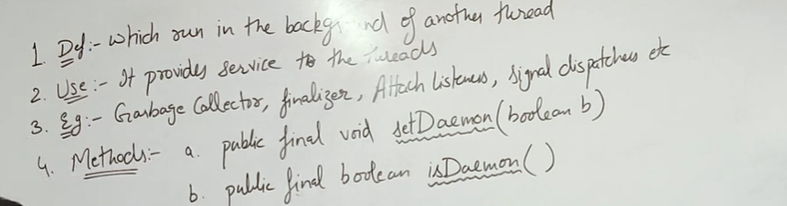
1>>>



Daemon Thread-> i)Runs in the background of other thread.

ii)Provides service to the threads.

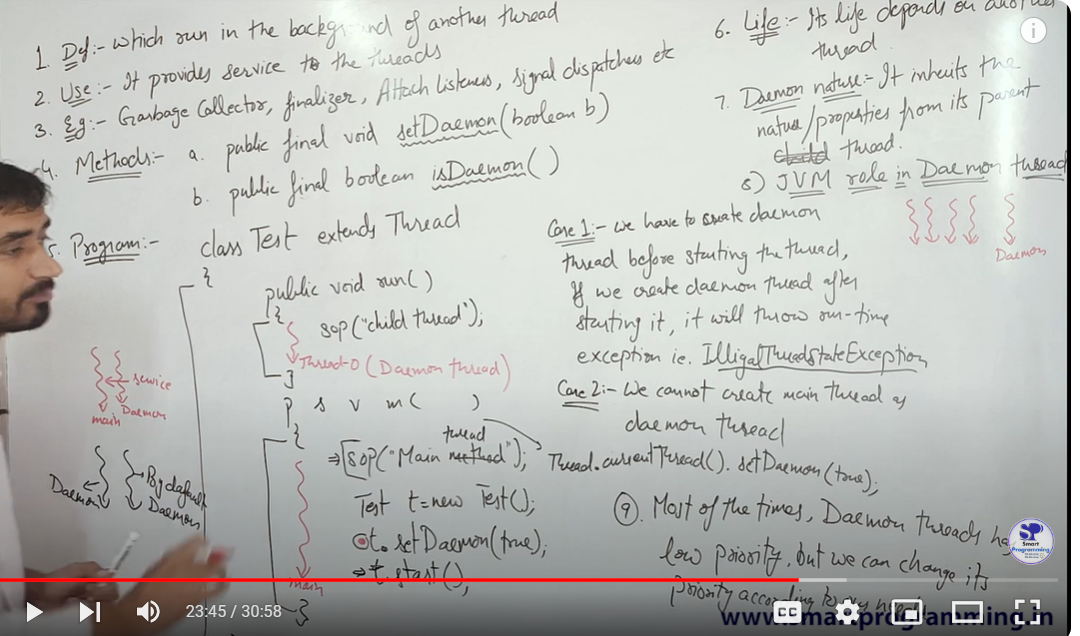
iii)E.g-Garbage collector , initializer , attach listeners , signal dispatchers etc.

The memory management of main method is handled by garbage collector if there is leak or need in memory .It deletes waste resources from the memory and free it . And hence Garbage Collector method runs in background and provides service( memory management ) to main method.

Other example like typing in notepad , implicitly highlights spelling errors by the use of a daemon method spelling checker.

iv) It has 2 methods -> setDaemon ( Marks the thread as a daemon thread, which runs in the background and terminates when all non-daemon threads finish. ) and isDaemon()

Checks if the thread is a daemon thread.



v)JVM role in Daemon threads: Daemon threads are terminated by the JVM when all user (non-daemon) threads have completed execution. The JVM does not wait for daemon threads to finish, meaning if only daemon threads remain, the program will exit or JVM shuts down/ terminate instantly once all daemon threads died by terminating non daemon threads instantly.

vi) Life: The lifecycle of a daemon thread is dependent on other threads, particularly the user threads. Once the user threads are finished, the daemon threads automatically stop, as they only serve to provide support to user threads.

E.g in above code thread created using object t will terminate only after main method ends since it needs to provide services till main method lifetime.

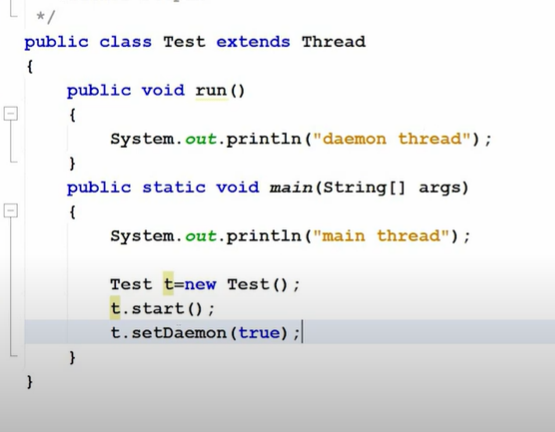
vii) n Java, a daemon thread is a background thread that inherits its properties from its parent thread. If a parent thread is a daemon, any thread it creates will also be a daemon by default unless explicitly set otherwise using setDaemon(false). Daemon threads support other threads, and they are terminated when all user threads finish execution.

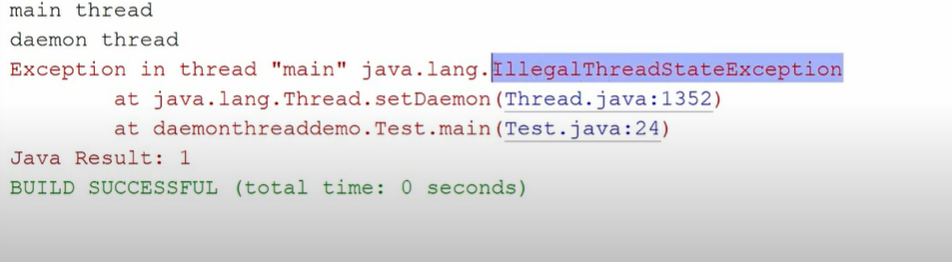
For non-daemon threads (also called user threads), the child thread inherits the non-daemon property from the parent thread. To make a child thread daemon inside non daemon thread we need to explicitly define setDaemon of child thread as true . Non-daemon threads keep the JVM alive until they complete, meaning the program will not terminate until all non-daemon threads have finished.

viii) Daemon thread priority: Generally, daemon threads are assigned lower priority compared to user threads, but their priority can be changed programmatically if necessary.

PROGRAMS :

a> A daemon thread must be marked as such before it is started. If you try to turn a thread into a daemon after starting it, Java will throw an IllegalThreadStateException. ( 1st setDaemon(true) then start() not opposite )





b>So we can never make main thread to be daemon .

