Practical session 13

This work should be completed before the next lecture.

Task 1: Prepare to use threads

Create a NetBeans project for this task.

Write a class `Message` that contains the following method, which outputs a string of text (`msg`) a number of times (`num`):

```
    public void run() {

        for (int i = 0; i < num; ++i) {

            System.out.println(msg + " " + i);

        }

    }
```

The string to output (`msg`) and the number of times (`num`) are instance variables (attributes) that must be initialised in the `Message` constructor using parameters values.

Write a class with a `main()` method that instantiates the `Message` class several times with different messages and different numbers of times, and calls the `run()` method for each object. Do NOT use threads at this stage.

Portfolio requirements:

- The NetBeans project for this completed task

Task 2: Using threads

Copy your NetBeans project from Task 1.

Modify your `Message` class to support threads. The `run()` method should execute when the thread starts. Modify your `main()` method so that each instance of the `Message` class is run on a separate thread. Is there any difference in output from the testing in Task 1? Why?

Now call `Thread.sleep(100)` to the loop within the `run()` method to make the loop take longer than a single time slice. Repeat the testing. What is the difference in output? (You may need to increase the sleep time gradually until you see a change in the output.) Can you explain what is happening?

Now modify the `Thread.sleep()` method call so that the time to sleep is chosen at random. Repeat the testing. What is the difference in output? Can you explain what is happening?

Portfolio requirements:

- The NetBeans project for this completed task

Task 3: Producer-Consumer

Download from Blackboard the `Lecture13Demo` NetBeans project.

Add two more classes, `Producer` and `Consumer`, which extend `Thread`. The constructor for each class should take a reference to a `Buffer` object.

The `Producer`'s `run()` method should have a loop to add the numbers 1 to 15 to the `Buffer` object. After each number is added, its thread should sleep for a short interval.

The `Consumer`'s `run()` method should have a loop to retrieve the next number from the `Buffer` object, then sleep for a short interval.

Add another class, `ProducerConsumerMain`, containing the following main method:

```
public static void main(String[] args) {
    Buffer b = new Buffer(2);
    Producer p1 = new Producer(b, 1);
    Producer p2 = new Producer(b, 2);
    Consumer c1 = new Consumer(b, 1);
    Consumer c2 = new Consumer(b, 2);
    p1.start();
    p2.start();
    c1.start();
    c2.start();
}
```

Without changing any of the code in the `Buffer` or `ProducerConsumerMain` classes, implement your `Producer` and `Consumer` classes so that your program has output similar to the following (each run will be different because of the random sleep intervals):

```
Producer 1 added 1 to buffer:[1]
Consumer 2 retrieved 1 from buffer: []
Consumer 1 attempting to remove from empty buffer - wait
Producer 2 added 1 to buffer:[1]
Consumer 1 retrieved 1 from buffer: []
Producer 2 added 2 to buffer:[2]
Producer 1 added 2 to buffer:[2, 2]
Producer 1 attempting to add to full buffer - wait
Producer 2 attempting to add to full buffer - wait
Consumer 1 retrieved 2 from buffer: [2]
...
```

Portfolio requirements:

- The NetBeans project for this completed task