Course Outline

Internet Programming and Development – LEA.BN

A. General Information

Course name	**Programming II: Object-Oriented Programming (OOP)**
Program Name	Internet Programming and Development
Course number	420-PZ4-AB
Start date	11/13/2020
End date	11/30/2020
Day(s) and times	9:00 – 14:30 with 1/2 hour for lunch
Classroom/lab number	On-line
Ponderation *Ratio of lecture, practical and homework hours*	2-2-3
Hours	60
Credits	2.33
Competency statement(s) and code(s)	DC76: Apply data structures and algorithms to programming
Prerequisite (if any)	(420-PV3-AB) Programming I
Semester	Fall 2020 – IPD-24
Teacher	Reza Shalchian
Teacher's contact info	MIO

B. Introduction

This course is part of the Internet Programming and Development program leading to the *Attestation d'études collégiales* (A.E.C). It should be taken in the 1st semester of the program.

Using a computer workstation connected to the Internet, the student will improve the programming skills learned previously by adding a whole new paradigm: Object-Oriented Programming.
This course focuses on introducing the student to basic principles of Object-Oriented Programming, such as classes, their design and implementation, constructors, try-catch clauses, and exceptions. It also ventures deeper into the standard Java library by exposing the student to Java Collection classes, for-each loop, text file input-output and network connectivity. Emphasis is put on practical application of the skill to create a solid foundation for following programming courses.

C. Course Objectives

By the end of this course, students should be able to perform the following:

Competency code(s) and statements	Element(s) of the competency One per row	Performance criteria (if applicable)
DC76: Apply data structures and algorithms to programming	E1. Analyze the problem	P1.1 Correct specification of input data. P1.2 Correct specification of output data. P1.3 Correct specification of the nature of the procedure. P1.4 Correct identification of the conditions for executing the algorithm. P1.5 Proper identification of the algorithms to be created.
	E2. Organize data in memory	P2.1 Analysis of the context in which the data is to be used. P2.2 Comparison of the features of the different data structures. P2.3 Choice of appropriate data structures. P2.4 Correct creation of arrays, linked lists, stacks, queues, lists, trees in a programming language.
	E3. Model the classes	P3.1 Proper identification of class attributes and methods. P3.2 Proper application of encapsulation and inheritance principles. P3.3 Proper graphic representation of the classes and their relationships. P3.4 Compliance with nomenclature rules.
	E4. Produce algorithms for the methods	P4.1 Choice of a way to represent algorithms that is in accordance with company requirements. P4.2 Definition of a logical sequence of operations.

Competency code(s) and statements	Element(s) of the competency One per row	Performance criteria (if applicable)
		P4.3 Identification of processing structures appropriate for each operation. P4.4 Search for an effective algorithmic solution. P4.5 Precise representation of the chosen algorithmic solution. P4.6 Inclusion of all data necessary to interpret the algorithm. P4.7 Appropriate verification of algorithm correctness. P4.8 Accurate representation of algorithms.
	E5. Translate the algorithm into the programming language	P5.1 Effective use of environment editing features. P5.2 Application of the syntax and semantic rules specific to the language used. P5.3 Rigorous application of coding standards. P5.4 Appropriate application of the principles of structured programming. P5.5 Judicious exploitation of the language's possibilities. P5.6 Recording of pertinent comments in accordance with business requirements.
	E6. Program the classes	P6.1 Appropriate choice of instructions, types of primitive data and data structures.
	E7. Compile the program	P7.1 Effective use of the environment's compilation features. P7.2 Detection of compilation errors. P7.3 Correction of compilation errors.

Competency code(s) and statements	Element(s) of the competency One per row	Performance criteria (if applicable)
	E8. Validate the program	P8.1 Efficient use of the environment's execution and debugging features. P8.2 Correct preparation of the test cases necessary for the verification of the functioning of the program. P8.3 Accurate interpretation of the results. P8.4 Appropriate debugging of the program according to the algorithm. P8.5 Verification of the pertinence of the solution, given the initial situation. P8.6 Identification of the errors and deficiencies of the algorithmic solution developed. P8.7 Appropriate modification of the algorithmic solution.

