Java Concurrency (Multithreads)

Goals

- Learn how to:
  - Create and start Java Threads
  - Work with SwingWorker

1. Creating and Starting Threads

- Create a subclass of Thread and override the run() method. The run() method is what is executed by the thread after you call start().

```
public class MyThread extends Thread {
    public void run(){
        System.out.println("MyThread running");
    }
}
```

- To create and start the above thread you can do like this:

```
MyThread myThread = new MyThread ("ThreadOne");
myThread.start();
```

1. Creating and Starting Threads

- Create a class that implements java.lang.Runnable. The Runnable object can be executed by a Thread.

```
public class MyRunnable implements Runnable {
    public void run(){
    System.out.println("MyRunnable running");
    }
 }
```

- To create and start the above object you can do like this:

```
Thread myThread = new Thread (new MyRunnable("ThreadTwo"));
myThread.start();
```

If you call "myThread.run()" no thread will be created

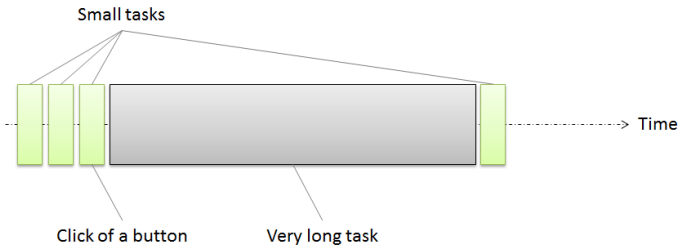
You can get the thread name by calling "Thread.currentThread().getName();"

2. Thread in Java Swing

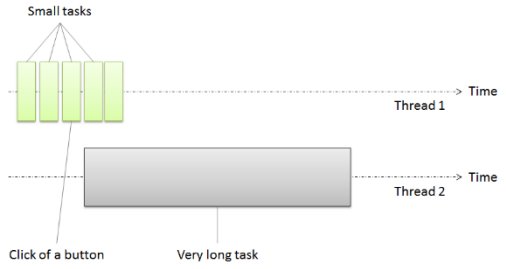
- A Swing programmer deals with the following kinds of threads:
  - Initial threads, the threads that execute initial application code.
  - The event dispatch thread, where all event-handling code is executed. Most code that interacts with the Swing framework must also execute on this thread.
  - Worker threads, also known as background threads, where time-consuming background tasks are executed.

3. Worker Threads

- With single threaded applications, the user clicks the button that starts the process and then has to wait for the task to finish before the user can do something else with the application.



- Multithreading address this problem. The application handles small tasks, such as button clicks, by one thread and the long taking tasks by another thread.



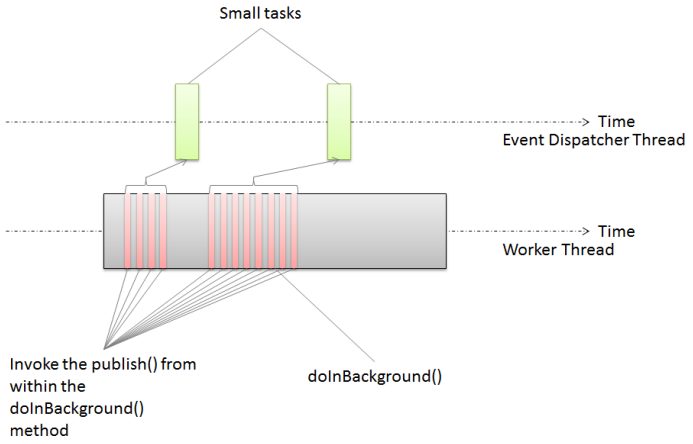
3. Worker Threads
Common methods

Method	Description
doInBackground	Defines a long computation and is called in a worker thread.
done	Executes on the event dispatch thread when **doInBackground** returns.
execute	Schedules the SwingWorker object to be executed in a worker thread.
get	Waits for the computation to complete, then returns the result of the computation (i.e., the return value of **doInBackground**).
publish	Sends intermediate results from the **doInBackground** method to the process method for processing on the EDT.
process	Receives intermediate results from the **publish** method and processes these results on the EDT.
setProgress	Sets the progress property to notify any property change listeners on the EDT of progress bar updates.

