Lancaster University

SCC.211 Operating Systems

Overview

Dr Amit Chopra
School of Computing & Communications, Lancaster University, UK
amit.chopra@lancaster.ac.uk

Course Aims

Lancaster University

Theory and practical application of operating system concepts and concurrent systems

General understanding of issues in writing concurrent systems

Understand the role of operating systems play in computing

Experience in designing and implementing complex data structures to meet resource and operational system constraints

How This Course is Taught

Lancaster University

Week 1 – 5

Dr Amit Chopra

Overview of approaches for concurrent programming

Week 6-10

Dr Andrew Scott

Role and functionality of the Operating System (OS)

Main Modes of Delivery

Lancaster University

Lectures

- Will be recorded + uploaded

Lab

- With academics and TAs

Overall Module Assessment

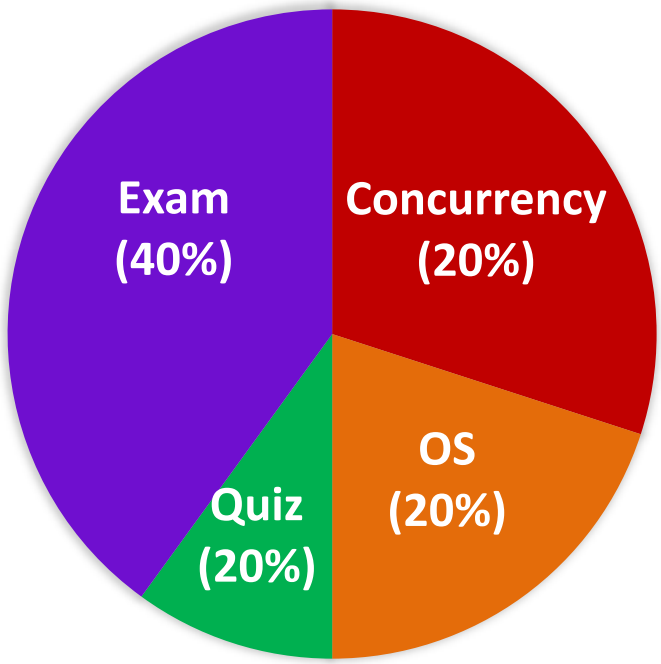
Lancaster University

Coursework

Two programming courseworks (40% total)

Four Moodle quizzes (20% total)

- Feedback in labs, in class, solutions
- More details on SCC.211 Moodle page



Coursework is submitted online and plagiarism checked automatically

Feedback and Queries

Lancaster University

Detailed/personal

- Academics and TAs in lab sessions
- On-the-spot verbal feedback
- Best chance to clarify anything you don't understand
- Email

- Solutions, with explanations where appropriate

What is Plagiarism

Lancaster University

- Submitting someone else's code
- Sharing code with each other
- Paying someone else to do it for you
- Working on assignment together & submitting individually
- If you feel like it’s cheating, it probably is

We catch people each year, please don’t be one!

What We Expect from You

Lancaster University

- Integrity and Effort
- Come to lectures
- Go to labs (they are compulsory)
- Use textbooks and Internet
- Take notes (it does help, promise)
- Try to keep up with coursework and time yourself

What You can Expect from Us

Lancaster University

- Put effort in explaining things and communicating clearly
- Lecture notes on Moodle
- Ensure labs are running smoothly and TAs are supportive
- Provide extra support if needed
- Offer prompt feedback on exercises
- Respond to email
  - Note: We get tons of email/requests

Contact

- **Use the Moodle forum**
- **Email**
  - May be delayed in replying
  - Tried/looked for solutions yourself!
  - Exhausted other contact options

Recommendations

Lancaster University

Books

- Jean Bacon, Concurrent Systems, 2002.
- Burns, A., and Davies, G., “Concurrent Programming”, Addison Wesley, 1993, ISBN 0-201-54417-2
- Operating Systems, Principles and Practice. Thomas Anderson, Michael Dahlin,
- Switzer, R., “Operating Systems: A Practical Approach”, Prentice Hall, 1993, ISBN 0-13-640152-X
- Lea, D., “Concurrent Programming in Java”, Addison Wesley, 1997, ISBN 0-201-69581-2

Papers (see Moodle for complete list)

- “Information-Driven Interaction-Oriented Programming:BSPL, the Blindingly Simple Protocol Language,” Munindar P. Singh
- “An Evaluation of Communication Protocol Languages,” Chopra et al.
- “Mandrake: A Protocol-Based Programming Model” Christie et al.

