THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE, TRINIDAD AND TOBAGO

DEPARTMENT OF COMPUTING AND INFORMATION TECHNOLOGY

FACULTY OF SCIENCE AND TECHNOLOGY

COMP 2603 - OBJECT ORIENTED PROGRAMMING I

Semester 1, Undergraduate, Year 2, Level 2

Pre/Co-requisites: **COMP 1603**
Course Type: **Core**
Credits: **3**
Mode of Delivery: **Face-to-Face**
Estimated Study Hours:

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

1. Course description

This course provides a comprehensive introduction to the concepts and techniques of object oriented programming. Students will be taught fundamental object oriented constructs such as classes, methods and objects and core concepts such as encapsulation, abstraction, inheritance and composition. Students will learn how to develop user interfaces using an object oriented toolkit (e.g. Swing in Java) and the course gives a preliminary introduction to object oriented design. At the end of the course, students will be able to develop object oriented programs. The course will be delivered using a combination of face-to-face lectures and interactive hands-on computer lab sessions, along with eLearning activities using various online resources. Assessments will take the form of written examinations and individual programming assignments.

2. Rationale

An object oriented programming paradigm essentially aims to model real world artefacts in a manner that is modular, reusable and flexible. This course shows students how to break down large modelling problems into sub-problems and develop solutions that feature self contained, modular code.

3. Course Aims

COMP2603 aims to develop practical object oriented programming skills in undergraduate students while promoting an understanding of the theoretical concepts and design considerations behind these techniques. The course also aims to expose students to the development tools and programming APIs commonly used in the field.

4. UWI Graduate Outcomes

This course concentrates on the following qualities of the distinctive UWI graduate:

1. Design and implement systems and software based upon critical review of developing trends, approaches and research.
2. Solve computing problems using algorithms and multiple programming languages.
3. Apply computer science theories and methods in the design and analysis of systems producing a solution to a variety of problems across multiple disciplines.
4. Design a system, component or process that illustrates the interplay between theory and practice.

5. Course Learning Outcomes

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

1. Describe the fundamental concepts and vocabulary of an object-oriented approach.
2. Analyse a real-world situation in an object-oriented way.
3. Understand and interpret designs expressed in UML diagrams.
4. Design object-oriented solutions containing multiple classes and collaborations.
5. Implement object-oriented models using an appropriate programming language.
6. Develop graphical user interfaces using objects within a programming framework.
7. Generate event-driven code for making graphical user interfaces interactive and functional.
8. Explain the impact of small design changes on object-oriented program behaviour/outcomes.
9. Apply basic object-oriented techniques to real-world programming problems.
10. Use appropriate collections for data storage and manipulation.

6. Program Goals and Course Learning Outcomes Matrix

Programme Level Learning Outcomes **At the end of the programme, students will be able to:**	**Course Learning Outcomes** **At the end of the course, students will be able to**									
	1	2	3	4	5	6	7	8	9	10
Design and implement systems and software based upon critical review of developing trends, approaches and research.	x	x	x							
Devise new and innovative ways to use computers in other disciplines such as in science (e-science) and the arts.				x	x			x	x	x
Solve computing problems using algorithms and multiple programming languages					x	x	x		x	

Programme Level Learning Outcomes At the end of the programme, students will be able to:	Course Learning Outcomes At the end of the course, students will be able to									
	1	2	3	4	5	6	7	8	9	10
Demonstrate inter-personal skills, teamwork, and efficient use of appropriate technology.				x	x	x	x			
Develop a professional development framework for maintaining expertise in the field of computer science.	x	x							x	x
Apply computer science theories and methods in the design and analysis of systems producing a solution to a variety of problems across multiple disciplines	x	x	x	x				x	x	x
Design a system, component or process that illustrates the interplay between theory and practice.								x	x	x
Demonstrate effective written and oral communication techniques when completing reports and presentations.				x	x	x	x			

7. Course Assessment Descriptions

In this course, assessments are designed to test student knowledge and practical skills through a combination of theoretical/analytical exercises and practical software-related problem-solving tasks.

There are four coursework assessments in this course worth 50% collectively: 2 programming assignments and 2 written coursework exams based on theoretical and practical concepts. The remaining 50% is assessed in a final 2 hour written examination.

All assignments must be submitted through myElearning by the respective deadlines. Emailed assignments will not be accepted.