D. Evaluation Plan

Evaluation	%	Approximate date	Link to competenc(ies) and element(s)
Test 1 Theory	10	**19-11-2020]**	**1 – 8**
Test 1 Practical	10	**19-11-2020]**	**1 – 8**
Test 2 Theory	15	**30-11-2020]**	**1 – 8**
Test 2 Practical	15	**30-11-2020]**	**1 – 8**
Group Project Demo	30	**27-11-2020]**	**1 – 8**
Assignments	20	**During the sessions**	**1-8**
Final evaluation **Minimum of 40% of final grade**			

E. Course Content and Schedule

Date	Objective/Element	Specific content	In class	Location	Online
Class 1		**Objects and Classes** To describe objects and classes, and use classes to model objects To use UML graphical notation to describe classes and objects To demonstrate how to define classes and create objects To create objects using constructors To access objects via object reference variables To define a reference variable using a reference type To access an object's data and methods using the object member access operator (.) To define data fields of reference types and assign default values for an object's data fields To distinguish between object reference variables and primitive data type variables To use the Java library classes Date, Random	☐		☐

Date	Objective/Element	Specific content	In class	Location	Online
		To distinguish between instance and static variables and methods To define private data fields with appropriate get and set methods To encapsulate data fields to make classes easy to maintain To develop methods with object arguments and differentiate between primitive-type arguments and object-type arguments To store and process objects in arrays To create immutable objects from immutable classes to protect the contents of objects To determine the scope of variables in the context of a class To use the keyword this to refer to the calling object itself			
2		**Objects and Classes** To apply class abstraction to develop software To explore the differences between the procedural paradigm and object-oriented paradigm To discover the relationships between classes To design programs using the object-oriented paradigm To create objects for primitive values using the wrapper classes (Byte, Short, Integer, Long, Float, Double, Character, and Boolean) To simplify programming using automatic conversion between primitive types and wrapper class types To use the BigInteger and BigDecimal classes for computing very large numbers with arbitrary precisions To use the String class to process immutable strings	☐		☐

Date	Objective/Element	Specific content	In class	Location	Online
		To use the StringBuilder and StringBuffer classes to process mutable strings			
3		**Inheritance** To define a subclass from a superclass through inheritance To invoke the superclass's constructors and methods using the super keyword To override instance methods in the subclass To distinguish differences between overriding and overloading To explore the toString() method in the Object class To discover polymorphism and dynamic binding To describe casting and explain why explicit downcasting is necessary To explore the equals method in the Object class To store, retrieve, and manipulate objects in an ArrayList To implement a Stack class using ArrayList To enable data and methods in a superclass accessible from subclasses using the protected visibility modifier To prevent class extending and method overriding using the final modifier	☐		☐
4		**Abstract Classes and Interfaces** To design and use abstract classes To generalize numeric wrapper classes, BigInteger, and BigDecimal using the abstract Number class To specify common behavior for objects using interfaces To define interfaces and define classes that implement interfaces To define a natural order using the Comparable interface	☐		☐

Date	Objective/Element	Specific content	In class	Location	Online
		To make objects cloneable using the Cloneable interface To explore the similarities and differences among concrete classes, abstract classes, and interfaces To design classes that follow the class-design guidelines			
5		**Test 1**	☐		☐
6		**Exceptions** To get an overview of exceptions and exception handling To explore the advantages of using exception handling To distinguish exception types: Error (fatal) vs. Exception (nonfatal) and checked vs. unchecked To declare exceptions in a method header To throw exceptions in a method To write a try-catch block to handle exceptions To explain how an exception is propagated To obtain information from an exception object To develop applications with exception handling To use the finally clause in a try-catch block To use exceptions only for unexpected errors To rethrow exceptions in a catch block To create chained exceptions To define custom exception classes To use try-with-resources to ensure that the resources are closed automatically To understand how data is read using a Scanner	☒		☐
7		**Generic , An Introduction to Data Structure** To know the benefits of generics	☐		☐

Date	Objective/Element	Specific content	In class	Location	Online
		To use generic classes and interfaces To declare generic classes and interfaces To understand why generic types can improve reliability and readability To declare and use generic methods To use raw types for backward compatibility To understand that generic type information is erased by the compiler and all instances of a generic class share the same runtime class file To know certain restrictions on generic types caused by type erasure To explore the relationship between interfaces and classes in the Java Collections Framework hierarchy To use the common methods defined in the Collection interface for operating collections To use the Iterator interface to traverse the elements in a collection To use a for-each loop to traverse the elements in a collection To compare elements using the Comparable interface and the Comparator interface To use the static utility methods in the Collections class for sorting, searching, shuffling lists, and finding the largest and smallest element in collections			
8		**I/O** To discover how I/O is processed in Java To distinguish between text I/O and binary I/O	☐		☐