Weeks 1-5: Coursework and Quizzes

Lancaster University

- Coursework will be up on Moodle
  - Use of Java threads and synchronization
  - Must be submitted on Friday Week 3
  - In-lab marking in Weeks 4 and 5
- Two 30-minute multiple choice Moodle quizzes
  - Quiz 1 based on material from Weeks 1-2
    - Released Monday Week 3, due Friday Week 3
  - Quiz 2 based on material from Weeks 3-4
    - Released Monday Week 5, due Friday Week 5

Weeks 1-5 (slide 1 of 3)

Lancaster University

1

Introduction to concurrency
- In real life & computing
- Java threads

2

Race conditions & Locks
- The emergence of concurrency
- Atomicity, race condition, mutual exclusion

3

Programs, Processes, Processors, Patterns
- ❑ Understanding the distinctions between them
- ❑ Understanding concurrency and parallelism
- ❑ Concurrency Patterns

4

Semaphores
- A more general concurrency mechanism for managing cooperation between processes; Java wait/notify

Weeks 1-5 (slide 2 of 3)

Lancaster University

5

Classic Synchronization Problems

- Readers-Writers; Producer-Consumer using Semaphores

6

Spinlocks and their Implementaiton

Barrier Synchronization

7

Pitfalls

- Flawed mutual exclusion, unfairness, livelock and deadlock.
- Dining Philosophers

8

Shared-nothing coordination

- Protocols
- Decentralized Systems
- Addressing Pitfalls of Control Flow

Weeks 1-5 (slide 3 of 3)

Lancaster University

9

Information Protocols

- Information causality
- Integrity
- Protocol enactments

10

Selection of Advanced Topics

- Safety, Liveness

Aim Beyond the Specific Topics

Lancaster University

- Illuminate fundamental concepts
- Illuminate shared memory and non-shared memory concurrency
- Combine practice and research
- Develop skills via labs and the programming exercise
- Develop your critical thinking abilities

Lecture 1 Objectives

Lancaster University

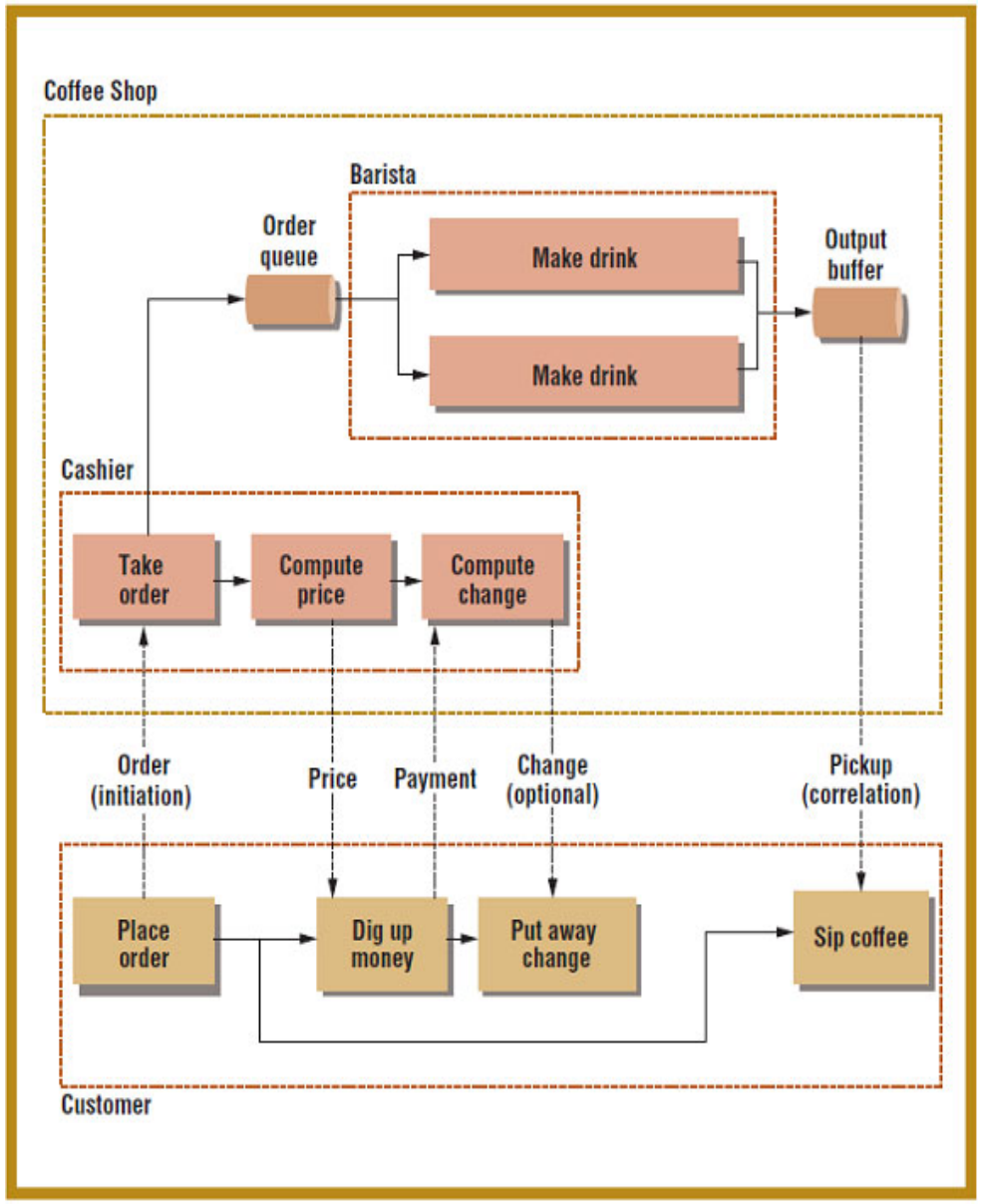
- Concurrency is every day life
- Concurrency in computing
- Java threads

