1)JVM provides the priority to each thread & according to these priorities, JVM allocates the processor. If 2 properties have same priority , it depends on JVM and other algorithms which one will be assigned CPU first.

2) Priorities are represented in the form of integer values which range from 1–10:

1 → MIN\_PRIORITY

5 → NORM\_PRIORITY

10 → MAX\_PRIORITY

Below are not the priorities:= 0, < 1, > 10

Priority levels :MINIMUM\_PRIORITY LOW\_PRIORITY . MEDIUM\_PRIORITY etc are not valid.

3)Methods:

public final void setPriority(int value)

public final int getPriority()

4) Nature: Priorities are inherited from the parent thread . By default, the main thread's priority is 5. Hence any thread we create inside main thread has priority value = 5 by default.

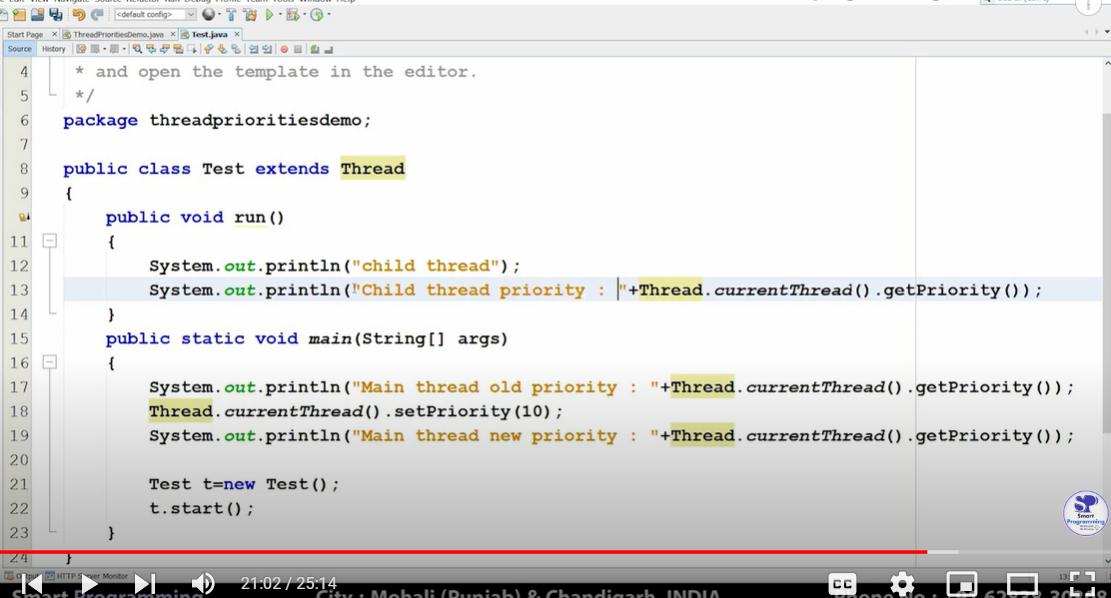
5) If priority value is not between 1 and 10, it will throw an exception, specifically IllegalArgumentException.

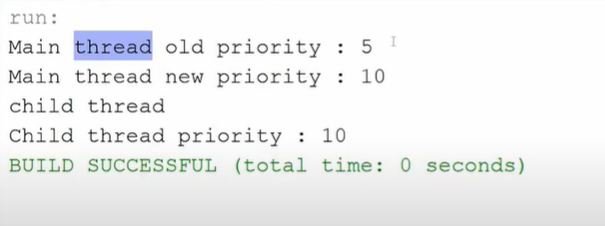
6)Priorities depends on platform. Windows do not support priorities . It means we should follow priority depending on user needs but windows will not execute in correct order of priority since it does not support it.

7) When a thread is created inside the main method, it inherits the priority of the main thread at the time of its creation. Any changes to the main thread's priority after the child thread has been started will not affect the child thread’s priority. The child thread retains the priority it inherited at the time of its creation, regardless of subsequent changes to the main thread's priority.

8) If a thread is created inside the main method but is started after the main thread's priority is changed, the child thread will inherit the updated priority of the main thread at the time it is started. In this case, the child thread reflects the current priority of the main thread when start() is called, not when the thread is created.

A>>





\*\*\*IF t.start() was started earlier before changing thread priority to 10 , then still its priority waould have been 5.

B>>

