

## Agenda

- i) Program vs Process
- ii) Concurrency vs Parallelism
- iii) Threads
- iv) How to start with multi-threading
- v) Demo example

Program:- Something that can be executed to cause a particular action.

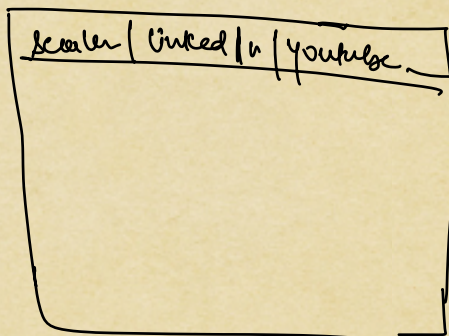
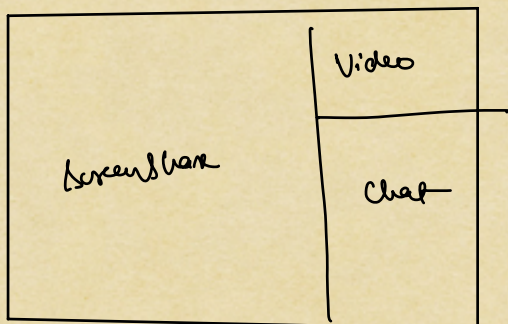


Process :- When a program runs in CPU we call it a process.



\* executes on the processor

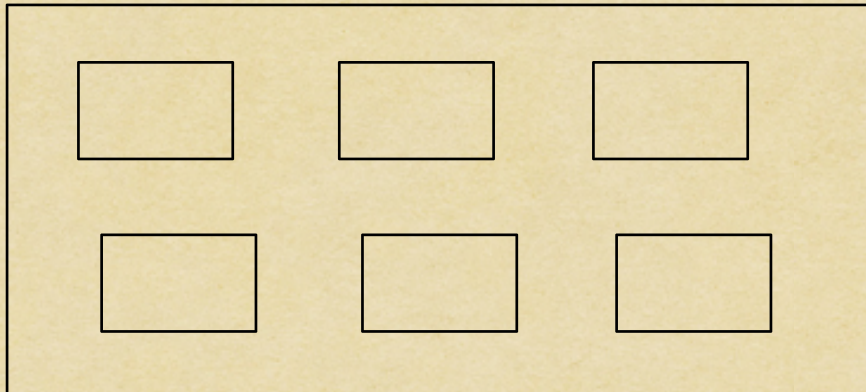
\* all data of an executing process is stored at RAM.





Quiz 1

Cores



Cores = brain  
↓  
individual  
execution pts in a processor

Threads

→ Unit of CPU execution  
↓  
basic / smallest

$4 + 5 = 9$   
↓  
print

\* A core executes a thread.

\* Every process to execute needs atleast 1 thread.

→ Any process can have N no. of threads [even if processor has single core]

→ Any core at a time can execute 1 thread

↓  
hyperthreading ⇒ 2 threads

In general, if you have N cores ⇒ 2N threads



Win 98  
↓  
Pentium 4  
↓  
1 core CPU

Dual Core / Core 2 duo  
↓  
2 cores

i3 / i5 / i7 / i9  
↓  
4 core - 16 cores

M1 / M1 Pro / M1 Max — M2 / M2 Pro / —

AMD's Threadripper ⇒ 64 cores

How could we run multiple things in a single core :-

Ans ⇒ Concurrency

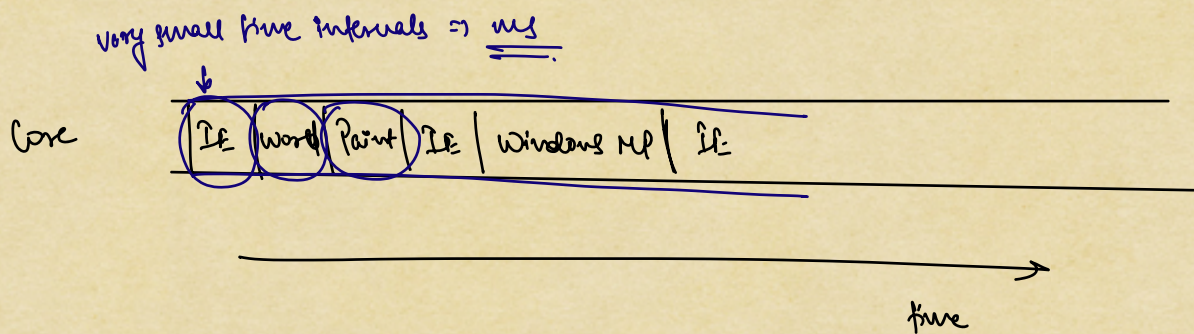
## % Concurrency vs Parallelism

⇒ We always want to optimise processor usage, there should be no idle scenario for processor  
↓  
not doing anything

⇒ Print ⇒ I/O printer  
Disk ⇒ I/O CD reader  
App from user

process  
waiting state  
↓  
context switch  
↓  
switching b/w multiple processes on a single core

