UNIVERSITY OF TECHNOLOGY, JAMAICA

SYLLABUS OUTLINE

FACULTY: Engineering & Computing (FENC)

SCHOOL/DEPT: School of Computing & Information Technology (SCIT)

COURSE OF STUDY: Bachelor of Science in Computing

LEVEL: Two (2)

MODULE TITLE: **OBJECT ORIENTED PROGRAMMING**

MODULE CODE: **CIT2004**

DURATION (Hours): Sixty (60)

CREDIT VALUE: Four (4)

PREREQUISITES: Programming II (CMP1025)

1.0 MODULE DESCRIPTION

This module introduces students to the key concepts of the object oriented programming (OOP) paradigm, exploring analysis, design and implementation activities. Students are taught to analyze a list of requirements, then design and implement their own solutions in any of the popular OOP languages such as Java and C++.

2.0 MODULE OBJECTIVES/LEARNING OUTCOMES

Upon completing the module, the student should:

i. recognize the characteristics of the Object Oriented paradigm and Other paradigms
ii. generate an Object Oriented Analysis & Design (OOA & D)
iii. create an Object Oriented Analysis (OOA)
iv. create an Object Oriented Design (OOD) with Unified Modeling Language (UML)
v. perform testing and debugging of Object Oriented programs
vi. create an Object Oriented program from a UML design
vii. judge the main concepts of Object Oriented Programming (OOP)
viii. appraise Object Oriented programs

3.0 MODULE CONTENT

UNIT 1 Object Oriented Programming, Classes, Data Abstraction and Information Hiding (8 hrs. lect., 4 hr. tut, 12 hrs. lab)

Expected Learning Outcomes:
Upon completion of Unit 1, the student should be able to:

1.1. list the main characteristics of the Object Oriented Programming (OOP) paradigm
1.2. differentiate between the Object Orient Programming paradigm and other programming language paradigms such as the Procedural Programming paradigm
1.3. explain the main characteristics of the Object Oriented Programming (OOP) paradigm
1.4. prepare an Object Oriented Analysis (OOA) from the requirements of a small application
1.5. construct an Object Oriented Design (OOD) using the Unified Modeling Language (UML) from the OOA for a small application
1.6. implement the OOD for a small application in a supporting programming language such as C++ or Java
1.7. explain the use of access modifiers used to hide or expose class members
1.8. make use of the three types of constructors in class implementations
1.9. explain the use of dynamic memory allocation for objects of a class
1.10. make use of constant and static values

Content

- **Programming Language Paradigms**
  - **Characteristics of the paradigms**
  - **Object Oriented Programming Paradigm vs. Other Programming Paradigms**
- **Development of applications using Object Oriented Programming**
  - **Object Orient Analysis (OOA)**
    - **Encapsulation and the role of Access Specifiers / Modifiers**
    - **Dependency, Association, Generalization and Realization**
  - **Object Oriented Design (OOD) with Unified Modeling Language (UML)**
    - **Classes and Class Relationships**
  - **Implementation using an Object Oriented Language**
    - **Default, Primary and Copy constructors**
    - **Constant and Static values**
    - **Dynamic Memory Allocation**
    - **Separating classes and driver files**
- **Classes and Objects**
  - **Definition of terms - “class” and “object”**
  - **Class members (attributes and methods)**
  - **Method types**
- **Definition of object-based programming**

UNIT 2 Inheritance, Overriding and Overloading, and Polymorphism (10 hrs. lect., 5 hr. tut, 15 hrs. lab)

Expected Learning Outcomes:
Upon completion of Unit 2, the student should be able to:

2.1. explain how inheritance facilitates software reuse
2.2. identify candidates for inheritance relationships during object-oriented analysis
2.3. represent inheritance relationships in UML diagrams
2.4. represent inheritance relationships in code
2.5. identify method overriding and overloading
2.6. differentiate between overriding and overloading
2.7. implement overriding and overloading in code
2.8. explain why objects of derived classes can be treated as objects of their base class
2.9. state the order in which constructors and destructors are called in an inheritance hierarchy
2.10. differentiate between concrete classes and abstract classes
2.11. explain the Yo-Yo problem
2.12. implement single and multiple inheritance
2.13. demonstrate sub-type polymorphism

