COMPX234-25A
Assignment 1

Total Marks: 10:
Due: 31th March 2025, 11.59 PM

Important

There is zero tolerance for plagiarism.
Credit all sources you refer to.
Students found plagiarising will be reported to the disciplinary committee.
You are expected to follow the University's guidelines here:
https://www.waikato.ac.nz/students/academic-integrity/student-information/plagiarism.
Assignments will be checked against anti-plagiarism checkers.

---

1. The Problem

The problem in the assignment is an example of a Producer-Consumer problem. In our system we have several machines (M) that occasionally need to print some text. We also have a small number of printers (N) that can print text given to them. We however have more machines wanting to print than printers available to us (M > N).

To enable printing, our system uses a print queue. When a machine has something to print, it creates a print request for the document to be printed, which gets inserted in the print queue. When a printer is free, it retrieves the print document at the head of the queue, prints the required text and removes that item from the queue.

Since resources in any given system are finite, the queues have limits and so is the case in our system. The size of our queue is also limited and is equal to the number of printers in our system. This presents a challenge. If the queue is already full and a machine creates a new print request, by default the new request will overwrite the request at the tail of the queue. This overwriting is undesirable as it means information is lost.

2. Starter Code

You have been given four files, `Assignment1Task`, `printDoc`, `PrintList` and `Main` in both python and Java. `printDoc` and `printList` create the data structure to be used as the queue. `printList` also makes `queueInsert` and `queuePrint` methods available that can be used to insert objects in the queue and print objects from the queue. You do not need to modify these files. `Main` is the wrapper around Assignment1 class and is used to start the entire program. You do not need to modify this file either. For this assignment you will work with the `Assignment1Task` file. The file has functions and

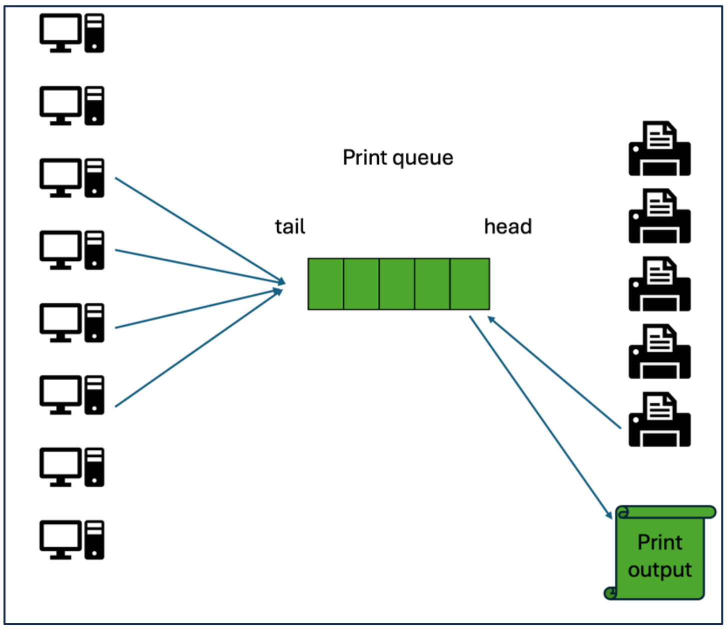


Figure 1: visual representation

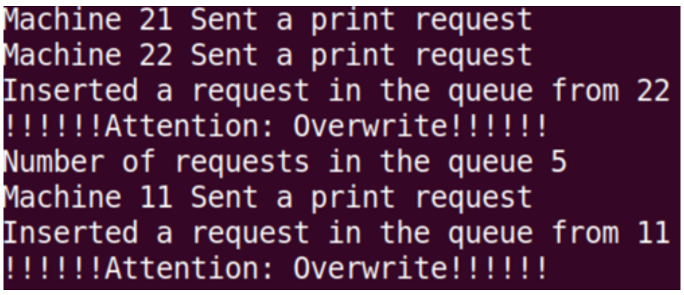
comments that will be helpful in writing code. If you choose to use the python files, you will need to rename `Assignment1Task` to `Assignment1`.

3. Tasks

Task 1: Your first task is to use the starter code given in `Assignment1Task` and implement the following bevahiour for the machines and printers. Implement the machines and printers as threads in Java and processes in Python.

1. The machines sleep for some time and then wake up and send a print request.
2. Printers also sleep for some time. When they wake up they print the document at the head of the queue.

The functions needed to implement the above functionality are available in the starter code. Once this functionality has been created and `Assignment1Task` is executed using `Main`, you will see output that looks like Figure 2. The output will display machines sending print requests and printers printing those requests. When a machine sends a print request to a queue that is already full, you will see an `` `Attention: Overwrite' `` message.



```
Machine 21 Sent a print request
Machine 22 Sent a print request
Inserted a request in the queue from 22
!!!!!!Attention: Overwrite!!!!!!
Number of requests in the queue 5
Machine 11 Sent a print request
Inserted a request in the queue from 11
!!!!!!Attention: Overwrite!!!!!!
```

Figure 2: Sample output after Task 1

Task 2: Your second task is to design and implement a solution using the synchronization solutions discussed in the lectures to resolve the problem of overwriting. Your solution will,

1. Control the machines' access to the print queue such that no overwriting takes place. We consider that the size of our queue is 5. If there are 5 messages in the queue already, then no machine should be allowed to send a print request, until a printer prints one of the messages and makes space available.
2. Control the machines' and printers' access to the print queue such that no two devices (machines and printers) are accessing the queue at the same time.

Assignment submission

1. You will create a repository by the name COMPX234-A1 on the Waikato gitlab server.
2. This will be your main assignment repository and the one that will be marked. No submission is required on moodle.
3. You will need to regularly commit code updates to this repo as you work on the assignment. This will create a verifiable history of commits and will show how the code evolved.
4. The marking will only consider commits done prior to the deadline. The markers will also use your commit history to mark your code, not just the final output.
5. You may also be required to demonstrate and explain your code.

Extensions

- Extension requests should be made at least two days before the due date using the extension form on moodle.
