COURSE TITLE: **Object-Oriented Programming II**

COURSE CODE: COMP 3607

TYPE: Elective

LEVEL: 3

SEMESTER: 1

START DATE: SEP-03-2018

DEPARTMENT and FACULTY: DCIT/FST

CREDITS: 3

PRE-REQUISITE(S): COMP 2603

ESTIMATED STUDY HOURS:

Two 1-hour lectures, One 2-hour lab, 6 hours per week independent study

Lectures:

Day	Time	Room
Monday	9:00 a.m. to 9:50 a.m.	FST 114
Wednesday	9:00 a.m. to 9:50 a.m.	FST 113
Friday	12:00 p.m. to 1:50 p.m.	CSL1

LECTURER: Dr. Phaedra Mohammed (Phaedra.Mohammed@sta.uwi.edu)

Office Hours: Mondays 10:00 a.m. to 11:00 a.m.
Wednesdays 10:00 a.m. to 11:00 a.m.

TUTOR: Mr. Inzamam Rahaman (Inzamam.Rahaman@sta.uwi.edu)

COURSE OVERVIEW

This course looks at the main tools of modern object-oriented software development. These include design-support techniques and tools (principally design patterns) and programming-support tools (principally IDEs). The course has a strong emphasis on project design and programming using object-oriented software design patterns. Each pattern represents a best practice solution to a software problem in a specific context and the course examines the rationale and benefits of using patterns for such cases. Numerous problems will be studied to investigate the implementation of good design patterns.

Computer programmers and software engineers must be able to use a variety of programs and systems for designing solutions to common information technology issues. The object-oriented programming paradigm has made it easier to handle software development involving complex tasks since it easily facilitates the decomposition of problems into modular entities. The course will allow students to practice advance concepts in object-oriented design. This course will help motivated students to be primary contributors to any small to mid-sized commercial or open-source software project.

COURSE CONTENT

1. Principles of good software design. Examine what causes software to rot.
2. Unified Modelling Language (UML) - Class, Sequence and Use-Case Diagrams
3. Design Patterns: Strategy, Observer, Factory, Singleton, FlyWeight, Command, Adapter, Facade, Template Method, Iterator, Composite, State, Proxy and Mediator
4. Object Persistence using relational databases and other forms
5. Model View Controller (MVC) architecture
6. Concepts of Code Refactoring and testing

COURSE LEARNING OUTCOMES

Upon the successful completion of this course, the student will be able to:

1. Analyze problems using UML tools
2. Design object-oriented solutions using UML tools
3. Discuss the principles of good object-oriented design
4. Use design patterns to facilitate good object-oriented design
5. Explain the reasoning for each object-oriented design principle.
6. Apply knowledge of design patterns to solve common programming problems
7. Draw high level class diagrams in UML for each pattern.
8. Describe the consequences of applying each pattern to the overall software quality of a system.

COURSE ASSESSMENT (Approximate)

Assessment	Learning Outcomes								Weighting %	Assessment Description
	1	2	3	4	5	6	7	8		
Assignment 1	X	X	X	X					20%	Programming problems, design exercises, short answer questions
Assignment 2	X	X	X	X	X	X				
Assignment 3	X	X	X	X	X	X	X			
Coursework Exam 1	X	X	X	X	X	X			30%	Problems and short answer questions
Coursework Exam 2	X	X	X	X	X	X	X			
Final Examination	X	X	X	X	X	X	X	X	50%	Problems and short answer questions
TOTAL %									100%	

TEACHING STRATEGIES

Problem-based learning will be used as the main teaching strategy for the course. Interactive lectures and labs will be used to demonstrate, in a practical manner, the concepts taught in lectures. Feedback from assignments/problem sheets/course work tests will be used to tailor subsequent lectures/tutorials as necessary. Further, students will be given ample opportunity to develop correct, readable programs using object-oriented programming techniques.

RESOURCES

Lecture notes: Available on myElearning - see course website

Reading Materials (Selected Chapters):

- Craig Larman (1998). Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design. Prentice Hall.
- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides (1977). Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley.
- Martin Fowler (1999). Refactoring: Improving the Design of Existing Code. Prentice Hall.
- Permanand Mohan (2013). Fundamentals of Object-Oriented Programming in Java, CreateSpace Independent Publishing Platform. ISBN-10: 1482587521
- Harvey Deitel and Paul Deitel (2002). Java: How to Program, 5th Edition. Prentice-Hall.
- Bruce Eckel (2002). Thinking in Java, 3rd Edition. Prentice-Hall. Free electronic copy available on the Internet from http://www.mindview.net/Books/TIJ/.

Design Tools and IDEs:

- StarUML: http://staruml.io/
- IntelliJ IDEA: https://www.jetbrains.com/idea/
- Netbeans: https://netbeans.org/

COURSE CALENDAR (Approximate):

Week	Topic
1-2	UML - Class, Sequence and Use-Case Diagrams using a UML Modelling Tool Principles of good software design. Software rot. Introduction to Code Refactoring Introduction to Design Patterns: Singleton **Assignment 1 (Released: Week 2)**
3-7	Strategy, Observer, Factory, FlyWeight, Command, Adapter, Facade, Template Method, Iterator, Composite, State, Proxy and Mediator **Assignment 2 (Released: Week 5)** **Course work Exam 1 (Week 6**)
8	Code Refactoring and Testing (JUnit) **Assignment 3 (Released: Week 9)**
9	Model View Controller architecture
10	**Course work Exam 2 (Week 10)**
11	Object Persistence: using relational databases
12	Putting it all together
13	Revision