Concurrency in the Real World: Coffee Shop

Lancaster University

Figure from Gregor Hohpe's excellent 2005 IEEE Software article *Your Coffee Shop Doesn't Use Two-Phase Commit*

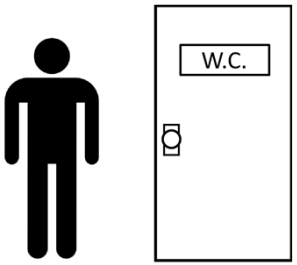
- Customer places Order with Cashier, who places the Order in a queue (as a cup with drink name on it). Customer and Cashier are free to go on to other business.
- Barista takes Orders from the queue and can make two drinks at a time. When a drink is ready, it is placed on counter for Customer to pick up.
- Concurrent because multiple actors: Customer, Cashier, Barista and the two coffee machines.
- Actors decoupled: work concurrently on several Orders at the same time! Highly efficient!
- (A synchronous alternative: Customer places for Order with Cashier and waits there till it is ready.)

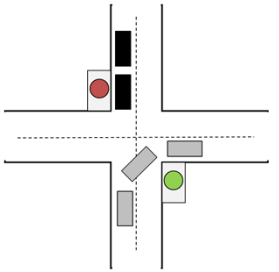


Concurrency Components

Lancaster University

	Actors	Shared Resources	Access Rules
Actors	People	People	Vehicles
Shared Resources	WC	Food in cupboard	Streets
Access Rules	Rule: If the door is locked, wait	Write a note if gone out to buy food.	Don't move if light is red/amber; From red->green, wait for all cars to first exit

