# Course Outline

# Full-Stack Developer – LEA.BN

## General Information

|  |  |
| --- | --- |
| **Course title** | **Programming II** |
| Course number | 420-JB4-AB |
| Hours | 60 |
| Ponderation  *Ratio of lecture, practical and homework hours* | 2-2-3 |
| Credits | 2.33 |
| Competency statement(s) and code(s) | 00Q6 - Use an object-oriented development approach |
| Prerequisite (s) | 420-JA4-AB |
|  |  |
| Cohort | FSD-05 |
| Start date | May 18, 2022 |
| End date | June 3, 2022 |
| Day(s) and times | M-F: 9:00-12:00 & 12:30-2:30 *(No Class May 23, 2022)* |
| Classroom/lab number | Online |
| Semester | W2022 |
| Teacher | Pargol Poshtareh |
| Teachers’ contact info |  |
| Course format (F2F, online, hybrid) | Online |

## Introduction

|  |
| --- |
| This course is part of the Full-Stack Developer program leading to an Attestation of Collegial Studies (A.E.C.). It should be taken in the first semester of the program.  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. Students will learn about the principles of encapsulation, inheritance, the concept of “static”, interfaces, abstract classes, and polymorphism. Emphasis is put on practical application of the skill to create a solid foundation for following programming courses. |

## Course Objectives

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

|  |  |
| --- | --- |
| **00Q6** | |
| **Statement of the Competency** | **Achievement Context** |
| Use an object-oriented development approach. | * Based on a problem * Using nomenclature and coding rules |
| **Elements of the Competency** | **Performance Criteria** |
| 1. Analyze the problem. | * Breakdown of the problem based on the requirements of an object-oriented approach * Proper identification of input and output data and the nature of the processes * Accurate identification of the classes to be modelled * Proper identification of the algorithms to be created |
| 1. Model the classes. | * Proper identification of class attributes and methods * Proper application of encapsulation and inheritance principles * Proper graphic representation of the classes and their relationships * Compliance with nomenclature rules |
| 1. Produce the algorithms for the methods. | * Appropriate identification of the operations necessary for each method * Proper identification of a logical sequence of operations * Appropriate verification of algorithm correctness * Accurate representation of algorithms |
| 1. Create the graphic interface. | * Appropriate choice of graphic elements for display and data input * Proper layout of graphic elements * Proper set-up of graphic elements |
| 1. Program the classes. | * Appropriate choice of instructions, types of primitive data and data structures * Logical organization of the instructions * Proper programming of messages to be displayed for the user * Proper integration of the classes into the program * Proper program performance * Compliance with the language syntax * Compliance with coding rules |

|  |  |
| --- | --- |
| 1. Document the code. | * Clear comments in the computer code * Clear record of the programming support documentation * Appropriate use of the documentation generators |
| 1. Apply the procedure for managing versions of the programs. | * Proper configuration of the version control system * Systematic submission of the modified code * Sound management of branches and conflicts |

## Evaluation Plan

|  |  |
| --- | --- |
| **INSTRUCTIONS FOR TEACHERS CONCERNING FINAL EVALUATIONS** | |
| **40% MINIMUM** | **40% MAXIMUM** |
| For the final evaluation | For a single evaluation task |
| Each course must have some form of final evaluation of sufficient weighting to attest the student’s achievement of the competencies and the competency elements attached to the course. This evaluation should account for a minimum of 40%. | Given the intensive nature of the Continuing Education A.E.C. courses, a single evaluation task may have a maximum weight of 40% of the final grade. In other words, a single evaluation task with a weighting of 41% and above is not recommended. |
| **The final evaluation may include several evaluations tasks. When combined, these may exceed 40%.**  Examples of a final evaluation  1 final evaluation with a weight of 40%.  5 evaluation tasks with a weight of 10%, for a total of 50%.  1 evaluation task with a weight of 25%, and another with a weight of 20%, for a total of 45%. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation task** | **%** | **Approximate date** | **Link to competency(ies) and element(s)** | **Select if part of the final evaluation!** |
| In Class Exercises 2 @ 5% each | 10 |  | 1-6 |  |
| Assignment 1 | 10 | Class 4 | 1-6 |  |
| Test 1 | 20 | Class 5 | 1-6 |  |
| Assignment 2 | 10 | Class 8 | 1-6 |  |
| Test 2 | 30 | Class 9 | 1-6 |  |
| Project | 20 | Class 12 | 1-7 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Course Content and Schedule

