Muhammad Amsyar bin Abdul Malek

2022020070028

Write a description of the commonly used methods of Thread class, which are:

public void start()

- Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.
- The result is that two threads are running concurrently: the current thread (which returns from the call to the start method) and the other thread (which executes its run method).
- It is never legal to start a thread more than once. In particular, a thread may not be restarted once it has completed execution.

public void run()

- If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns.
- Subclasses of Thread should override this method.

public void sleep()

- The method used to put thread in that currently in executing to sleep or temporarily cease execution for specified number of milliseconds, but subject to the precision and accuracy of the system timers and schedulers.
-

public void join()

- Waits at most millis milliseconds for this thread to die. A timeout of 0 means to wait forever.
- This implementation uses a loop of this.wait calls conditioned on this.isAlive. As a thread terminates the this.notifyAll method is invoked. It is recommended that applications not use wait, notify, or notifyAll on Thread instances.

public void yield()

- A hint to the scheduler that the current thread is willing to yield its current use of a processor. The scheduler is free to ignore this hint.
- Yield is a heuristic attempt to improve relative progression between threads that would otherwise over-utilise a CPU. Its use should be combined with detailed profiling and benchmarking to ensure that it actually has the desired effect.

public void stop()

- The thread represented by this thread is forced to stop whatever it is doing abnormally and to throw the Throwable object obj as an exception. This is an unusual action to take; normally, the stop method that takes no arguments should be used.
- It is permitted to stop a thread that has not yet been started. If the thread is eventually started, it immediately terminates.

public void interrupt()

- Tests whether the current thread has been interrupted. The interrupted status of the thread is cleared by this method. In other words, if this method were to be called twice in succession, the second call would return false (unless the current thread were interrupted again, after the first call had cleared its interrupted status and before the second call had examined it).
- A thread interruption ignored because a thread was not alive at the time of the interrupt will be reflected by this method returning false.

public int getPriority()

- Returns this thread's priority.

public int setPriority(int priority)

- Changes the priority of this thread.
- First the checkAccess method of this thread is called with no arguments. This may result in throwing a SecurityException.
- Otherwise, the priority of this thread is set to the smaller of the specified newPriority and the maximum permitted priority of the thread's thread group.
-

public String getName()

- Returns this thread's name.

public void setName(String name)

- Changes the name of this thread to be equal to the argument name.
- First the checkAccess method of this thread is called with no arguments. This may result in throwing a SecurityException.

public Thread currentThread()

- Returns a reference to the currently executing thread object.

public boolean isAlive()

- Tests if this thread is alive.