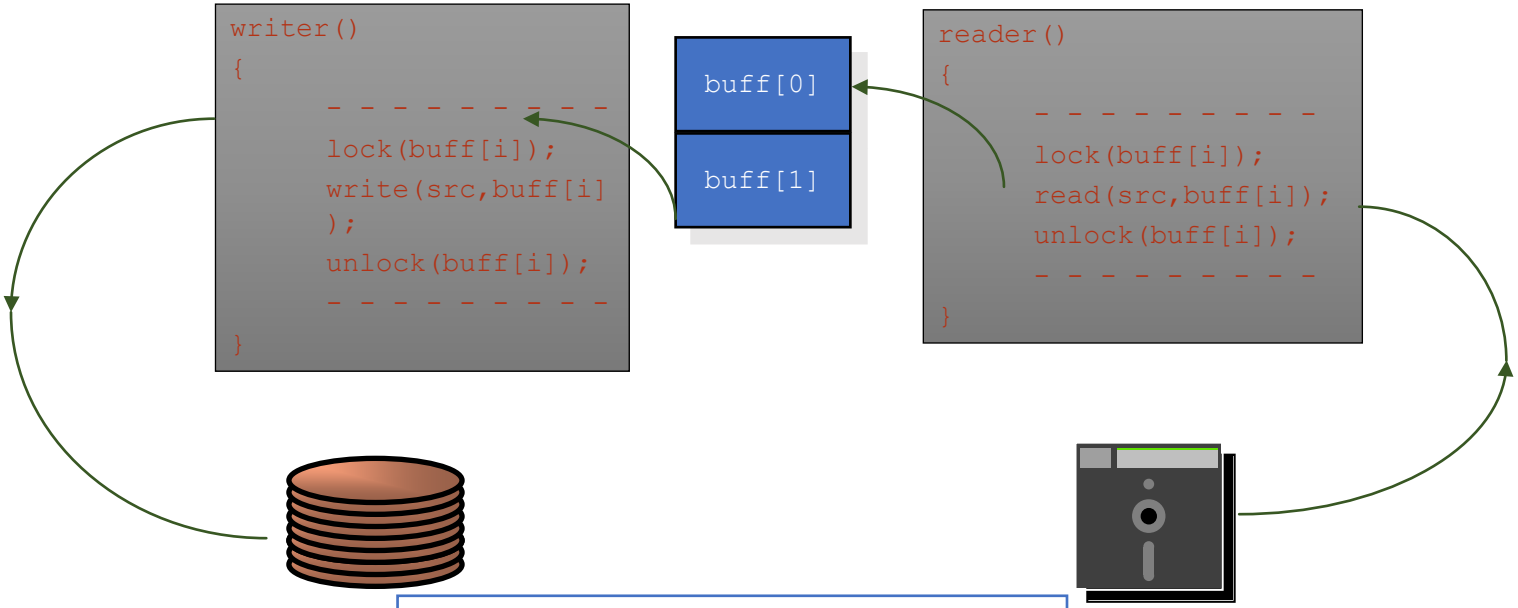
Serving multiple clients concurrently

Modern applications need threads



"Editing" and "Printing" documents are in background

Multithreaded/ Parallel file copy



Cooperative parallel *synchronized* threads

How Multithreading?

