project – Concurrent programming with Java

Comp3071 - Programming Languages

# Objective

Understand the concepts of *(and implementation strategies for)* **Concurrent Programming.**

# Topics

* General concepts of **Concurrent Programming.**
* Implementation of concurrency in **Java.**

# reading

**Chapter 13**

# instructions

#### preliminary

*Select a partner*

#### Answer Questions (based on reading)

1. What is the best action a system can take when deadlock is detected?
2. In the producer-consumer example of Section 13.3, suppose that we incorrectly replaced the release(access) in the consumer process with wait(access). What would be the result of this error on execution of the system?
3. **Compare** the Java mechanisms for cooperation and competition synchronization with that of Ada.
4. What happens if a monitor procedure calls another procedure in the same monitor?

#### Programming

1. **Select a scenario from the table below (or propose your own) that you would like to implement as a simulation in Java using threads and synchronization to control access to shared resource(s):**

|  |  |
| --- | --- |
| *Title* | *Description* |
| *Bike Shop Simulation* | A bike shop has customers continuously, concurrently renting bikes (a shared, limited resource) for a variable (random) amount of time. When there are no bikes available customers, unfortunately, must wait until one is returned. |
| Fish Counter | Customers continuously, concurrently arrive and take a number. When one of the employees is free they call the next number. Customers require a variable (random) amount of service time. |
| Doctors Office | Patients continuously arrive for appointments with a doctor. When the doctor is seeing other patients a new arrival must wait. Patients require a variable (random) amount of service time. When there are no patients the doctor must wait. |
| CPU simulation | Simulate the behavior of a single processor CPU where processes are continuously created and enter with a certain amount of required running time. Each process gets the CPU for at most a time slice; if it hasn’t completed and then it returns to a ready queue. When a process is running other processes must wait. Some processes randomly need to wait on input. |

1. **Design your solution using UML before you start programming:** 
   1. What are your classes, instance variables, methods?
   2. What classes will extend threads?
   3. How do you synchronize the shared resource(s)?
   4. What are the inputs from the user and/or from an initial configuration file? (well-designed code should not contain hard coded values)
   5. What values should be displayed to illustrate the behavior of your multi-threaded solution?
   6. What type of interface (GUI, console)?
   7. What would sample dialogs look like?
2. **Implement your program** (if the design is complete and correct this step should be easy).
3. **Test your solution using a variety of different test cases**
4. **Present your program** 
   1. *Describe your UML design*
   2. *Explain, in general, how you manage multiple threads and synchronize resources (please do not step through your code line by line).*
   3. *Demo several of your test cases.*
   4. *Aim for 5-8 minutes.*

# Evaluation

|  |  |  |
| --- | --- | --- |
| *Requirement* | *Percent* | *Notes* |
| *Design* | *20* | *Object-Oriented, Elegant, Complete* |
| *User Interface* | *10* | *Whether GUI or console based, it should be able to demonstrate the behavior of your multithreaded solution.* |
| *Threads & synchronization* | *40* | *Successfully create and manage multiple threads and synchronize the use of shared resource(s)* |
| *Simulation* | *20* | *Correct and complete according to the scenario described* |
| *Presentation* | *10* |  |

# Submitting your work

1. Make sure that your name is in ***all***of your project files.
2. If you have **more than one file** for your solution, make a **.zip file** for your project
3. In **Blackboard**, attach your solution file to the submission for this assignment.