Content

- **Introduction to Inheritance**
- **Inheritance "is a" Generalization**
- **Single and Multiple inheritance**
- **Base class objects and Derived class objects**
- **Constructors, Destructors, and Inheritance**
- **Overriding and Overloading**
  - **The "Anatomy" of a Method**
  - **Understanding Overriding**
  - **Understanding Overloading**
- **Concrete Class vs. Abstract Class**
- **Polymorphism**
  - **Types of polymorphism**
  - **Liskov's Substitution Principle**
  - **Dynamic vs. Static Binding**
- **The Yo-Yo problem**

UNIT 3 Exception Handling, Object Persistence, and Emerging Trends (8 hrs. lect., 4 hr. tut, 12 hrs. lab)

Expected Learning Outcomes:
Upon completion of Unit 3, the student should be able to:

3.1. state what type of errors exception handling is able to handle
3.2. compare and contrast traditional error handling with exception handling
3.3. implement defensive programming using exception handling
3.4. define object persistence
3.5. explain the difference between sequential and random access files
3.6. implement sequential files in a complete program to achieve persistence
3.7. rewrite code to initialize a random access file for first time use
3.8. rewrite code to store and retrieve information to and from a random access file
3.9. implement random access files in a complete program to achieve persistence
3.10. describe emerging trends in object-oriented programming, citing technical reports, and variances in language implementation.
3.11. list sources of information on the nuances of a specific implementation of a particular object-oriented language on a specific platform.
3.12. explain how to make the transition to using another OOP language

Content

- **Traditional Error Handling**
- **Exception Handling**
- **Models of Exception Handling**
- **Checked vs. Unchecked Exceptions**
- **Unhandled exceptions**
- **Quirks and problems with exception handling**
- **Persistence and Streams**
- **File Streams, Organization, and Access Methods**
- **Sequential Access Files**
    - **Characteristics of Sequential Access Files**
    - **Working with Sequential Access Files**
- **Random Access Files**
    - **Characteristics of random access files**
    - **Working with Random Access Files**
- **Technical reports and Core language features**
- **Language variants and New OOP languages**

4.0 LEARNING AND TEACHING APPROACHES

- Lectures to provide coverage of the concepts and techniques to generate understanding.
- Tutorials to provide opportunities for a high level of interaction and discussion of the concepts.
- Student participation in labs and tutorials to create an environment in which all students have the opportunity to learn, develop insight into the material and provide useful critique and feedback on the issues explored, ultimately providing a rich overall learning experience
- Case discussion and analyses to provide real-world understanding, critical thinking and discussion of issues related to data structures.
- Supervised and unsupervised lab sessions to facilitate demonstration, practice and application of techniques, to develop students' competence in the use of the supporting programming language tools and to broaden understanding of the concepts covered in the module

5.0 ASSESSMENT PROCEDURES

Coursework **50%**

Lab Project 10%
This project will assess the student's ability to design a solution to a problem using the techniques taught and implement the solution using the supporting programming language.

Theory Tests (2 tests @ 10% each) 20%
The theory tests provide the opportunity to assess the knowledge and competencies expressed in the learning outcomes.

Lab Tests (2 tests @ 10% each) 20%
The lab tests provide the opportunity to assess the individual student's ability to implement the data structures and algorithms using the techniques taught, using the supporting programming language.

Final Examination **50%**

6.0 BREAKDOWN OF HOURS

Lecture		26 (2 hours * 13)
Tutorial		13 (1 hour * 13)
Lab	– *supervised*	13 (1 hour * 13)
	– *unsupervised*	30 (2 hours * 15)
Assessment		8 (8 hours)

Total ***90 hours***

7.0 TEXTBOOKS AND REFERENCES

Recommended Text:

1. *C++ How to Program*. P. J. Dietel and H. M. Dietel. Prentice Hall. Latest Edition.
2. *Java How to Program*. P. J. Dietel and H. M. Dietel. Prentice Hall. Latest Edition.
3. *The Unified Modeling Language User Guide*. G. Booch, J. Rumbaugh and I. Jacobson. Addison-Wesley. Latest Edition.
4. *Thinking in C++*. B. Eckel. Latest Edition.
5. *Thinking in Java*. B. Eckel. Prentice Hall. Latest Edition.
6. *Head First Java*. K. Sierra and B. Bates. O'Reilly Media. Latest Edition.

8.0 SYLLABUS WRITER/DEVELOPER

Janet Walters
Devon M. Simmonds
Jeffrey Roach
Jeffrey Roach, Christopher Slowley and Felix Akinladejo
Lalitha Jonnalagadda
David W. White

9.0 DATE OF REVISION

September 4, 2011

10.0 DATE OF ACCEPTANCE

_____
(Programme Director)

_____
(OCDE)