Overview page

Object oriented programming
Level: Intermediate
Contact time: One evening a week for 10 weeks

Computer programs are used by most of us in our daily lives. Software is present in most modern devices in one form or another.

This module teaches how to write an object-oriented program. You will gain an understanding of classes, and how they enable code re-use. You will also implement a range of useful data structures.

PDF document information

Contact hours: Up to 2 hours per week for tutorials. Learners will also be required to watch lecture recording and carry out computer-based activities on their own PC.

Synopsis: Computer programs are used by most of us in our daily lives. Software is present in most modern devices in one form or another. This module teaches how to write an object-oriented program. You will gain an understanding of classes, and how they enable code re-use. You will also implement a range of useful data structures. You will learn about testing and error handling within your software. This module will use the Java programming language.

Notes: This module is aimed at professional learners who understand procedural programming and are looking to learn the basics of object-oriented programming. It is expected that learners will have experience of the indicative contents of the micro-credential “Introduction to imperative programming”.

Assessment: Learners will be assessed on their understanding through short programming exercises, a programming project, and a test.

Aim: This module aims to introduce learners to object-oriented programming.

Learning outcomes: Apply an object-oriented approach to software development; Make use of inheritance; Implement interfaces; Utilize facilities for generic programming; Test and debug software; Understand and use simple search and sort algorithms; Use a range of file handling techniques.

Syllabus: Classes, object, and encapsulation; Instance variables and method implementation; Public interfaces of a class; Constructors and overloading; Access and mutator methods; Reference types; Static variables and methods; Inheritance; Overriding methods; Data structures such as lists and queues; Compile and runtime errors; Error handling; Exceptions; System testing; File formats and file access.