DAY-58

------

MULTI-THREADING

----------------------------------------

REFER fig:1

--> Single tasking is a process of executing only one task at a time.

On a ram if single task/process is getting executed then it is reffered as single tasking operating system.

note: the program under execution on the ram is called as task/process.

REFER fig:2

--> Multi-tasking is a process of executing several task simultaneously.

On a ram if multiple task/process is getting executed then it is reffered as multi tasking operating system.

REFER fig:3

--> In the above scenario only a single ram is used inside the mother board to execute the process/task, by doing so the cpu time is utilized properly

--> Until one task gets executed the remaining files will be in the waiting state, hence the execution time of all the files will be decresed.

in order overcome this probelm :

REFER fig:4

--> In the above scenario for every task a seprate processor is kept which will solve the problem of execution time.

But, The biggest disadvantage of this technique is when ever process/task are dependent on another task then it leads to a probelm called as

'mutal exclusive resources'.

NOTE: fig:4 is also refferd as "PARALLEL PROCESSING".

MULTI-TASKING operating systems [1990's]:

-----------------------------------------

--> In case of multi-tasking the process of switching from one task to another task is reffered as "context switching".

--> Process schedular present within the operating system is responisble for context switching.

--> basically to execute multi-task on a single processor this technique is used.

note: The multi-tasking is also called as "concurrent processing".

THREAD:

--------

--> A thread is a smallest unit of process/task.

--> A thread is light weight process

--> The line of execution or path of execution or the flow of execution is called as 'THREAD'

note: all the task by default will have one thread and it is called as 'main thread'.

SINGLE- THREADED APPLICATION:

------------------------------

EXAMPLE:

--------

// single threaded application.

import java.util.\*;

class sample1

{

public void sq\_num() throws InterruptedException

{

for (int i=0;i<5;i++)

{

Thread.sleep(2000);

System.out.println("sq\_num"+(i\*i));

}

}

}

class sample2

{

public void double\_num() throws InterruptedException

{

for (int i=0;i<5;i++)

{

Thread.sleep(2000);

System.out.println("double\_num"+(i\*2));

}

}

}

class Demo

{

public static void main(String[] args) throws InterruptedException

{

System.out.println("main method..!");

sample1 s1= new sample1();

sample2 s2 = new sample2();

s1.sq\_num();

s2.double\_num();

}

}

OUTPUT:

--------

main method..!

sq\_num0

sq\_num1

sq\_num4

sq\_num9

sq\_num16

double\_num0

double\_num2

double\_num4

double\_num6

double\_num8

--> If a single thread is used to execute all the methods in a program or application then it is called as "single threaded application."

REFER fig:7

--> OS with the help of process schedular allocates the specific time for every process/task present on the RAM

In order to execute a java program OS will allocates one region ie JRE[java runtime environment]

The time given by the PS will be collected by Thread schedular then it will release main thread to execute the given task.

Thread schedualr:

--------------------

It is a s/w which will control and co-ordinate the lines of program present within a java file.

It responisblity is to perform context swicthing among threads.

Function blocking call:

-----------------------

It is mechanism where one function/method is blocked for execution untill another function/ method executes even though the methods are indepedent

of each other.

MULTI-THREADED APPLICATION:

----------------------------

--> TO over come abobe probelem we can create multiple threads in the program

separate threads for execution of independent methods is called as multi-threaded application.

Creation of threads in java:

-------------------------------

1. creation of threads by extending the thread class

2. by using the runnable interface

1.creation of threads using thread class:

------------------------------------------

EXAMPLE:

-------

// creation of threads by extending thread class

import java.util.\*;

class MyThread1 extends Thread

{

public void run()

{

try

{

for (int i=0;i<5;i++)

{

Thread.sleep(2000);

System.out.println("sq\_num"+(i\*i));

}

}

catch (InterruptedException e)

{

}

}

}

class MyThread2 extends Thread

{

public void run()

{

try

{

for (int i=0;i<5;i++)

{

Thread.sleep(2000);

System.out.println("double\_num"+(i\*2));

}

}

catch (InterruptedException e)

{

}

}

}

class Demo1

{

public static void main(String[] args)

{

System.out.println("main method..!");

MyThread1 t1= new MyThread1();

MyThread2 t2 = new MyThread2();

t1.start();

t2.start();

//t1.run();

//t2.run();

}

}

OUTPUT:

-------

main method..!

double\_num0

sq\_num0

sq\_num0

double\_num2

sq\_num1

sq\_num1

sq\_num4

double\_num4

sq\_num4

sq\_num9

sq\_num9

double\_num6

sq\_num16

sq\_num16

double\_num8

double\_num0

double\_num2

double\_num4

double\_num6

double\_num8

REFER fig:8