CS321
ADVANCED PROGRAMMING TECHNIQUES

Course Outline

Instructor:	Prof. G. Mierzwinski
Office:	TBD – Online Meetings
Phone:	TBD – Email
E-mail:	gmierzwi@ubishops.ca / gregory.mierzwinski@ubishops.ca

Lectures:	**Tuesday-Friday**	**13:30 - 15:00**	**in N211 (Nichols Building)**
Labs:	**Tuesday**	**8:30 - 11:30**	**in N110, N113**

Office-hrs: **None. Only available online by appointment.**

A. Course description

Computer science is often defined as the science of algorithms, their implementation and execution.

In CSC211 you were exposed mainly to algorithmic problem solving and a programming language (C++) with which to express their solutions. Later you were given an introduction to classes and how to model the conceptual space of your problem.

With the increasing complexity of software, another type of problem solving has become of prime importance.. *structural*, *synthetic* or *engineering* problem solving. This has to do primarily with the construction and combination of software objects that solve the original problem. The contrast can be seen, for example, in the difference between describing an algorithm to find a minimum cost path through a graph and constructing an airplane out of ready made components. Component based software construction comes with its own particular challenges that we shall examine in the course.

In CS321 we will explore in further detail the architecture of Object Oriented solutions using Java. Here is a list that broadly covers what we will discuss:

1. Intro to Java – procedural programming with Java - Arrays
2. Introduction to classes and OO modeling
3. Industrial-strength class design,
4. Advanced class features (inner classes, generic classes)
5. Programming with exceptions
6. Inheritance and Polymorphism
7. Object-oriented architectures (Composition vs Inheritance)
8. UML: A language to express design.
9. Architecture of class libraries
10. Software design patterns
11. Serialization, reflection
12. Programming with threads
13. Possibly some newer design methodologies i.e extreme programming, agile programming

In short this course deals with *class design* and the architecture of complex software solutions. i.e. how we specify, construct and combine objects to create new objects that will allow us to model real-world entities in our problem domain. We will also examine and use parts of the design language UML and talk about modes of communication between objects. Much of this will be used in illustrating the construction of class libraries. We shall be using the Java foundation classes and the Swing GUI libraries as practical examples of much of the theory.

B. Organization of the course

The class will meet for 1 1/2 hrs on:

Tuesday – Friday 13:30 - 15:00

C. Evaluation.

The course components are weighted as follows

Component	**Weight**
Final Project	50%
Labs	25%
Assignments	25%

ASSIGNMENTS: There will be 5 programming assignments during the term, each of approximately 2 weeks duration.

Policy on late assignments: *Only 1 assignment can be submitted up to 1 week late. All other late assignment submissions will not be graded and you will receive 0 for it.*

LABS: There will be 10 labs over the entire semester that will give students hands-on experience on the theory of the previous class sessions. The final one or two sessions will be reserved for working on the final project.

Policy on late labs: *Only 1 lab can be submitted up to 1 week late. All other late assignment submissions will not be graded and you will receive 0 for it.*

For programming assignments and labs we will be using the **IntelliJ IDEA Java IDE**.

MIDTERM: Due to the evolving situation and uncertainty surrounding COVID, there will not be a midterm exam this semester. This means that the all other components will have a higher weighting.

FINAL PROJECT: The final examination in this course will consist of a final project that will have you build a simulation of some form and utilize everything you've learned in the course. Here are a couple examples:

- Streaming Service Simulation
-

D. Topics and the tentative sequence of things.

- Introduction, Java overview
- Arrays in Java
- Classes and Object-oriented Programming
- Industrial strength class design
- Cloning
- Documentation with Javadoc
- Data abstraction and the use of Interfaces

- Types of class constructs and their usage
    - Inner classes
    - Anonymous inner classes
    - Local classes etc.
- Name spaces : Packages in Java
- Inheritance
- Types of inheritance
- Introduction to UML

- The Object class (the mother of all objects)
- Polymorphism and Dynamic binding
- Inner classes, anonymous classes
- Java Generics
- First look at a professional class library
    - The Java Collection classes

- The GUI-event handling pattern in Java
- Another professional library: SWING
- All you wanted to know about Exceptions
- Java Input/Output Streams
- Serialization

- **Reflection**

- **Intro to Software Engineering Patterns**
    - Iterator Pattern
    - Composite pattern
    - Decorator Pattern
        - How streams use the decorator
    - Observer pattern
    - Template method pattern
    - Singleton pattern
    - Strategy Pattern
    - Visitor pattern

- **Programming with Threads**

E. Textbooks and other readings:

There will be many readings from journals which I will be putting on-line**.**
Mostly we will be using the online java tutorials at Oracle..
http://docs.oracle.com/javase/tutorial/java/index.html

Here are some books in which you will find most of the material that we will examine

Thinking in Java 4th Ed.
By Bruce Eckel

An excellent book for learning basic and intermediate Java in depth
This also exists as a free on-line book from Eckels' website http://www.mindviewinc.com/Books/
The downloads page has the 3rd edition which mostly will be fine for our purposes.
You can also look it up online here

The Patterns that will be examined are covered in
Design Patterns : Elements of reusable Object-Oriented Software E. Gamma, R Helm, R Johnson, J Vlissides

Head First Design Patterns
by Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra
An excellent introduction to design patterns

Code Craft: The Practice of Writing Excellent Code
by Pete Goodliffe
Recommended supplemental reading.