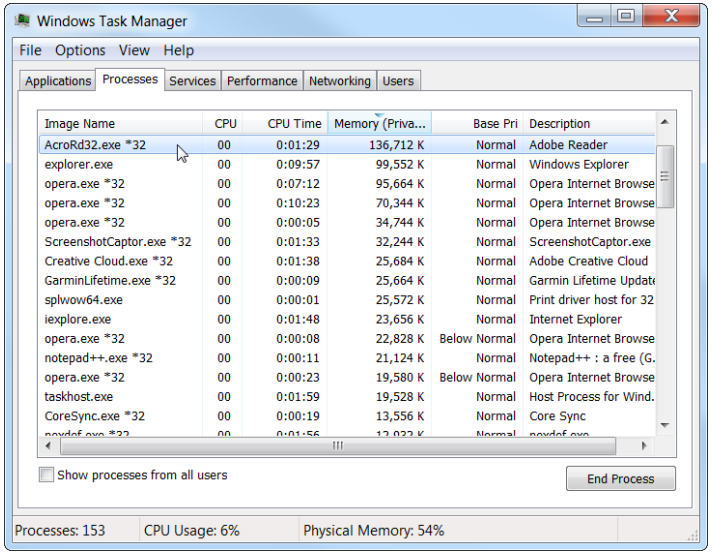


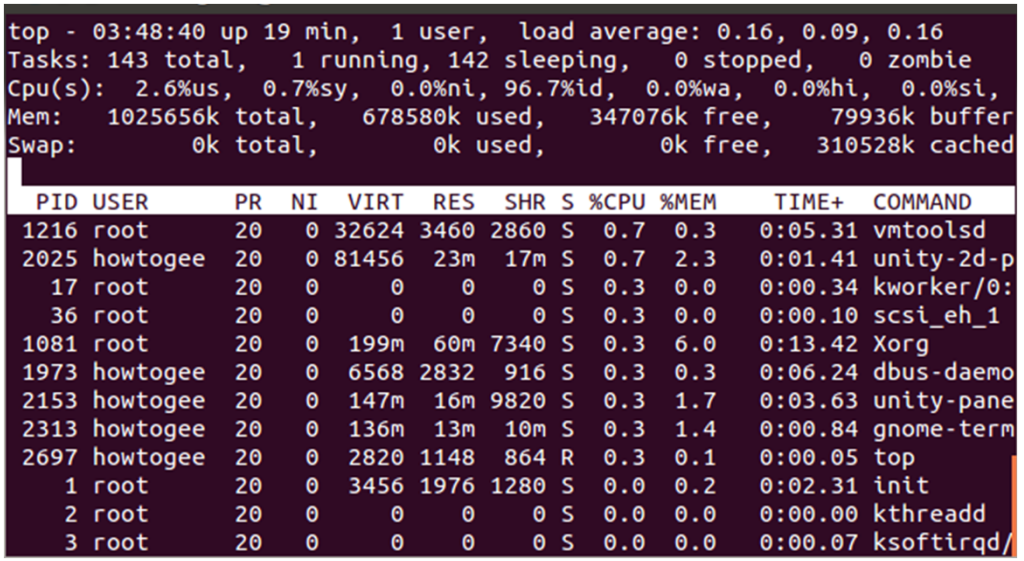


Concurrency: Phenomenon

Lancaster University

Multiple, independent loci of computation (actors, processes, threads) interacting over shared resources.



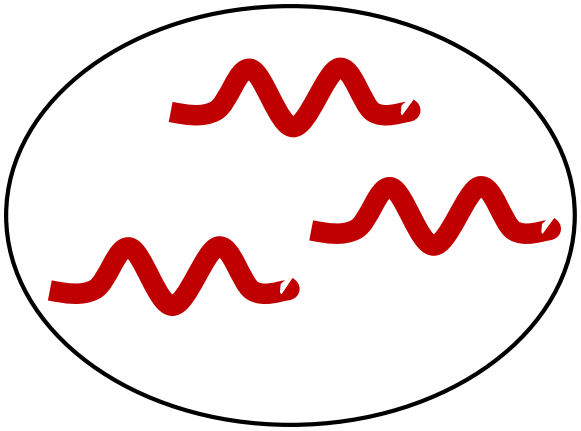


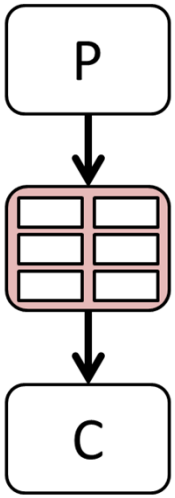
Uninteresting: *Alice and Bob have separate food stores.*

Interesting: *They have a shared food store.*

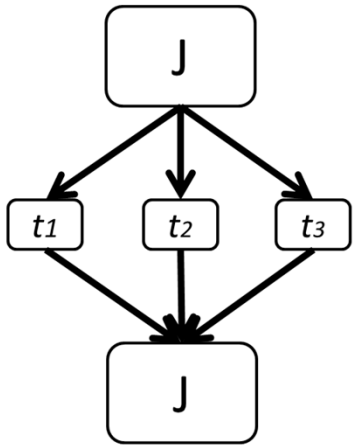
In Computer Systems…

- **Multiple processes inside an OS**
- **One or more threads in a single process**
  - **Thread:** An actor, that is, an independent locus of computation, realized as a sequence of instructions with own program counter
  - **Scheduler:** Method for selecting which thread to run from the pool of active threads
    - E.g., round robin
- **Shared resources**
  - Memory (global, heap)
  - Hardware devices





