Introduction

CPEN333 – Software Design for Engineers II

2023 W1

University of British Columbia

CPEN 333

- Course Instructor: Farshid Agharebparast
  - E-mail: farshid@ece.ubc.ca
    - Do include CPEN 333 as a part of a descriptive subject of your emails
  - Office: Kaiser, room 3045
  - Office hours: Please see the course webpage for info (or by appointment)

- Lecture: Fri 13:00-15:00, *MCLD 2018*
  - *(See announcement for location of 1st lecture)*

- Lab section A (Mon 18:00-20:00; *McLeod 4006*) and B (Thu 11:00-13:00; *McLeod 4006)*

CPEN 333

- Software Design for Engineers II

- Course description from the UBC calendar
  - Use of operating systems abstractions;
  - real-time systems;
  - principles of concurrent and multi-threaded programming;
  - information structures;
  - introduction to object-oriented analysis; design, and modelling using UML;
  - software testing.

Tentative learning outcome

- Implement object-oriented programs (in Python).
- Understand operating system abstractions: process, thread ...
- Develop multi-task applications: multiprocessing, multithreading ...
- Solve communication issues between multiple threads and processes.
- Identify and deal with issues of thread/process synchronisation.
- Identify and prevent issues of deadlock and starvation in multi-tasking systems.
- Design, implement and test non-trivial, real-time, multitasking software system.
- Understand different approaches to software development, e.g. Waterfall vs. Agile
- Apply UML to model software system architecture, design and behaviour.
- Understand real-time systems and scheduling.
- Write software test cases.

Programing language

- We will use the Python 3 programing language.
  - We will start by exploring python fundamentals and object-oriented programing with Python.
  - We may also use some C programming (whenever necessary)

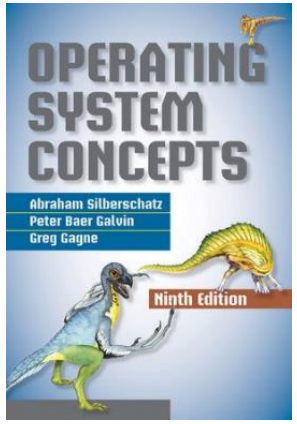
- Object-oriented program design
  - Note that python very well support both procedural and object-oriented programming.

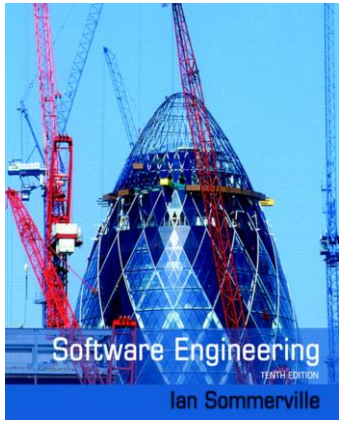
- Tools and methodologies for communicating design using UML (Unified Modelling Language) and systematic testing

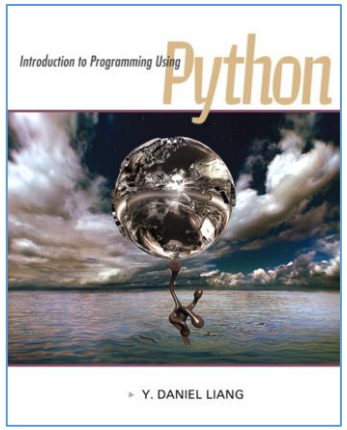
Some References

- Example of online documentations and resources:
  - Python: https://docs.python.org/3/
  - UML standard: https://www.omg.org/spec/UML/

- Books (optional examples):





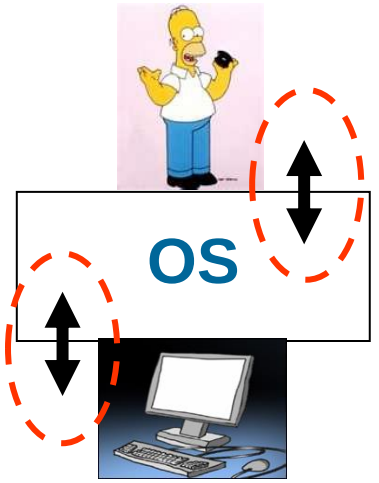


- We will also use other references and books for course concepts such as: multi-tasking, standard library, real-time systems, …
  - e.g. readings or reference links to eBooks available from UBC library

Operating systems

- Operating systems are the essential part of any computer system.
  - Good knowledge of OS abstraction also helps with developing better programs and being a power user.

- An operating system is
  - a program
  - that acts as an intermediary between a user of a computer and the computer hardware
  - and provides an environment in which a user can execute programs.



- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Operating Systems Examples

- Some OS options available:

UNIX

macOS

Windows

Linux

iOS

android

Google Chrome OS

FreeRTOS

Mbed

Images sources: wikipedia.org
(Wikimedia Commons License)

OS abstractions

- OS (from a system view) is a resource allocator and provides many services: process management, memory management, storage management, protection and security, …

- Focus will mainly be on concepts related to process management:
  - process vs thread
  - inter-process communication
  - synchronization
  - …

Multi-tasking

- We will use Python to implement multi-tasking applications
  - Below multi-tasking is used as a general term

- Multi-tasking software (concurrency and parallelism)
  - multi-processing, multi-threading …

- Mechanisms for communication and data sharing

- Mechanisms for synchronization

Real-time systems

- The definition of what constitutes a real-time system may depend on the context.
  - A real-time system implies that there is something significant and important about its response time, involving real-time scheduling.
- Many practical real-time systems are embedded systems that use sensors to obtain real-world input, process it, and generate a response by controlling some actuators.
- Real-time systems classification:
  - Hard real-time systems – task must be serviced by its deadline
  - Soft real-time systems – no definitive guarantee as to when the critical real-time process will be scheduled (system degradation)

Software engineering

- Discussing some important software engineering concepts

- Issues surrounding the processes and methods of engineering software
  - Example: software lifecycle, programming and testing, Agile development, requirements, software architecture modeling and design, etc.

- Using Unified Modeling Language (UML)

Course Webpage

- **Canvas**: http://www.canvas.ubc.ca
  - All course materials will be on Canvas including: announcements, lab instructions, lecture notes, submissions dropboxes, grades …
  - You are expected to check the Announcements regularly for important updates.
- **Discussion Board**: Online discussion boards are solely for academic discussions. You must be professional, courteous and fully follow the rules. All questions or concerns related to grades or personal matters must be communicated with the instructor via email. The violator's access will be revoked, subject to further investigations. See the course's Canvas for more info.

Other Important Info

- Please refer to Canvas for the other related info and documents, including: the course syllabus