Date	Objective/Element	Specific content	In class	Location	Online
		To read and write bytes using FileInputStream and FileOutputStream To read and write primitive values and strings using DataInputStream/DataOutputStream To store and restore objects using ObjectOutputStream and ObjectInputStream, and to understand how objects are serialized and what kind of objects can be serialized To implement the Serializable interface to make objects serializable To serialize arrays To read and write the same file using the RandomAccessFile class To discover file/directory properties, to delete and rename files/directories, and to create directories using the File class To write data to a file using the PrintWriter class To read data from a file using the Scanner class To develop a program that replaces text in a file			
9		Version Control (Git), Logging, Unit Testing	☐		☐
10		**Project Lab**	☐		☐
11		**Project Demo** Review	☐		☐
12		**Test 2**	☐		☐

F. Required Textbooks / Materials / Course Costs

Title / Item Name	Cost
None required	

G. Bibliography (if applicable)

Add resources here (e-books, articles, videos, websites, etc.)

Introduction to Java Programming by Y. Daniel Liang. Available on loan from the Continuing Education office

H. Teaching Methods

The course is a combination of theory and practical work. Students will be required to:

- Listen to lectures
- Watch demonstrations
- Accomplish regular work in the laboratory
- Work in groups of 1 to 3 students for a project

It requires your individual presence and your active, consistent and sustained participation in your individual work. Your individual responsibilities are to complete the work assigned and be ready to work at the start of each class.

Hands on experience is mandatory to your success in this course. Homework assignments or project milestones are due on the day specified for handing in the assignment, at 11:55pm if no time is specified.

- Lectures/Demonstrations: Important material from the text and outside sources will be covered in class. You should plan to take careful notes as not all material can be found in the texts or readings. Discussion is encouraged as is student-procured, outside material relevant to topics being covered.
- Assignments: Review Assignments, Case Problems, Concepts Reviews, Skills Reviews, Independent Challenges and other projects and readings will be periodically assigned to help support and reinforce material in the course. These assignments may require the application of various software applications.
- Assignment submission: Assignments submission can be done with Lea or a Cloud Source Control tool, as detailed in class.
- Tests: The exams will be closed book/note and will test assigned readings and material discussed in class.
- Practical Test: The practical test will be on a lab computer only, with access to all online documentation, but no communication between students (be it electronic or verbal).
- Team Project (if any): The project focuses on methodologies and tools seen in this course. This project is structured to be small, but somewhat realistic given the time available in the course.
- Classroom Activity: Participation and Discussion
- JAC Portal: All material will be distributed on the JAC Portal. Class notes, instructional material, and student assignments will be posted on the class website. Students are encouraged to go to the website https://johnabbott.omnivox.ca/ in order to obtain file downloads and view other items of interest throughout the semester.

I. Departmental Policies

Please refer to the following documents concerning policies in place at the Centre for Continuing Education:

Summary of Continuing Education Departmental Policies and Guidelines (June 2020)

Online Civility and Student Code of Conduct (Continuing Education version) (June 2020)

J. Classroom Policies

Policy to ensure that issues relating to late submission, or resubmission, of work to be dealt with in an equitable manner A teacher may deduct up to 10% per calendar day for late assignments that are submitted without a valid excuse.
Policy dealing with the expectations of classroom behaviour, including use of cell phones, laptops and other technology
Other

K. College Policies

Please refer to the following document which summarizes some of the key policies in place at the College. See the specific policies for more information.

Summary of College Policies and Guidelines (June 2020)

Cheating and Plagiarism

Please refer to the following documents concerning cheating and plagiarism at John Abbott College:

Policy 7: Institutional Policy on the Evaluation of Student Achievement (IPESA)

See articles **9.1** and **9.2**.

Academic Integrity: Cheating and Plagiarism Procedure

Religious Holidays

Please refer to the following document concerning absences:

Policy 7: Institutional Policy on the Evaluation of Student Achievement (IPESA)

See articles **3.2.13** and **4.1.6**.

Student Rights and Responsibilities

Please refer to the following document concerning student rights and responsibilities:

Policy 7: Institutional Policy on the Evaluation of Student Achievement (IPESA)

See articles **3.2** and **3.3**.

Changes to Course Evaluation Plan in the Course Outline

Please refer to the following document concerning absences:

Policy 7: Institutional Policy on the Evaluation of Student Achievement (IPESA)

See article **5.3**.

Student Code of Conduct

Please refer to the following document concerning the College's student code of conduct and discipline procedures:

Policy 13: Policy on Student Conduct and Discipline Procedures (September 15, 2009)