##### Course Content

|  |  |
| --- | --- |
| Objects and Classes  Inheritance  Abstract Classes and Interfaces  Exceptions  Generics  I/O |  |

##### Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date or class** | **Topic(s)** | **Additional info** | **F2F** | **Online** |
| 1 | Objects and Classes | Objects and classes, use classes to model objects  UML graphical notation to describe classes and objects  Define classes and create objects  Create objects using constructors  Access objects via object reference variables  Define a reference variable using a reference type  The object member access operator (.)  Define data fields of reference types, assign default values  Object reference variables and primitive data type variables  Java library classes Date, Random  Instance and static variables and methods  Define private data fields with get and set methods  Encapsulate data fields to make classes easy to maintain  Methods with object arguments, differentiate between primitive-type arguments and object-type arguments  Store and process objects in arrays  Create immutable objects from immutable classes  Scope of variables in the context of a class  Using the keyword this to refer to the calling object itself |  |  |
| 2 | Objects and Classes | Apply class abstraction to develop software  The differences between the procedural paradigm and object-oriented paradigm  The relationships between classes  The object-oriented paradigm  Create objects for primitive values using the wrapper classes (Byte, Short, Integer, Long, Float, Double, Character, and Boolean)  Automatic conversion between primitive types and wrapper class types  BigInteger and BigDecimal classes  The String class to process immutable strings  StringBuilder and StringBuffer classes to process mutable strings |  |  |
| 3 | Inheritance | Define a subclass from a superclass through inheritance  The superclass’s constructors and methods using the super keyword  Override instance methods in the subclass  Differences between overriding and overloading  The toString() method in the Object class  Polymorphism and dynamic binding  Casting and explicit downcasting  The equals method in the Object class  Store, retrieve, and manipulate objects in an ArrayList  Implement a Stack class using ArrayList  Enable data and methods in a superclass accessible from subclasses using the protected visibility modifier  Prevent class extending and method overriding using the final modifier |  |  |
| 4 | Abstract Classes and Interfaces  Assignment 1 Due | Design and use abstract classes  Numeric wrapper classes, BigInteger, BigDecimal, the abstract Number class  Common behavior for objects using interfaces  Define interfaces, define classes that implement interfaces  The Comparable interface  The Cloneable interface  The similarities and differences among concrete classes, abstract classes, and interfaces  Design classes that follow the class-design guidelines |  |  |
| 5 | Test 1 |  |  |  |
| 6 | Exceptions | Exceptions and exception handling  Advantages of using exception handling  Error (fatal) vs. Exception (nonfatal), checked vs. unchecked  Declare exceptions in a method header  Throw exceptions in a method  A try-catch block  How an exception is propagated  Obtain information from an exception object  The finally clause in a try-catch block  Exceptions only for unexpected errors  Rethrow exceptions in a catch block  Chained exceptions  Custom exception classes  Try-with-resources  How data is read using a Scanner  Version Control (Git),  Logging, Unit Testing |  |  |
| 7 | Generics | Benefits of generics  Generic classes and interfaces  Generic methods  Raw types for backward compatibility  Generic type information is erased by the compiler and all instances of a generic class share the same runtime class file  Restrictions on generic types caused by type erasure  Relationship between interfaces and classes in the Java Collections Framework hierarchy  The common methods defined in the Collection interface for operating collections  The Iterator interface to traverse the elements in a collection  The for-each loop to traverse the elements in a collection  Using the Comparable interface and the Comparator interface  Using 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  Assignment 2 Due | How I/O is processed in Java  Text I/O and binary I/O  Using FileInputStream and FileOutputStream  Using DataInputStream/DataOutputStream  Store and restore objects using ObjectOutputStream and ObjectInputStream, how objects are serialized and what kind of objects can be serialized  Implement the Serializable interface to make objects serializable  Serialize arrays  RandomAccessFile class  File/directory properties, delete and rename files/directories, and create directories using the File class  Write data to a file using the PrintWriter class  Read data from a file using the Scanner class  Develop a program that replaces text in a file |  |  |
| 9 | Test 2 |  |  |  |
| 10 | Project |  |  |  |
| 11 | Project |  |  |  |
| 12 | Project Demo |  |  |  |