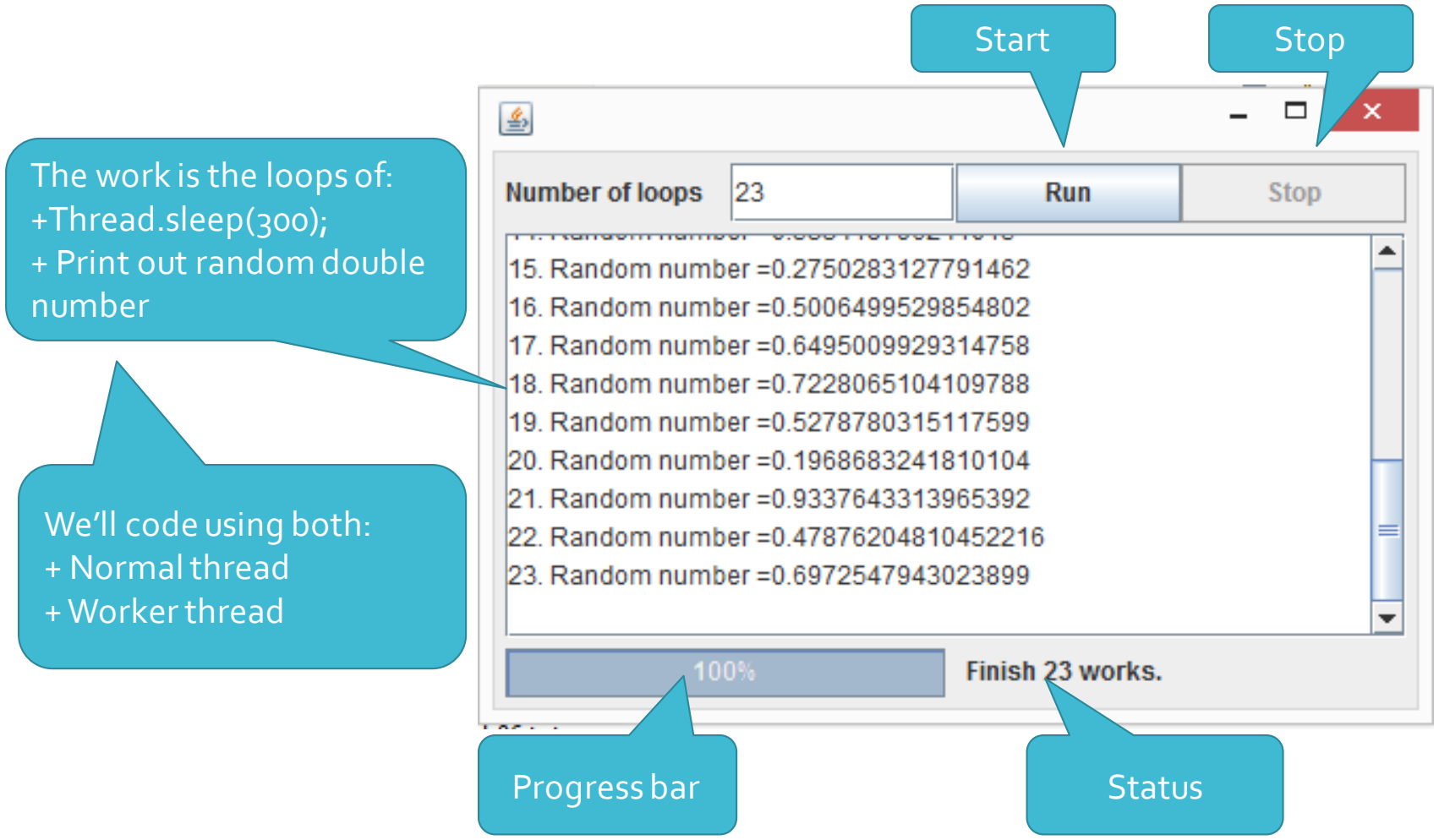
3. Implements SwingWorker SwingWorker class template

```
public class WorkerClass extends SwingWorker<Integer, String> {


// constructor

public WorkerClass() {

}


// main process

protected Integer doInBackground() throws Exception {

     //...

     setProgress(0..100);

     publish("Update content here");

     // ...
}


// displays published values

protected void process(List<String> publishedVals) {}


// code to execute when doInBackground completes

protected void done() {

     //...

     int retNum = (int)get();

     //...

   }

}
```

(1) Type of return value in background process
(2) Type of observation value

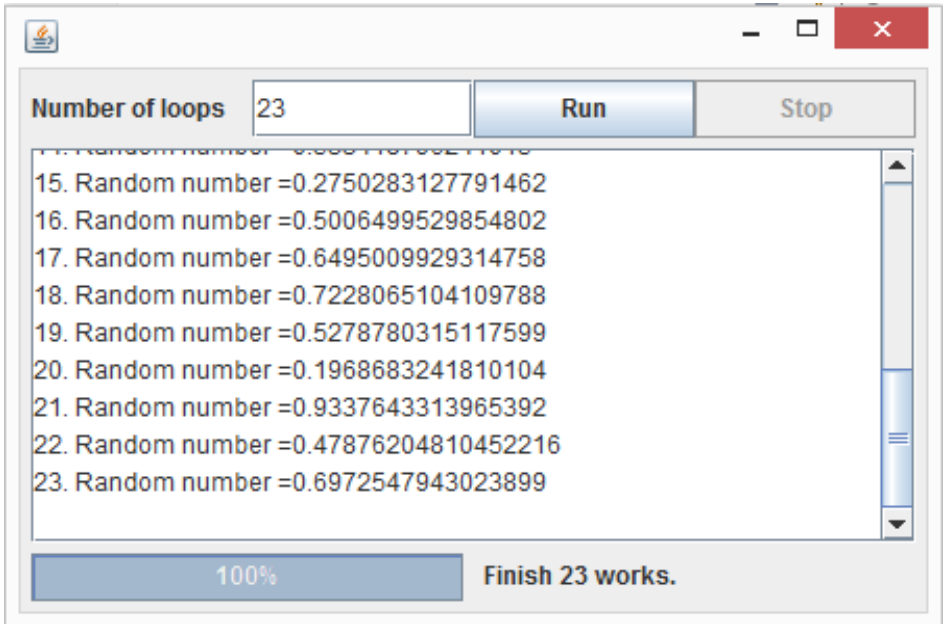


PRACTICE


Start
Stop
The work is the loops of:
+Thread.sleep(300);
+ Print out random double number
We'll code using both:
+ Normal thread
+ Worker thread
Number of loops
23
Run
Stop
15. Random number =0.2750283127791462
16. Random number =0.5006499529854802
17. Random number =0.6495009929314758
18. Random number =0.7228065104109788
19. Random number =0.5278780315117599
20. Random number =0.1968683241810104
21. Random number =0.9337643313965392
22. Random number =0.47876204810452216
23. Random number =0.6972547943023899
100%
Finish 23 works.
Progress bar
Status

2. Design - MVC pattern

- Model:
  - Thread/SwingWorker
- View:
  - JTextArea txtArea.
  - JProgressBar progressBar
  - JLabel lblStatus
- Controller:
  - private JTextField textField;
  - JButton btnRun
  - JButton btnStop



3. Implements - GUI
Create the components