Concurrent Programming with Threads (Page 1 of 2)

1. Make a coin-flipping class that implements `Runnable`. The `run` method should flip 1000 coins and print out whenever it sees 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 `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 could 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) {
    ExecutorService tasks = ...
    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() {
    ExecutorService tasks = ...
    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
  }
}
```

Concurrent Programming with Threads (Page 2 of 2)

3. Here is some code that will put 5 JLabel objects (components that display strings) in a JFrame.

```
public class StaticJLabels extends JFrame {
  public StaticJLabels() {
    super("Flipping Coins");
    Container contentPane = getContentPane();
    contentPane.setLayout(new GridLayout(5, 1));
    for(int i=0; i<5; i++) {
      JLabel label = new JLabel("Label " + i);
      label.setFont(new Font("SansSerif", Font.PLAIN, 60));
      contentPane.add(label);
    }
    setSize(400, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
  }

  public static void main(String[] args) {
    new StaticJLabels();
  }
}
```

Copy this class from my multithreaded-programming-exercises project, then modify it so that each JLabel counts coin flips. Create 5 coin-flipping tasks and associate each with a JLabel. Have each task flip 1000 coins and print the number of heads in the label, updating the count as each head is found. Which approach (separate class, interface, inner class) should you use for the code that has the “run” method?

Hints:

- Use setText to change the text in the label.
- Remember that String.format is like printf, but it outputs the formatted string instead of printing the formatted string.