## Required Textbooks / Materials / Costs

|  |  |
| --- | --- |
| Title / Item | Cost $ |
|  |  |
|  |  |
|  |  |
|  |  |
| Technical requirements for this course (hardware, software, High speed Internet connection, etc.) | |
|  | |

## Bibliography (books, articles, videos, websites, podcasts, etc.)

|  |
| --- |
| Optional:   * Deitel, P. J., & Deitel, H. M. (2018). *Java: How to program early objects* (11th ed.). New York, NY: Pearson. ISBN-13: 9780134743356. * Liang, Y. D. (2020). *Introduction to Java Programming and Data Structures: Comprehensive version* (12th ed.). New York: Pearson. ISBN-10: 0136520235, ISBN-13: 9780136520238. |

## Teaching Methods

|  |
| --- |
| The course is a combination of theory and practical work. Students will be required to:  • Work alone  • Work in groups  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.  Léa, the course management system within Omnivox, will be used in this course.  Learning Activities:  • Lectures/Demonstrations: Important material from the text and outside sources will be covered in class. Taking notes is required as not all material will be in the texts or readings. Discussion is encouraged as is student-procured, outside material relevant to topics being covered.  • Hands-On Exercises/Assignments/Case Problems, Concepts Reviews, Skills Reviews, Independent Challenges: will support and reinforce material in the course. These assignments may require the application of various software applications. Assignments submission will be done with Lea or a Cloud Source Control tool, as detailed in class. Assignments or project milestones are due on the day specified for handing in the assignment, at 11:55pm if no time is specified.  • Tests (Combination of Theory and Practical): The theory component will be closed book/notes and will test assigned readings and material discussed in class. The practical component 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: 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. Project milestones are due on the day specified, at 11:55pm if no time is specified. The project demo occurs during class time.  • Classroom Activity: Participation and Discussion |

## Departmental Policies and Classroom Policies

### Classroom Policies

|  |
| --- |
| Late submission of work  Work submitted late will result in a 10% deduction from the grade, per calendar day. |
| Classroom behaviour |
| Online etiquette |

### Departmental Policies

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

[Continuing Education Policies and Guidelines](http://continuingeducation.johnabbott.qc.ca/continuing-education-policies-december-1-2020-final/)

(version: December 1, 2020)

## College Policies

Please refer to the following document concerning the provisos related to course outlines as a response to Covid-19.

### [Provisos for Course Outlines (Covid-19)](http://continuingeducation.johnabbott.qc.ca/wp-content/uploads/2022/01/Covid-19-provisos-updated-January-12-2022.pdf)

(version: winter 2022)

|  |  |
| --- | --- |
| **Topic** | **Resource** |
| Student rights and responsibilities  (see articles 3.2 and 3.3) | [Policy 7:IPESA - Institutional Policy on the Evaluation of Student Achievement (version: June 12, 2019)](http://departments.johnabbott.qc.ca/publications/Policies/Policy%20No.%207%20-%20IPESA%20FINAL.pdf) |
| Changes to evaluation plan in the course outline  (see article 5.3) |
| Religious holidays  (see article 4.1) |
| Cheating and plagiarism  (articles 9.1 and 9.2) |
| Cheating and plagiarism | [Academic Integrity: Cheating and Plagiarism Procedure (version: October 22, 2021)](https://johnabbott.omnivox.ca/intr/academic/policies/)  ***You will need to log into Omnivox to access this document.*** |
| Code of conduct | [Policy 13: Policy on Student Conduct and Discipline Procedures (version: September 21, 2021)](https://www.johnabbott.qc.ca/wp-content/uploads/2021/09/POLICY-13_-_policy_on_student_conduct-Final-BOG-2021-09-21.pdf) |

*DISCLAIMER: Policies may be updated during the academic year. Should a link in the section above no longer work, please refer to the college website:* [*https://www.johnabbott.qc.ca/the-college/official-documents/*](https://www.johnabbott.qc.ca/the-college/official-documents/)