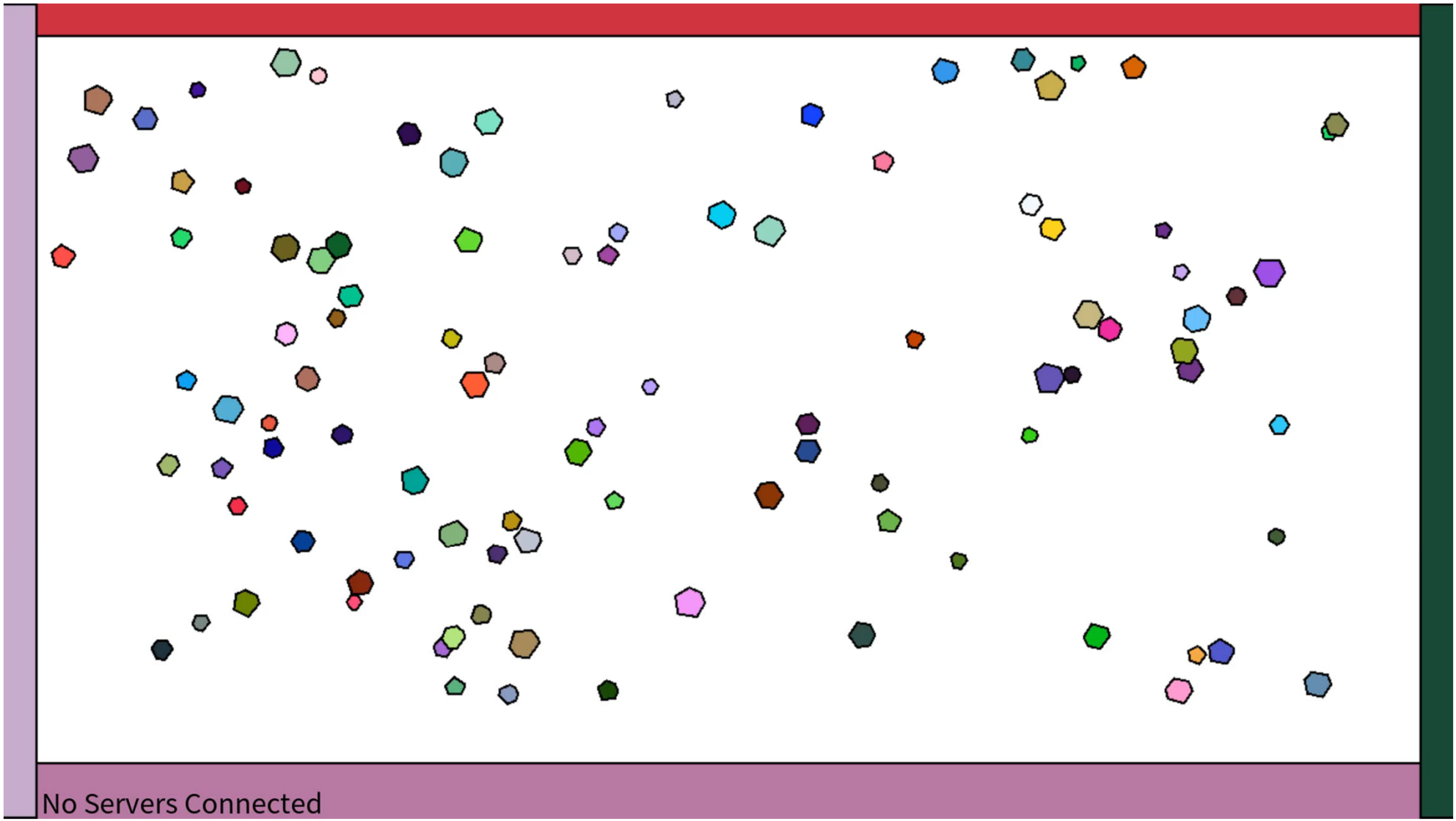
Producer-consumer
(Must share/access same data)



Parallel Execution
(Data shared across threads)

In Computer Systems…

Lancaster University



What Makes Concurrency Challenging

Lancaster University

- To coordinate the actors (as minimally as possible!) so that collectively, they produce only correct outcomes.
- Examples of Correct Outcomes
  - All drink orders should eventually be served, but none more than once.
  - Only one person should be out buying food at any time.
  - Rule out certain manners of traffic accidents

Why Study Concurrency in Computing?