Assessment	Learning Outcomes										Weighting %	Assessment Description	Duration
	1	2	3	4	5	6	7	8	9	10			
Assignment 1			✓	✓						✓	5%	Take home programming exercises	7-10 days
Assignment 2			✓	✓	✓	✓				✓	10%		10-14 days
CW Exam 1	✓	✓	✓	✓	✓			✓	✓		15%	Short answer theory questions, programming code exercises, problem-solving tasks, code analysis	1.5 hours
CW Exam 2	✓	✓	✓	✓	✓		✓	✓	✓	✓	20%		
Final Examination	✓	✓	✓	✓	✓		✓	✓	✓	✓	50%		2 hours
TOTAL %											100%		

8. University Grading Scheme (Undergraduate Level)

Grades	Ranges
A-to A+	*(A- : 75 to 79; A: 80 to 89; A+: 90 to 100)*
B-to B+	*(B-: 60 to 64; B: 65 to 69; B+: 70 to 74)*
C-to C+	*(C-: 50 to 54; C+: 55 to 59)*
F1 to F3	*(F1 40 to 49; F2: 30 to 39; F3: 0 to 29)*

9. Teaching Strategies

Teaching Method	Description
Interactive Lectures	Live lectures delivered twice weekly that include breaks for student activities at appropriate intervals.
Direct Instruction	Lectures and labs include time for guided and independent practice. Activities: Create mind/concept maps, free writes, one-sentence summary, one minute papers
Guided Instruction	Direct and structured instruction that includes extensive instructor modelling and student practice time during labs. Showing and explaining examples, model strategies, demonstrate tasks, classify concepts, define vocabulary, scaffold steps
Inquiry-based Learning	Students learn or apply material in order to meet a challenge, answer a question, conduct an experiment, or interpret data. Code writing / problem solving during lab sessions
Problem-based Learning	Students produce one or more solutions or resolutions to problems presented in a realistic story or situation, exercises/ activities/tasks from lectures, labs and worksheets
Directed Discussions	Online and during lectures and lab sessions. Direct, specific, or open-ended questions that are connected to learning outcomes and include varied cognitive processes

10. Learning Resources/ Reading

Required/Essential

- Benjamin J. Evans and, David Flanagan (2023). Java in a Nutshell: A Desktop Quick Reference. O'Reilly Media; 8th Edition
- Kishori Sharan and Peter Späth (2022). Learn JavaFX 17: Building User Experience and Interfaces with Java . Apress; 2nd Edition

Online Resources

- BlueJ IDE: https://www.bluej.org/
- Lecture notes: Available on myElearning. See course website.
- Ralph Morelli and Ralpha Walde. (2016) Java, Java, Java: Object-Oriented Problem Solving (available online at: https://open.umn.edu/opentextbooks/textbooks/java-java-java-object-oriented-problem-solving)

11. Course Calendar (Approximate)

Week	Topic	Reading	Learning Activities	Assessment	Assessment Date
1	Intro to Java, Classes & Objects	Lecture Notes on myeLearning, selected chapters/ sections from recommended texts, online tutorials/ guides	Lectures, Lab Sessions, Discussion Forum		
2	Encapsulation, Information Hiding, Classes & Methods		Lectures, Lab Sessions, Online Video Quiz	Assignment 1 Given	Week 2
3	Object Relationships and Equality		Lectures, Lab Sessions, myeLearning Quiz	Assignment 1 Due	Week 3
4	Inheritance, Polymorphism		Lectures, Lab Sessions, myeLearning Wiki		
5	Polymorphism, Abstract Classes		Lectures, Lab Sessions, Discussion Forum		
6	Interfaces and Abstractions		Lectures, Lab Sessions, Online Video Quiz	CW Exam 1	Week 6
7	Container Classes		Lectures, Lab Sessions, myeLearning Quiz	Assignment 2 Given	Week 7
8	Graphical User Interfaces		Lectures, Lab Sessions, myeLearning Wiki		

9	Event Driven Programming		Lectures, Lab Sessions, Discussion Forum	Assignment 2 Due	Week 9
10	Design Fundamentals		Lectures, Lab Sessions, Online Video Quiz		
11	Container Classes		Lectures, Lab Sessions, myeLearning Quiz	CW Exam 2	Week 11
12	Course Review		Lectures, Lab Sessions, myeLearning Wiki		