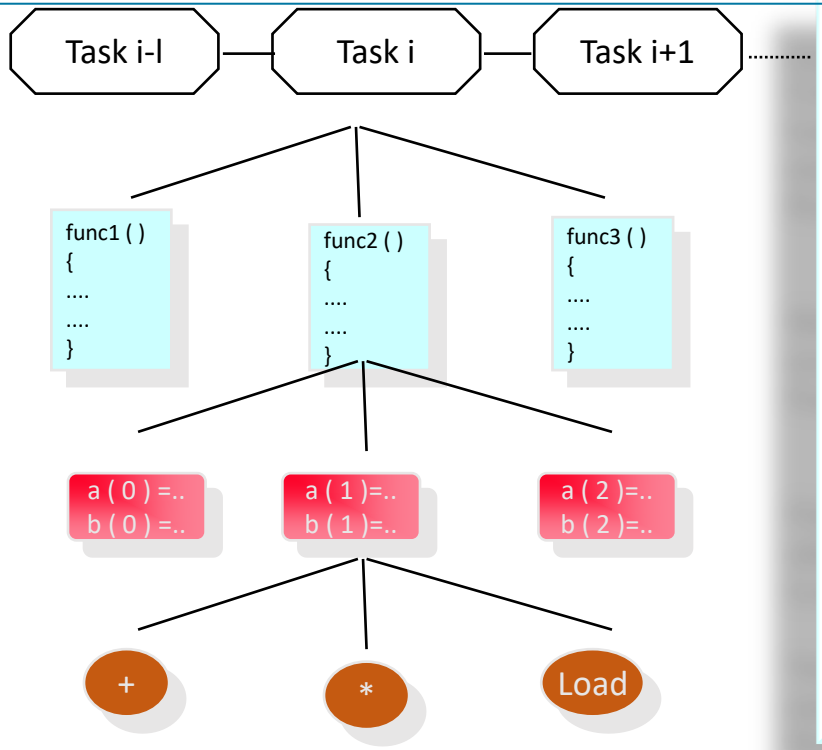
Levels of parallelism

Sockets

Threads

Compilers

CPU



Code-granularity
Code Item
Large grain
(task level)
Program

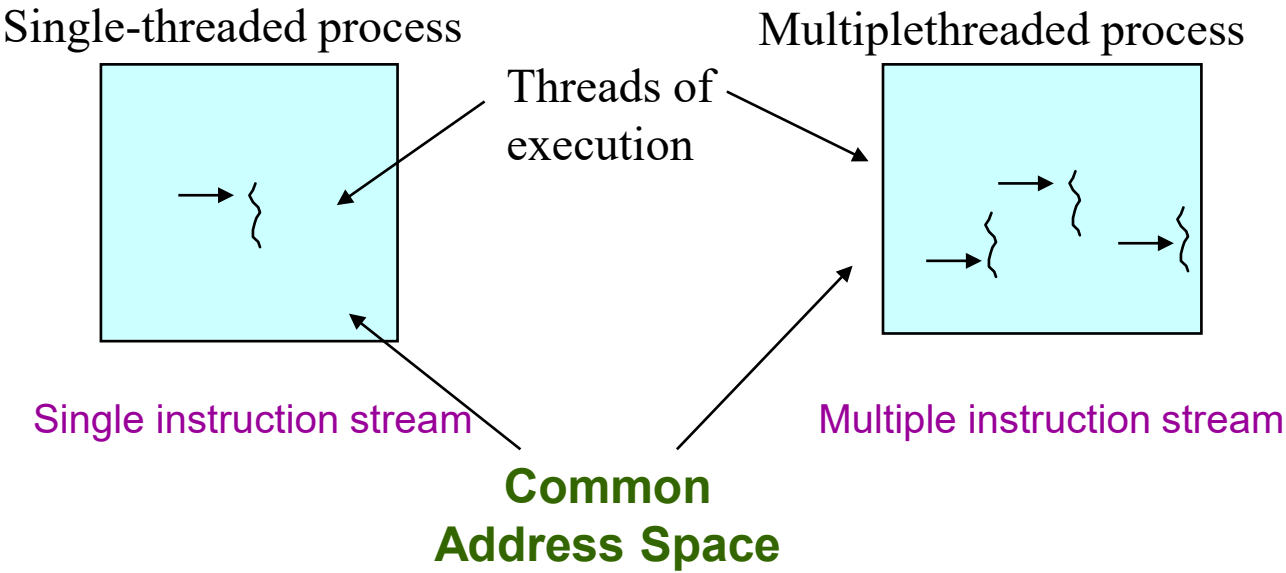
Medium grain
(control level)
Function (thread)

Fine grain
(data level)
Loop (Compiler)

Very fine grain
(multiple issue)
With hardware

Single and multithreaded processes

Threads are light-weight processes within a process



A thread is …

- A piece of code that runs concurrently with other threads.
- Each thread is a statically ordered sequence of instructions.
- Threads are being extensively used to express concurrency on both single and multiprocessor machines.
- Programming a task having multiple threads of control
  - ➢Multithreading or multithreaded programming.

Multithreading in Java

Java threads

- Java has built in support for multithreading.
  - ➢Synchronization
  - ➢Thread scheduling
  - ➢Inter-thread communication:

currentThread	start	setPriority
yield	run	getPriority
sleep	stop	suspend
resume		

- Java Garbage Collector is a low priority thread.