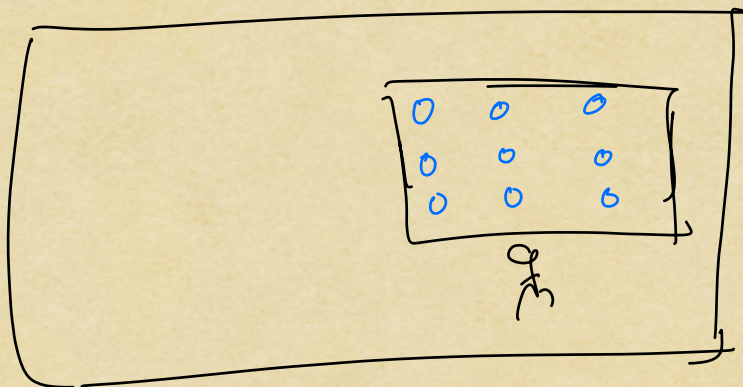




Concurrency  $\Rightarrow$  Machine executes a certain set of tasks, parallelly [illusion of parallel execution] over a window of time but not running at the same time.

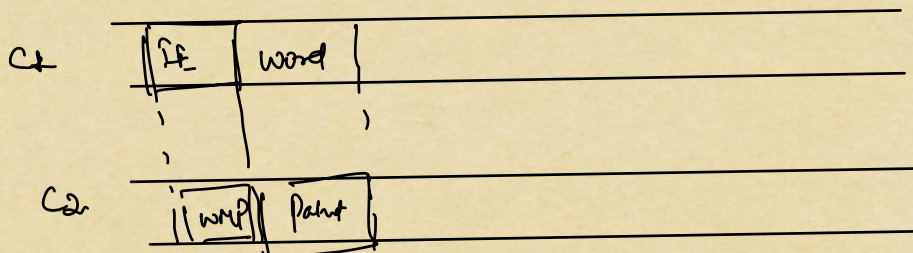
CPU scheduler handles all execution.

Baroque Nation:



Parallelism:

Machine executes one or more set of tasks parallelly, we need more than 1 core for parallelism.





### Ques-2

#### Concurrency

- \* execution one task at a time
- \* seems parallel over a window of time period
- \* not truly parallel
- \* can be with  $\geq 1$  core

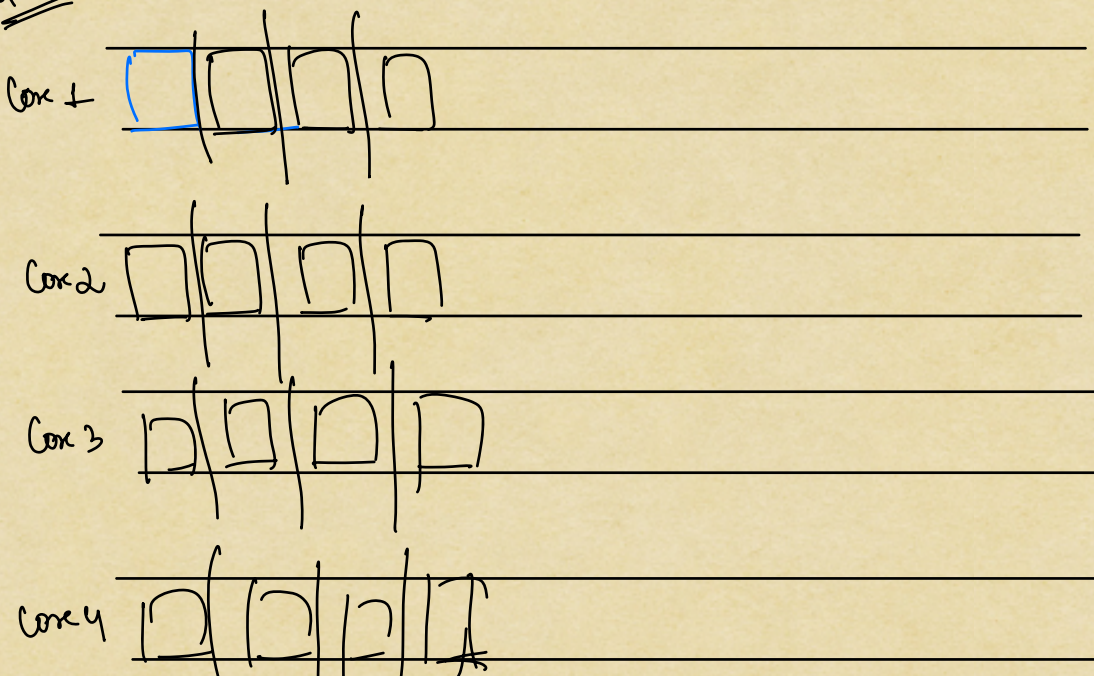
#### Parallel

- \* execution of more than 1 task at the same time
- \* truly parallelism
- \* can be done with  $\geq 1$  core

### Ques-3

In real world, combn of both Concurrency & Parallelism

Fixed core





\* Can we have parallelism when we have concurrency?

Ans  $\Rightarrow$  might / might not

\* Can we have concurrency when we have parallelism?