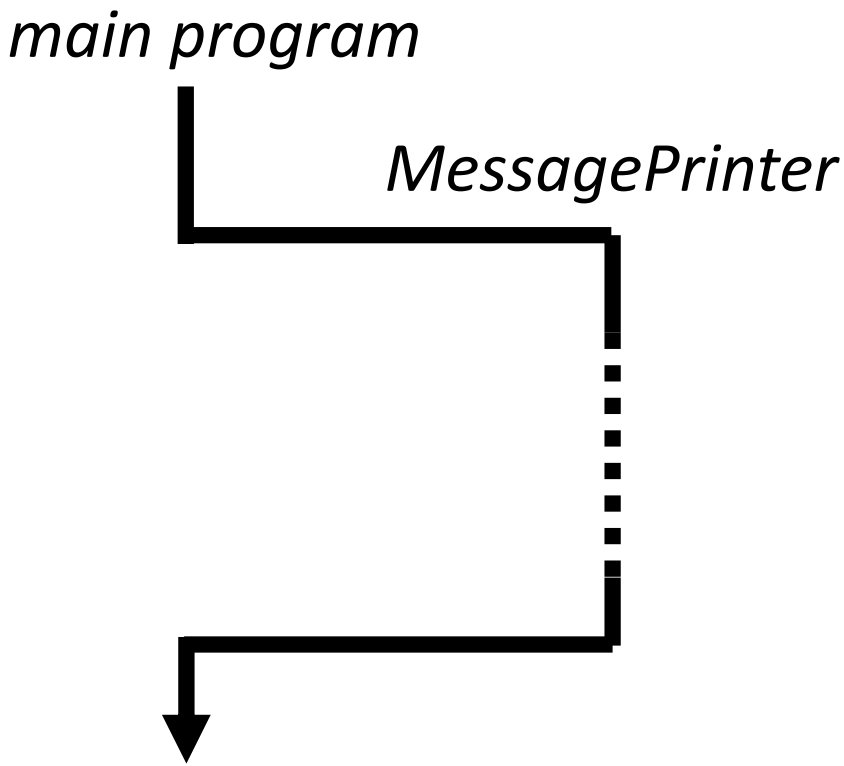
Lancaster University

1. Computing concerns representing (in software) the real-world, which is inherently concurrent because of the presence of multiple actors
   - Coffee shop
   - Business and social interactions
   - Teaching
   - Multiple Readers and Writers for a file
   - Internet of Things: Multiple sensors
2. For responsiveness of software to environmental events
   - E.g., to be responsive to user input. Undesirable: 
3. For utilizing computing resources, e.g., CPUs, more efficiently
   - Structure system/program into multiple actors that can work on tasks independently so that even if some are blocked waiting for some information, the others can proceed.

