UNIVERSITY of BRADFORD

Concurrent and Distributed Systems

COS6012-B

Coursework 1

Weighting: 50% of final module mark

Due date: 6 March 2026, 3pm

Department of Computer Science

University of Bradford

Learning Outcomes Assessed

- **LO1**: Demonstrate knowledge and understanding of fundamental concurrency concepts and methods.
- **LO2**: Demonstrate knowledge and practical skills in using concurrent programming tools and libraries for building concurrent systems.

Submission Requirements

Submit via Canvas:

1. A PDF report.
2. A ZIP containing a Java project.
3. A video demonstration of up to 5 minutes, in MP4 or similar format. You may use any screen-recording tool to create the video, for example, Microsoft Teams.

Academic Integrity

Referencing and plagiarism:
`https://www.bradford.ac.uk/library/help/referencing-and-plagiarism/`

Coursework Description

Theoretical and practical aspects in the construction of concurrent systems (Implementation and report 2500 words).

Scenario

Betty's Café is introducing a special buffet experience for Mother's Day. Customers can book a two-hour session during which they can enjoy a variety of cakes and speciality teas and coffees, and alternate periods of eating and drinking with playing the piano and listening to music. All sweets and hot drinks are prepared by the café staff and made available in a shared buffet area. Because the café is extremely busy, items may temporarily become unavailable. For example, a customer may wish to have tea and cake, but cake may not be available at that moment. In such cases, the customer must wait until both desired items are available before taking them to their table. Meanwhile, the café staff continue to prepare new items and replenish the buffet. Customers may also play the piano; however, a maximum of two people can play simultaneously.

Your task is to implement a concurrent system that models this scenario., assigning a random execution time to each action performed by customers and staff. Your implementation must produce verbose output and must fully adhere to the requirements described in the following sections.

Question 1 (10 marks): Interference

Explain the concept of interference in concurrent systems and provide an example of how it could occur within the context of this scenario.

Criterion	**Marks**
Clear explanation of interference in concurrent systems	5
Scenario-specific example showing how interference could occur at Betty's Café	5

Question 2 (10 marks): Deadlock

Describe what a deadlock is, explaining its causes and the strategies used to avoid it.

Criterion	**Marks**
Accurate definition of deadlock	3
Explanation of its causes	3
Appropriate strategies to avoid it	4

Question 3 (10 marks): Deadlock in this scenario

Explain how a faulty implementation of this scenario could lead to a deadlock, providing a concrete example. Then propose a suitable strategy to prevent it.

Criterion	**Marks**
Concrete faulty-implementation example demonstrating how deadlock arises	5
Correct and feasible strategy to avoid it	5

Question 4 (30 marks): System design and concurrency guarantees

Describe how you would design your solution for this scenario. Identify the threads and the main classes that you would define, and explain which concurrency concepts and Java synchronisation mechanisms you would use, and why. Your design must:

- Ensure mutual exclusion when accessing shared resources (10 marks)
- Be free from deadlock and livelock (10 marks)
- Incorporate fairness (10 marks)

Provide detailed justification for how your design satisfies each of these requirements.

Criterion (mutual exclusion)	**Marks**
Correct identification of shared resources	3
Appropriate use of Java concurrency primitives	3
Clear justification of how mutual exclusion is guaranteed	4

Criterion (deadlock-free, livelock-free)	**Marks**
Explanation of deadlock-free design choice	5
Explanation of livelock avoidance	5

Criterion (fairness)	**Marks**
Explanation of fairness mechanisms	5
Appropriate use of Java concurrency primitives	5

Question 5 (30 marks): Java implementation

Implement your solution in Java and simulate the system with an arbitrary number of customers (N > 5) and a small number of staff. For example, you may include one staff member preparing tea (bringing a small random number of freshly brewed cups to the buffet), one bringing cakes (a few at a time) and one preparing coffee (also adding a small random number of cups). Each customer should be allowed to serve themselves with: (a) coffee only; (b) tea only; (c) tea and cake; (d) coffee and cake; (e) cake only; or engage in another activity: (f) listening to music, or (g) playing the piano. These activities may alternate and repeat throughout the simulation.

Implement a GUI to configure and display the concurrent execution of the system. The GUI should be flexible and allow users to:

- Specify the initial number of customers
- Specify the number of staff
- Set the initial quantities of cakes, tea and coffee
- Provide an option to add new customers dynamically during execution
- Allow the simulation speed to be adjusted (slow/fast), by decreasing or increasing the random timing parameters used in the simulation

See below for an example of simulation output.

Criterion	**Marks**
Accurate implementation of the scenario logic, reflecting the proposed design	10
GUI allows configuration of customers, staff, initial stock and speed control	10
Representative example of simulation output	10

Question 6 (10 marks): Video demonstration

Provide a video demonstrating a running simulation of your system, explaining the functionality of the GUI and the generated output. Briefly present the key parts of your source code.

Criterion	**Marks**
Clear demonstration of the running simulation, explaining GUI functionality and output	5
Brief presentation of key source code elements	5

Example of simulation output

You are not required to follow this exact format, but your output should include sufficient detail to make the simulation easy to understand. Replace `<random time>` with an actual time (in seconds or milliseconds), based on the values generated by your methods. Feel free to include additional information, such as the presence of any waiting queues (and which customers are in them).

```
Starting program with 6 clients and 3 staff
Buffet = (2 cakes, 2 teas, 2 coffees)

Client-1 wants to play the piano.
Client-1 plays the piano for  <random time>.
Client-2 wants a cake.
Client-2 takes a cake and eats for  <random time>.
 Buffet = (1 cake, 2 teas, 2 coffees).
Client-3 listens to music for <random time>.
Client-5 wants coffee and cake.
Client-4 wants tea and cake.
Client-5 takes coffee and cake and eats for  <random time>.
 Buffet = (0 cakes, 2 teas, 1 coffee).
Client-4 waits.
Client-6 wants to play the piano.
Client-6 plays the piano for  <random time>.
Client-2 finished eating.
Client-2 wants a tea.
Client-2 takes a tea and drinks for  <random time>.
 Buffet = (0 cakes, 1 tea, 1 coffee).
Client-3 finished listening music.
Client-3 wants a coffee.
 Buffet = (0 cakes, 1 tea, 0 coffee).
Staff-1 brings 3 teas.
 Buffet = (0 cakes, 4 teas, 0 coffee).
Staff-1 returns to kitchen.
Client-1 finished playing the piano.
Client-1 wants a tea.
Staff-2 brings 2 cakes.
 Buffet = (2 cakes, 4 teas, 0 coffee).
Staff-2 returns to kitchen.
Client-4 takes tea and cake and eats for  <random time>.
 Buffet = (1 cake, 3 teas, 0 coffee).
Client-1 takes a tea and drinks for <time>.
 Buffet = (1 cake, 2 teas, 0 coffee).
Staff-3 brings 3 coffees.
 Buffet = (1 cake, 2 teas, 3 coffees).
Staff-3 returns to kitchen.
...
```