Which Thread?

Software Engineering Concepts (COMP3003/6007)

Addendum to part 3, Multithreading Design Structures.

When dealing with a GUI thread, and one or more other threads (which may have their own specific purposes), it's vitally important to know:

1. Which thread you're in "now". That is, when you look at a particular piece of code, you need to understand which thread(s) is (are) responsible for executing it.
2. When to perform actions in other threads. There are certain tasks that must be done in certain threads, and if you're not "in" that thread already, you need to do some thread communication.

1 Decision Tree

Here's how to decide whether you can perform a task *right here* in the current thread, or whether you need to send it to some other thread.

First, *are you in the GUI thread?*

- Yes (GUI thread). In particular, if the code is part of a GUI event handler (e.g., a lambda that you passed to `button.setOnAction()` or similar), then this is the GUI thread.

  What do you want to do?

  - GUI-related task (getting user input or changing something on-screen) → **Go right ahead!** You're in the right thread.
  - Other simple task – non-blocking, and not using any thread-confined resources → **Go right ahead!**
  - Task that blocks and/or requires a thread-confined resource → **Do it in another thread.**

- No (non-GUI thread). In particular, if you created the thread yourself (with `new Thread`, or indirectly via an `ExecutorService`) then this *is not* the GUI thread.

What do you want to do?

- GUI-related task → **Submit the task to the GUI thread**. Use `Platform.runLater()` for JavaFX (or `SwingUtilities.invokeLater()` for Swing, or an equivalent call for other GUI frameworks).
- Other simple task – non-blocking, and not using any thread-confined resources → **Go right ahead!**
- Task that blocks and/or requires a thread-confined resource.

  Is this thread intended to run this kind of task? (Is it okay to be blocked? Is it the one thread allowed to access a given thread-confined resource?)

  - Yes → **Go right ahead!**
  - No → **Do it in another thread.**

2 Doing Things in Other Threads

If you want to do *X*, but in another thread, you have the following general options:

- Start a new thread and do *X* in it. (Note that starting new threads has certain resource/performance overheads. If you have an existing one that can be re-used, that's probably more efficient.)
- Submit *X* as a task to an executor (thread pool).
- Have another existing thread that continually reads from a `BlockingQueue<Runnable>`, and put *X* into that queue.
- Have another existing thread that continually reads from a `BlockingQueue<SomethingElse>`, and put whatever object is necessary into the queue to cause the thread to perform *X*.

The last two could also be accomplished with just a monitor, if you wanted to.

> **Note**
>
> `Platform.runLater()` is for *non-GUI threads* to submit *GUI-related tasks* to the GUI thread. It is not useful for anything else.