MODULE PROFORMA		
Full module title: Object Oriented Programming		
Module code: 5COSC019C	Credit level: Level 5	Length: 1 Semester
UK credit value: 20	ECTS value: 10	
College and School: College of Design, Creative and Digital Industries; School of Computer Science and Engineering		
Module Leader(s): Saman Hettiarachchi		
Extension:	Email: saman.h@iit.ac.lk	
Host course and course leader: BSc Computer Science		
Status: Core - BSc Computer Science, BEng Software Engineering; Option - BSc Data Science & Analytics		
Subject Board: COMENG		
Pre-requisites: None	Co-requisites: None	
Study abroad: An alternative assessment can be produced to evaluate whether Study Abroad Students have passed the Learning Outcomes.		
Special features: None		
Access restrictions: None		
Are the module learning outcomes delivered, assessed or supported through an arrangement with an organisation(s) other than the University of Westminster. - No		
Summary of module content This module will teach the fundamental ideas behind the object-oriented approach to programming. It will provide students with knowledge and practical experience in writing computer programmes using object-oriented programming techniques. It will cover in a practical way the design and implementation of object-oriented software for software applications through the entire software development lifecycle.		

Learning outcomes

By the end of the module the successful student will be able to:

LO1 Identify and justify good practices in object-oriented software development. Communicate the aspects of object-oriented programming which are advantageous when compared to non-object-oriented paradigms;

LO2 Acquired detailed knowledge of concepts of object-oriented programming and apply characteristics, tools and environments to adapt to new computational environments and programming languages which are based on object-oriented principles;

LO3 Design implement applications based on an object-oriented programming language, given a set of functional requirements. Use APIs which have not been exposed to previously, in order to develop an application requiring specialised functionality;

LO4 Design and Implement graphical interfaces using an object-oriented programming language;

LO5 Apply appropriate techniques for evaluation and testing and adapt the performance accordingly.

Course outcomes the module contributes to

BSc Computer Science	L5.1 L5.3 L5.4 L5.5 L5.7 L5.8
BEng Software Engineering	L5.1 L5.2 L5.3 L5.4 L5.5

Indicative syllabus content

- Object Oriented Programming principles and characteristics.
- Classes and Objects. The usage of APIs. Packages and namespaces.
- Polymorphism. Inheritance. Abstraction. Encapsulation. Access Specifiers. Abstract Classes and Interfaces. Overloading and overriding.
- How to design classes. UML design (UML class diagram and UML use case diagram)
- Graphical User Interfaces. Event Driven Design.
- Collections and generics. Data structures.
- Principles of concurrency and implementing threads.
- Program correctness: Defensive Programming (secure coding, exception handling, recovery), the concept of specification.
- Input/output and streams.
- Fundamental design patterns (examples could include but are not limited to: factory, MVC, singleton, adapter, proxy, facade, decorator).
- The role of algorithms in the problem-solving process. Performance Issues. Testing. Unit testing. Refactoring.

Teaching and learning methods

This module will be delivered in a mixture of presentations and lectures and supervised practical work. The taught material will be delivered in a 2-hour lecture and 2-hour tutorial mode for each teaching week. Technology-enhance learning, such us online quizzes and multiple-choice questionnaires will be used to promote students' participation and engagement during classes. Outside formal class time, there will be online support via Blackboard: lectures will be recorded and made available to students, solutions to code exercises will be provided, Blackboard forum and Padlet will be used to open discussions with students and reply questions. The content of the module is highly practical with

sufficient emphasis on problem solving skills. These skills will be developed through a number of appropriate tutorial exercises extending the relevant theoretical material introduced in the lectures.

Elements of self-study will also be included in the module. Students will be required to self- study certain additional aspects of object-oriented programming and produce appropriate software solutions related to the material studied.

Activity type	Category	Student learning and teaching hours*
Lecture	Scheduled	24
Tutorial	Scheduled	24
Total Scheduled		48
Independent study	Independent	152
Total student learning and teaching hours		200

*the hours per activity type are indicative and subject to change.

Assessment rationale

The summative assessment strategy involves an individual coursework that will require for the students to demonstrate the practical skills they will have acquired with regards to the development of software applications using object-oriented programming, and an examination that will assess the students' understanding related to analysis, design of system based on object-oriented principles and the more theoretical aspects of the module.

The individual coursework allows the students to demonstrate their ability to design and develop a software application using an OOP language, given a set a functional requirement. It will cover LO1, LO2, LO3 and LO4.

The in-class test allows the students to demonstrate their theoretical and practical knowledge and understanding of object-oriented concepts, object-oriented design and the basic principle of concurrency using an OOP language. It will cover LO1, LO2, LO3 and LO5.

Assessment criteria

In order to pass the module a student will have to demonstrate a detailed understanding of the design and development of applications based on Object Oriented Programming.

The criteria used to determine students' performance will depend on the degree to which they are able to:

- Design a system using OOP for given a set of specification
- Develop GUI and console applications that can handle user input and output based on object-oriented development principles

To achieve higher marks a student should demonstrate:

- the ability of a very good usage of OOP principles
- appropriate design decisions for the classes and their hierarchy
- efficient coding and organisation

- usage of polymorphism, error handling, robustness, flexibility and maintainability and where appropriately the quality and justification for the chosen algorithms implemented or library used.

Assessment methods and weightings

Assessment name	Weighting %	Qualifying mark %	Qualifying set	Assessment type (e.g. essay, presentation, open exam or closed exam)
Coursework	*50%*	*30%*		*Code and report*
In-class test	*50%*	*30%*		*Closed examination*

Synoptic assessment

This module draws upon analytical, design, technical and development skills learned elsewhere in the course. This module therefore engenders synoptic learning and its assessment is inherently synoptic.

Sources Essential reading

Horstmann, C. (2016) *Big Java*, 6th edition. Wiley.

Further reading

Shildt , H. (2014) *Java. A Beginner's Guide*,

6th Edition Eckel, B. (2006) *Thinking in Java*,

4th edition. Prentice Hall.

Bjarne, S. (2014), *Programming: Principles and Practice Using C++*, 2nd edition. Addison Wesley.

Weisfeld, M. (2013) *The Object Oriented Thought Process*, 4th edition. Addison-Wesley Professional

Patrick, HW, and Sundar, N. (2001) *On to Java*.

Addison Wesley. Langr, J. (2005) *Agile Java*,

Prentice Hall.

Robert, S and Wayne, K. (2011) *Algorithms*, 4th edition. Addison Wesley.

Gamma, E and Helm, R, and Johnson, R and Vlissides, J. (1994), *Design patterns: elements of reusable object-oriented software*. Addison Wesley.

King, KN. (2000) *Java Programming from the Beginning*. W.W. Norton

& Company. The Java Oracle site,
http://www.oracle.com/technetwork/java/index.html