Ans  $\Rightarrow$  Yes

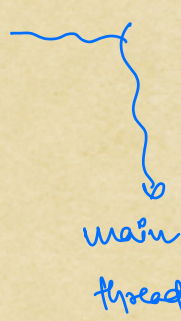
$\Rightarrow$  How to write multi-threaded code?

\* Don't think in terms of how many / what threads you need.

\* Think in terms of what tasks you want to do in parallel.

Ques 1 Create a multi-threaded program where we can print Hello world from a separate thread.

```
class Main {  
    psum() {  
        sout("Hello world");  
    }  
}
```



main thread



Step 1 Identify the task that you want to do parallelly  
print Hello world

Step 2 Create classes for all the tasks that has been identified  
class HelloWorldPrinter

\* name of the class should be a noun doing a verb.

Step 3 Make the class implement runnable interface

class HelloWorldPrinter implements Runnable

\* Runnable interface has a method called run(); this method holds the code for the task that we want to do.

Step 4 ⇒ Implement the run method

```
class HelloWorldPrinter implements Runnable {  
    public void run() {  
        System.out.println("Hello world");  
    }  
}
```

⇒ Task definition from Step 1 to 4



Steps

Thread creation and execution:-

a) go to the place from where you want to execute the code

b) Create the object of the task class

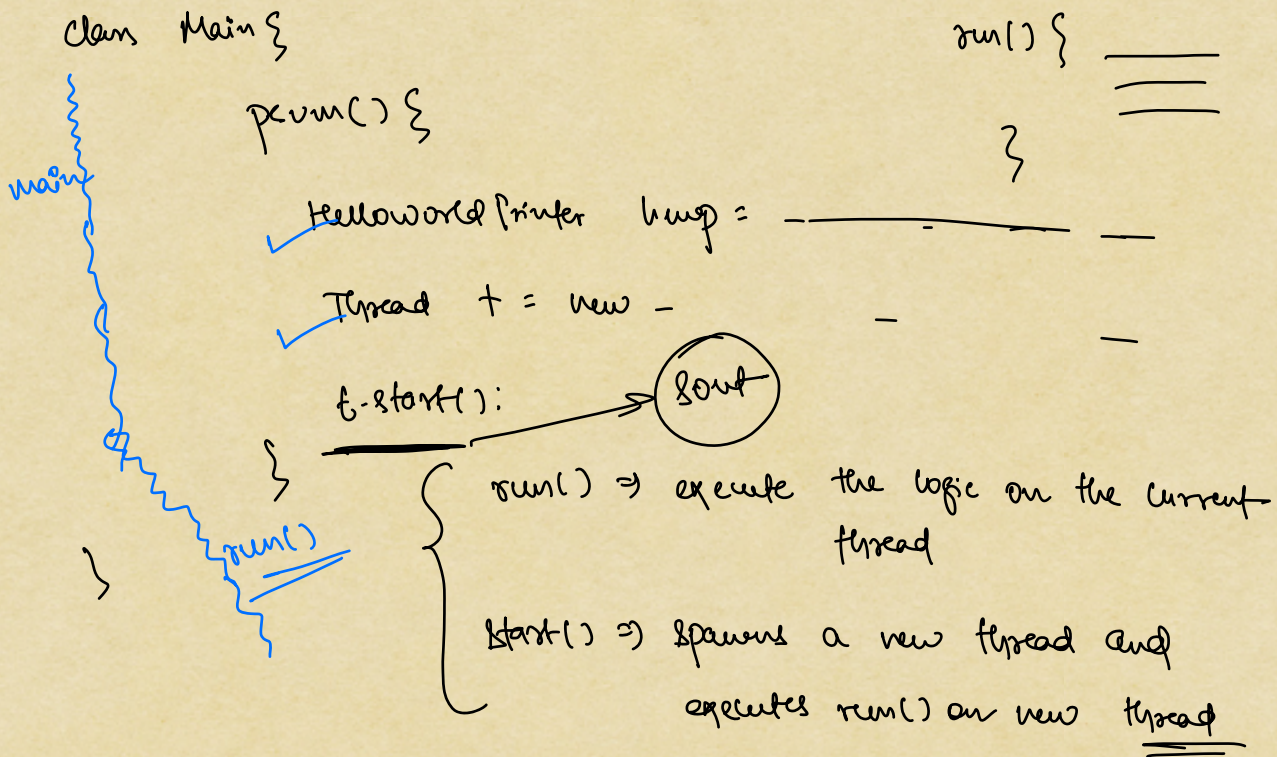
`HelloWorldPrinter hwp = new HelloWorldPrinter();`

c) create a thread using task object

`Thread t = new Thread(hwp);`

d) start the thread

`t.start();`





## Adder Subtractor

Ques 2 Take 2 numbers as input from user, and add the numbers and subtractor the number and prints result in 2 diff. threads.

i/p  $\Rightarrow$  2 number  $\Rightarrow$  main

t0  $\Rightarrow$  add & print

t1  $\Rightarrow$  subs & print

Since, we cant pass data to run, we pass data to the object of the task class