Multithreading

MultiTasking:

Executing more than one task simultaneously is called as multitasking.

Task:

Task is end goal to achieve or task is collection of multiple process.

MultiProcessing:

Executing more than one process simultaneously is called as multiprocessing.

Process: collection of multiple threads.

MultiThreading:

Executing multiple threads simultaneously is called as multi-threading.

Thread:

Thread is a light weight process or it is a smallest individual part of a task.

Two ways of creating Threads:

1.By extending thread-class:

A thread can be created by using thread class.

We need to override run() from runnable interface.

Thread class is implementing class of runnable interface.

In order to execute or run user defined thread we need to make use of start().

2.By implementing Runnable interface

Important Thread Methods:

Start():

Start method is non-static method present inside Thread class.

To allocate a thread dedicated stack we need to use start().

NOTE: newly created thread will be allocated a dedicated stack area but if we called user created run() without using start() in that case dedicated stack area will not be allocated to that thread.

Thread Life Cycle:

1. start
2. Running
3. Waiting
4. Stop

Thread schedular is part of jvm manages the life cycle of thread.

Executing Multiple Threads simultaneously:

Properties of threads:

1.Id Property:

The value for the Id property is given by the jvm. We cannot change value for the id property but we can fetch its value by using getId() method.

getId():

getId it is not-static method present inside Thread class. Return type of getId method is long.

2. Name Property:

This property of thread will have default value for every newly created thread.

We can fetch its value by using getName() method and assign new value to it by using setName() method.

getName():

It is non-static method present inside Thread class. Return type of getName() is String.

setName():

It is non-static method present inside Thread class. It can accept the arguments of String type.

3. Priority :

Default value for newly created thread will be 5. The maximum priority value is 10 and minimum value is 1. We can fetch its value by using getPriority() and assign new value to it by using setPriority().

getPriority():

It is non-static method present inside Thread. Return type of getPriority() method is int.

setPriority():

It is non-static method present inside Thread class. It can accept arguments of int type.

Data inconsistency:

When more than one threads are accessing shared resources being unaware of each others operations, then data inconsistency can occur.

To avoid this data inconsistency , we use synchronization.

Shared resources:

A shared resources is a resource which is accessed by more than one threads. A method or variable can be referred as shared resource.

Synchronization:

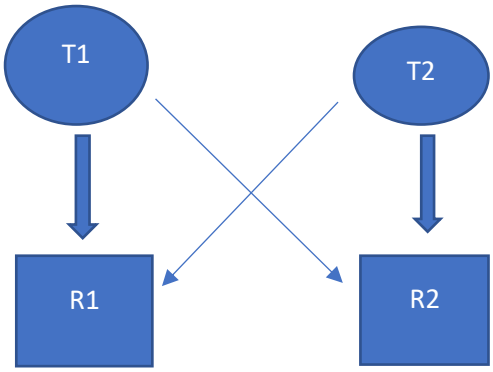
Synchronization can be achieved by using synchronized keyword.

If resource is synchronised then the thread which is accessing it will apply lock on it, so that another thread cannot access those resources at the same time.

Stop(): It is used to stop execution of thread.

Sleep(): sleep() accepts long type argument i.e. time in milliseconds. It is used to stop thread execution for specific time period.

Dead lock:



- In above diagram thread1 will apply the lock on resouce1.

: - thread2 will apply the lock on rsource2.

: -Thread1 wants to access resource2, and at the same time thread2 wants to access resource1 but thread1 will not release the lock on resource1 until and unless it gets access to resource2.

: -Similarly, thread2 will not release the lock on resource2. untill and unless it gets access to resource1.

: -Its leads to one abnormal situation, called as Dead lock. where program will not get terminated.

Daemon Thread:

In java, the threads created by extending the Thread class or by implementing the runnable interface are considered as user defined threads.

: - Although there are some predefine threads in java which are not created by the user and are known as Deamon Threads.

: - The Deamon threads are called for execution implicitly by the JVM whenever required.

: - The execution of the Deamon thread is not visible to the programmer. Deamon threads are low priority threads.

Garbage Collection: -

: - GC is one of the most important Deamon threads in java, it helps to make java strong in memory management.

: - The JVM implicitly called the GC thread, whenever a garbage value is detected in the memory.

: - the GC thread identifies the garbage value and removes it from the memory.

1. What is multi-threading
2. Why we use multithreading
3. What is synchronization
4. What is deadlock
5. Create a thread(by using both the ways)
6. Create a thread to represent thread properties.
7. What is use of synchronised keyword.