**Multithreading – Part-13 – Daemon Thread**

* **Daemon Threads:**

The threads which are executing in the background are called Daemon Threads.

Example:

Garbage Collector

Signal Dispatcher

Attach Listener

The main objective of daemon threads is to provide support for non-daemon threads (main thread). For example, if main thread runs with low-memory then JVM runs Garbage Collector to destroy useless objects, so that number of bytes of free memory will be improved. With this free memory main thread can continue its execution.

Usually daemon threads have low priority but based on our requirement daemon threads can run with high priority also.

We can check daemon nature of a thread by using isDaemon() method of thread class.

public Boolean isDaemon()

We can change daemon nature of a thread by using setDaemon() method.

public void setDaemon(Boolean daemon)

But changing daemon nature is possible before starting the thread, after starting a thread if we try to change the daemon nature then we will get runtime exection saying:

IllegalThreadStateException

* **Default Nature of a Thread:**

By default, main thread is always non-daemon and for all remaining threads daemon nature will be inherited from parent to child. That is if the parent thread is daemon then automatically child thread is also daemon. And if the parent thread is non-daemon the automatically child thread is also non-daemon.

Note:

It is impossible to chance daemon nature of main thread. Because it is already started by JVM at beginning.

* **Example:**

class MyThread extends Thread{

}

class Test{

public static void main(String[] args){

System.out.println(Thread.currentThread().isDaemon);

//false

Thread.currentThread().setDaemon(true);

//RE: IllegalThreadStateException

MyThread t = new MyThread();

System.out.println(t.isDaemon()); //false

t.setDaemon(true);

System.out.println(t.isDaemon()); //true

}

}

* **Note:**

Whenever last daemon thread terminates automatically all daemon threads will be terminated, irrespective their position.

Example:

class MyThread extends Thread{

public void run(){

for(int i = 0; i< 10; i++){

System.out.println(“Child Thread”);

try{

Thread.sleep(2000);

}catch(InterruptedException ie){}

}

}

}

class DaemonThreadDemo{

public static void main(String[] args){

MyThread t = new MyThread();

t.setDaemon(true); // Line-1

t.start();

System.out.println(“End of main thread”)

}

}

Note:

If we are commenting Line-1 both main and child are non-daemon and hence both threads will be executed until their completion.

If we are not commenting Line-1 then main thread is non-daemon and child thread is daemon, hence whenever main thread terminates automatically child thread will be terminated. In this case output is:

End out main Thread

Child Thread

Or

End of main Thread

Or

End of main Thread.