Multithreaded Programming

1. Make a coin-flipping class that implements `Runnable`. The `run` method should flip 1000 coins and print out whenever they get 3 or more consecutive heads. Make a task queue, and put 5 separate instances of the `Runnable` class in the queue. In the printouts, you can use the `Thread.currentThread().getName()` to identify the thread. You are following variation 1 of the basic threading approach (separate classes that implement Runnable), so your code will look something like this (or, you should call `execute` from a loop):

```
public class Foo implements Runnable {
  public void run() { loop, flip coins, check for 3+ heads in a row }
}
-----------------------------------------------------
public class Driver {
  public static void main(String[] args) {
    ...
    tasks.execute(new Foo()); // Multiple instances of Foo
    tasks.execute(new Foo());
    tasks.execute(new Foo());
  }
}
```

2. Do a similar task, but this time make only one instance of your main class (the one that implements `Runnable`). Still have 5 tasks in the queue. You are following variation 2 of the basic threading approach (main class implementing Runnable). Now your code will look roughly like this (or, with the calls to `execute` in a loop):

```
public class Foo implements Runnable {
  public Foo() {
    ...
    tasks.execute(this);
    tasks.execute(this);
    tasks.execute(this);
  }
  public void run() { loop, flip coins, check for 3+ heads in a row }
}
-------------------------------------------------------
public class Driver {
  public static void main(String[] args) {
    new Foo(); // One instance of Foo, not multiple
  }
}
```

3. Pop up a `Frame` or `JFrame` (or use an applet). Change the layout manager to `GridLayout` with 5 rows and 1 column. Create 5 coin-flipping tasks and associate each with a `Label` or `JLabel`. Which approach (separate class, interface, inner class) should you use for the code that has the “run” method? Have each task flip 1000 coins and print the number of heads in the label. Hints:

• Use setText to put text in the label.

• Remember that String.format is the Java equivalent of C’s sprintf.