HW10.1

There are two approaches to initiate a thread: one involves the class implementing the thread extending the Thread class,

while the other includes the class implementing the Runnable interface.

In this code, we are opting to extend the Thread class.

The subsequent step is to override the run method, which is a part of the Runnable interface, and this interface is implemented by the Thread class, as Thread inherently implements the Runnable interface.

In this code, we create three instances of the ThreadQuestionOne class. To start thread execution, we need to invoke the start method.

The start method, in turn, invokes the run method, which contains the code executed during thread execution.

Upon initiation, the scheduler allocates a location for the thread object transitions it to the ready state.

While the run method is executing the code, the scheduler can concurrently create an instance for another thread and assign its priority.

The line aT4_0.run(); initiates the thread and directly enters the running state. It operates as a regular function call that has been overridden, rather than functioning in a multi-threaded environment.

Nevertheless, this code will eventually trigger the condition if ( info == "first" ), where the code for starting a new Thread 'fourth' is located, initiating a new thread.

This new thread will follow the thread lifecycle and eventually reach the running state.

The line aT4_1.run(); performs a similar sequence of operations as the previous one but does not initiate a new thread. Finally, the last line, aT4_2.start(); launches a new thread and runs the code within the run method after assigning the priority.

first --->

fourth --->

first <---

second --->

fourth <---

third --->

second <---

third <---

However, changing "run" to "start" could potentially make it possible to attain this desired output.