Everything about thread is readily defined in the package *java.lang* and in a class Thread and interface Runnable in it.

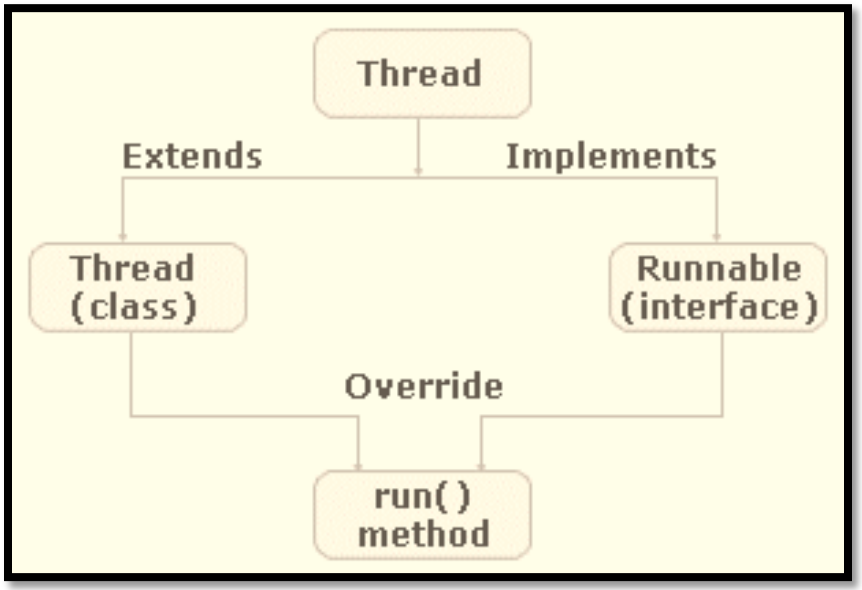
Running a thread in Java

Note:

- Thread starts executing only if start() is called

- Runnable is interface
  - So it can be multiply inherited
  - Required for multithreading in applets

Running a thread in Java



Creating Threads with `Threa`

Creating threads in Java

There are two ways to create and run a thread:

- Thread class

```
public class Thread extends Object { …
}
```

- Runnable interface

```
public interface Runnable
 {
        public void run( ); //work⇒ thread
 }
```

Thread class

```
public class Thread extends Object implements Runnable {
    public Thread();
    public Thread (String name); //Thread name
    public Thread (Runnable R);  //Thread R.run()
    public Thread (Runnable R, String name);
    public void run();     //if no R, work for thread
    public void start();  //begin thread execution
    ...
}
```

More Thread class methods

```
public class Thread extends Object {
        …
    public static Thread currentThread();
    public String getName();
    public void interrupt();
    public boolean isAlive();
    public void join();
    public void setDaemon();
    public void setName();
    public void setPriority();
    public static void sleep();
    public static void yield();
}
```

Creating thread in Java programs

1. Thread class
   - ➢ Extend Thread class and override the run method

   Example:

```
public class MyT extends Thread {
    public void run() {
          …                  // work for thread
    }
}
MyT T = new MyT () ;          // create thread
T.start();                    // begin running thread
…                             // thread executing in parallel
```

Creating thread : An example

```
class ThreadA extends Thread{
     public void run( ) {
        for(int i = 1; i <= 5; i++) {
           System.out.println("From Thread A with i = "+ -1*i);
        }
        System.out.println("Exiting from Thread A ...");
     }
}
class ThreadB extends Thread {
    public void run( ) {
       for(int j = 1; j <= 5; j++) {
         System.out.println("From Thread B with j= "+2* j);
       }
       System.out.println("Exiting from Thread B ...");
}
```

```
class ThreadC extends Thread

d C with k = "+ 2*k-1);

hread C ...");
```

```
lass ThreadC extends Thread

    public void run( ) {
        for(int k = 1; k <= 5; k++) {
            System.out.println("From Thread C with k = "+
*k-1);
        }
            System.out.println("Exiting from Thread C
..");
    }

lass MultiThreadClass

   public static void main(String args[]) {
        ThreadA a = new ThreadA();
        ThreadB b = new ThreadB();
        ThreadC c = new ThreadC();
        a.start();
        b.start();
        c.start();
        System.out.println("... Multithreading is over ");
    }
}
```