CE00527-5 Further Object Oriented Programming
Tutorial 9

In this tutorial you will explore using threads in Java programming

Part 1 – Using Threads - Basic Exercise

1 Develop 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) and should be initialised using parameters to the Message constructor. Test your class by instantiating several instances of it with different messages and different numbers of times and invoking the run() method against each instance. Do NOT use threads for this question.

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

3. Now add a sleep in 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?

4 Now modify the sleep 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?

Part 2 Producer-Consumer example – Advanced Exercise

In the lecture you were given code for a thread-safe buffer which allows communication between threads. Make a new Netbeans project and add this Buffer class to it.

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

Producer's run method should have a loop which adds the numbers from 1 to 5 to the Buffer object. After each number is added, its thread should sleep for a short interval.

Consumer's run method should have a loop that runs 5 times. Within the loop, it should 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 Main 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]
……
```

You should put in your portfolio

- **code listing, evidence of testing and answers to questions for all exercises in Part 1**
- **the design (such as a flow chart, structure diagram, state chart or activity diagram) of the logic for your game in Part 2**
- **the code listing for any new classes you created to implement the game in Part 2, plus any code you added to the existing classes**
- **evidence (including screen shots if applicable) of testing your game**
- **code listing for your Producer and Consumer classes**
- **sample output from running the program and evidence of testing (test code and output) for all exercises**