Java Threads

Lancaster University

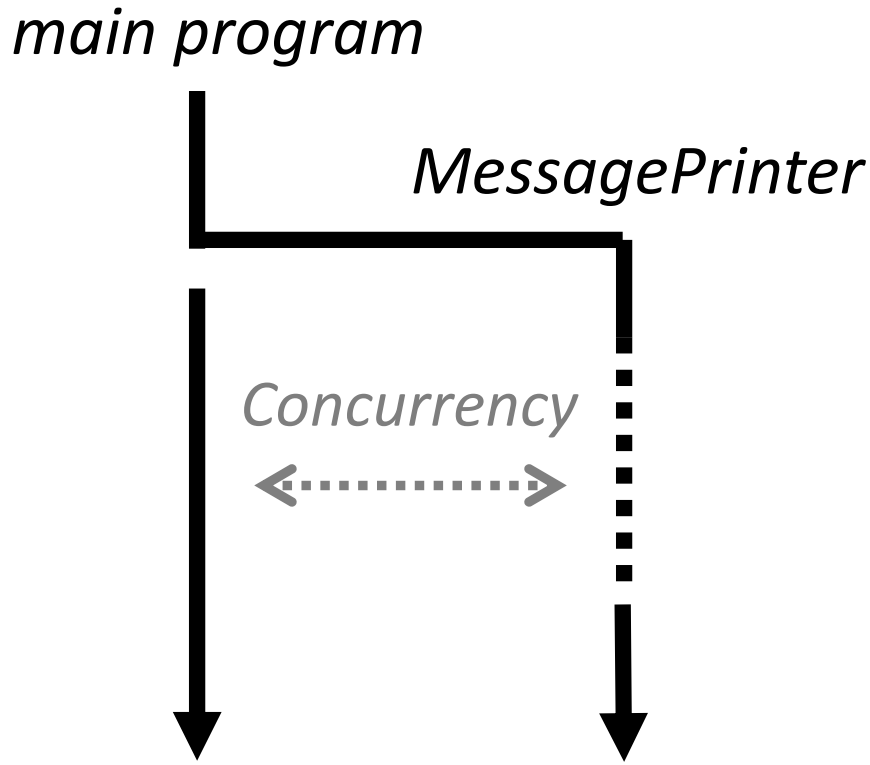
- Consider a **MessagePrinter** class with method **print_string()** that prints a string in a window
- Conventionally, calling program waits “a long time” for print operation completion before able to continue…



Java Threads

Lancaster University

- Can use a thread in MessagePrinter object
- Allow calling program to continue “immediately”



Lancaster University

The "Main" Program part

```
//Instantiate MessagePrinter object for the message
MessagePrinter mp = new MessagePrinter("Hello world");

//create a new thread for the MessagePrinter
Thread t = new Thread(mp);

//Start thread
t.start();

// … do other useful things while message prints
```

Lancaster University

Before showing the MessagePrinter class…

Java interfaces review

- Interface similar to classes: method signatures with no method bodies (implementations)
- Implementing an interface is like extending a class
- Must provide an implementation of all method signatures in the interface

Thread creation in Java

1) Define a class that implements **java.lang.Runnable**
- java.lang.Runnable is an interface
- Which has one method signature to implement: public void run();

2) Create an object of class **Thread** by passing an object of Runnable class to Thread constructor

Lancaster University

MessagePrinter class

```
public class MesssagePrinter implements Runnable {
  String message;
  public MessagePrinter(String m)
  {
       message = m;
  }

  public void run()
  {
     TextArea text = new TextArea(...);
     appendText(message);
     // The thread's work is now complete.
  }
}
```

This implements the Runnable interface

"Implementation" of run() method from Runnable Interface

Summary: Three steps to thread creation in Java

1. **Define class R which implements Runnable**

2. **Create your objects**
   - Make an instance of class R
   - Make a thread instance by passing instance of class *R* to the constructor of class **Thread**

3. **Class start() method of the thread instance**
   - This causes java to immediately execute **R.run()** as **a new thread**

Java Threads Alternative

Lancaster University

Program

```
public class startPrinter{
    public static void main (String[] args)
    {
        MessagePrinter mp = new
        MessagePrinter("I am thread t");
        Thread t = new Thread(mp);
        t.start();
        for(int j = 0; j < 1000; j++)
          System.out.println("I am main thread");
    }
}
```

You can use the *extends Thread*

```
public class MessagePrinter extends Thread{
  String message;
  public MessagePrinter(String m)
  {
          message = m;
  }
  public void run()
  {
        for(int i = 0; i < 1000; i++)
          System.out.println(message);
  }
}
```

Limited by Java's lack of multiple inheritance

Summary

Lancaster University

- **A concurrent system is one that has several independent loci of computation (informally and interchangeably referred to as actors, threads, processes, and so on)**
- The challenge is to ensure correct program outputs even when resources are shared between actors
- Concurrency in software is desirable for many reasons
  - For modelling concurrency in the real-world
  - For making responsive software
  - For taking advantage of computing resources
    - Don’t let CPU idle if there is a task that can use the CPU while others are blocked on IO (input/output)
    - Take advantage of multiprocessor, multicore systems (logically systems with more than one CPU)

Real-time systems, transaction processing systems, Operating Systems, IoT, sociotechnical systems

Acknowledgments

- Based on slides from Dr Peter Garraghan