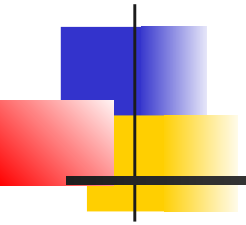
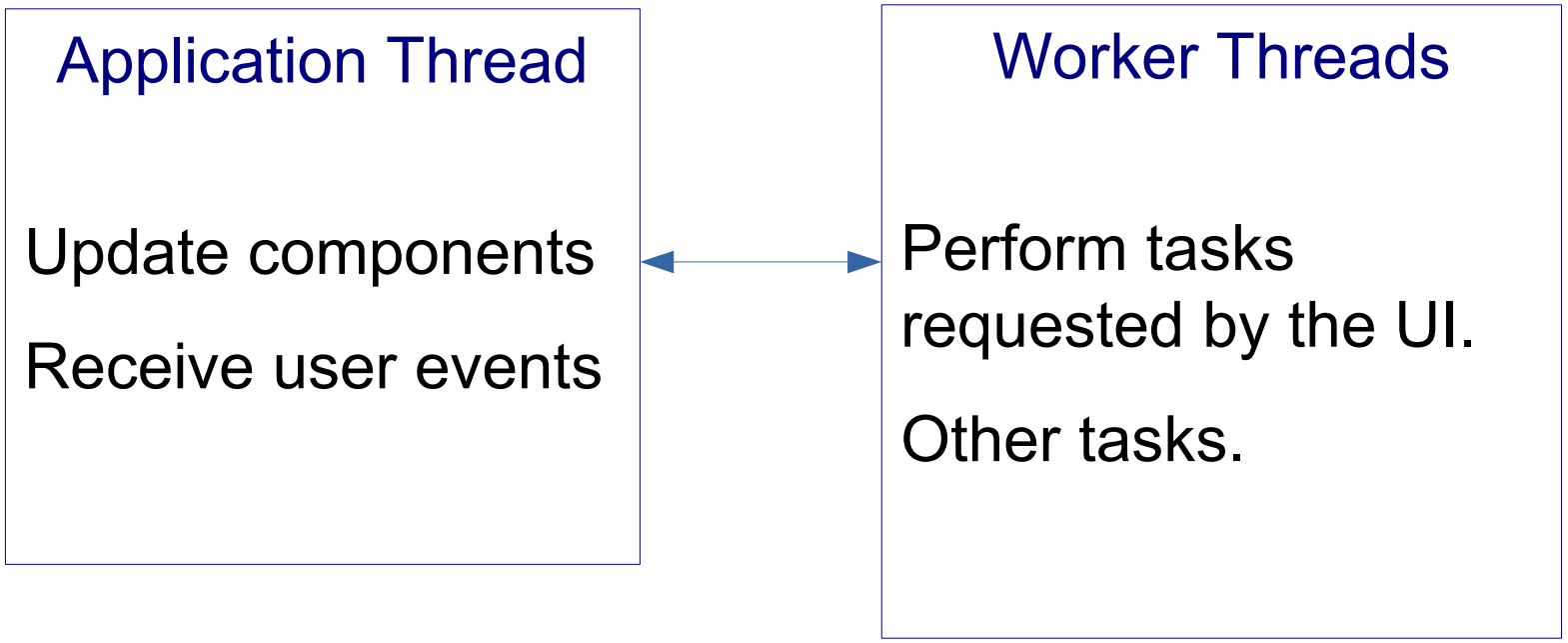


Running Code on the JavaFX Application Thread

U.I. operations and threads

JavaFX uses separate threads for managing the U.I and background tasks.

The reason is so that the U.I. is always "responsive", even if the app is working on something.



Application Thread

All UI operations must be done on the Application thread, and no other thread..

For example, if you try to show a stage on an ordinary thread like this:

```
Stage mystage = new ClickViewer( counter );
mystage.show();
```

JavaFX will throw IllegalStateException:

This operation is permitted on the event thread only

Starting a task on Application Thread

To solve this, you must start your code on the JavaFX Application thread, like this:

```
Platform.runLater( Runnable task );
```

runLater runs the task on the JavaFX Application thread.

? How do we create a *Runnable* for our ClickViewer ?

Anonymous Class for Runnable

The code we want to run:

```
Counter counter = new Counter();
Stage mystage = new ClickViewer(counter);
mystage.show();
```

Anonymous class & Platform.runLater(*Runnable*):

```
Counter counter = new Counter();
Platform.runLater( new Runnable() {
   public void run() {
      Stage mystage = new ClickViewer(counter);
      mystage.show();
   }
} );
```

Lambda for Runnable

The code we want to run:

```
Counter counter = new Counter();
Stage mystage = new ClickViwer(counter);
mystage.show();
```

using a Lambda for Platform.runLater(*Runnable*):

```
Counter counter = new Counter();
Platform.runLater(
    () -> {
        Stage mystage = new ClickViewer(counter);
        mystage.show();
    }
);
```

Summary

Graphics Frameworks use threads to separate work done on the user interface from other tasks.

This is to ensure the U.I. is responsive.

Programmer needs to start tasks on the correct thread.

There is a lab showing how to start a background task on a "worker thread".