12. University Policies and Expectations

a. Academic Integrity

Students are required to submit their own work for all assessments in this course. Plagiarism, collusion or copying of any form will not be tolerated. Offenses will be reported to the Department Head and university plagiarism policies will be applied. Cite sources fully when preparing presentations and reports. For all coursework and final exams, students are expected to do their own work without the assistance of the internet or personal notes unless allowed. You are not to share or discuss your working, designs, code files or solutions with other students. In the case of multiple streams weekly for lab exams (if applicable), you are expected to maintain the confidentiality of these exams. You are hereby prohibited from reproducing, re-publishing, re-broadcasting, re-posting, re-transmitting or transferring in whole or in part any Course Outlines, Course Materials or Lectures which have been provided to you as part of your course of study at The University of the West Indies (The UWI), without the prior permission of The UWI its authorised agents or copyright holders.

b. Accommodations for students with disabilities

Students should refer to the University of the West Indies St Augustine Campus, Student Disability policy https://sta.uwi.edu/resources/policies/Student_Disability.pdf

c. Course Conduct

Students are expected to:

- attend all lecture and lab sessions, follow instructions on handouts, review notices and content posted on myElearning weekly (at minimum). Less than 75% attendance may result in debarment from examinations. Polite, respectful behaviour is expected in all sessions.
- submit all assignments on time, and attend all coursework examinations. Students are also expected to verify their marks and submit queries, if any, within the time frames posted.
- report missed assignments or coursework examinations to the lecturer via UWI email within 3 days of the assignment's due date or the coursework examination date. Assignment due date extensions or makeup coursework examinations are only granted due to valid medical reasons, verified by the HSU.
- correspond professionally with the lecturer and tutor using UWI email accounts, where the course code is cited in email subjects, or via myeLearning messages/forums.

LECTURER: Dr. Phaedra Mohammed **EMAIL:** Phaedra.Mohammed@sta.uwi.edu

TUTOR: Mr. Daniel Rasheed **EMAIL:** daniel.Rasheed@sta.uwi.edu

COURSE CONTENT (not in order)

1.0. Object-Oriented Programming
1.1. Object-oriented design
    1.1.1. Decomposition into objects carrying state and having behaviour
    1.1.2. Class-hierarchy design for modelling
1.2. Definition of classes: fields, methods, and constructors
1.3. Subclasses, inheritance, and method overriding
1.4. Dynamic dispatch: definition of method-call
1.5. Subtyping (cross-reference PL/Type Systems)
    1.5.1. Subtype polymorphism; implicit upcasts in typed languages
    1.5.2. Notions of behavioural replacement:subtypes acting like supertypes
    1.5.3. Relationship between subtyping and inheritance
1.6. Object-oriented idioms for encapsulation and information hiding
    1.6.1. Privacy and visibility of class members
    1.6.2. Interfaces revealing only method signatures
    1.6.3. Abstract base classes
1.7. Using collection classes, iterators, and other common library components
1.8. Overloading, overriding of methods, constructors
1.9. Method types: accessors, mutators, static methods, instance methods
2.0. Basic Type Systems
2.1. Type safety and errors caused by using values inconsistently given their intended types
2.2. Goals and limitations of static typing
2.3. Generic types (parametric polymorphism)
2.4. Complementary benefits of static and dynamic typing
3.0. Algorithms and Design
3.1. Fundamental design concepts and principles
4.0. Fundamental Programming Concepts
4.1. Basic syntax and semantics of a higher-level language
4.2. Variables and primitive data types (e.g., numbers, characters, Booleans)
4.3. Expressions and assignments
4.4. Simple I/O including file I/O
4.5. Conditional and iterative control structures
4.6. Functions and parameter passing
5.0. Fundamental Data Structures
5.1. Arrays
5.2. Strings and string processing

5.3. Abstract data types and their APIs
5.4. Strategies for choosing the appropriate abstract data structure
6.0. Event-Driven and Reactive Programming
6.1. Events and event handlers
6.2. Separation of model, view, and controller
7.0. Software Design
7.1. System design principles: levels of abstraction (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion, re-use of standard structures
7.2. Design Paradigms such as structured design (top-down functional decomposition), object-oriented design, event driven design, function oriented, service oriented
8.0. Designing Interaction and HCI Foundations
8.1. Elements of visual design (layout, colour, fonts, labelling)
8.2. Principles of good design and good designers; engineering tradeoffs 9.0. Programming Interactive Systems
9.1. Software Architecture Patterns, e.g.,Model-View controller; command objects, online, offline (cross
9.2. Event management and user interaction
9.3. Modern GUI libraries (e.g. JavaFX) GUIbuilders and UI programming environments