THREADS

1) Write a program to print "Good Morning" and "Welcome" continuously on the screen in Java using threads.

```
Public Class Thread1 extends Thread {
    Public void run() {
        while (true) {
            Println("Welcome");
        }
    }
}

Public Class Thread2 extends Thread {
    Public void run() {
        while (true) {
            Println("Good Morning");
        }
    }
}

Public Class Exercise1 {
    main() {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();
        t1.start();
        t2.start();
    }
}
```

FIRST WAY

SECOND WAY

```
Class MyRunnable implements Runnable {
    Public void run() {
        while (true) {
            Print(Thread.currentThread().getName() +
                "Hello");
        }
    }
}

Public Exercise1 {
    main() {
        MyRunnable myRunnable1 = new MyRunnable();
        MyRunnable myRunnable2 = new MyRunnable();

        Thread t1 = new Thread(myRunnable1);
        Thread t2 = new Thread(myRunnable2);

        t1.start();
        t2.start();
    }
}
```

→ Runnable Interface

- functional interface that represents a task that can be executed concurrently by a thread

→ Platform.runLater()

- allows you to schedule a 'Runnable' to be executed asynchronously on the JavaFX application thread
- When you call it, the method returns immediately, allowing the calling thread to continue its execution

→ Non-daemon Threads

- threads that do not depend on the existence of other threads
- Lifecycle: they continue to run until one of the following conditions is met
  1) The run() method of the thread completes normally
  2) An uncaught exception occurs in the thread
  3) System.exit() method is called
  4) All non-daemon threads have terminated
- non-daemon by default
- main thread

→ Daemon Threads

- Considered to be background threads that provide support to non-daemon thread
- terminated automatically when all non-daemon threads have completed