SQA Advanced Unit Specification: general information

Unit title: Software Development: Object Oriented Programming

Unit code: HP2L 48

Superclass: CB

Publication date: August 2017

Source: Scottish Qualifications Authority

Version: 01

Unit purpose

This Unit is designed to enable candidates to develop a broad knowledge of the concepts, principles, and techniques of object oriented software development. Candidates will develop problem solving and object oriented technical skills. Candidates will then be required to demonstrate their proficiency in these skills through the creation of object oriented software solutions to problems. The emphasis is on the development and testing of the class libraries required for the problem domain. These will be reinforced by developing the appropriate practical skills in implementing and testing object libraries. It is recommended that this Unit is delivered in tandem with the Unit *Systems Development: Object Oriented Analysis and Design* (HP2M 48) to give candidates an insight into the full development lifecycle.

On completion of the Unit the candidate should be able to:

1 Investigate object oriented programming techniques and apply them to a design.
2 Implement a solution from an object oriented design using object oriented techniques.
3 Test the completed product.

Recommended prior knowledge and skills

Access to this Unit will be at the discretion of the Centre. However, it is recommended that candidates should have achieved the Core Skill of *Problem Solving* at SCQF level 6 before taking this Unit. It is also strongly recommended that candidates are familiar with fundamental programming concepts at SCQF level 7. This may be demonstrated by possession of one or more of the following SQA Advanced Units: *Developing Software: Introduction* (HP1R 47), *Software Development: Developing Small Scale Standalone Applications* (HP2N 47) or *Software Development: Programming Foundations* (HP2P 47).

Alternatively, candidates may have considerable practical work experience and a portfolio of programs which demonstrate their competence with the standard programming constructs of sequence, selection and iteration.

Credit points and level

2 SQA Credits at SCQF level 8: (16 SCQF credit points at SCQF level 8*)

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from National 1 to Doctorates.*

Core Skills

Opportunities to develop aspects of Core Skills are highlighted in the Support Notes of this Unit specification.

There is no automatic certification of Core Skills or Core Skill components in this Unit.

Context for delivery

If this Unit is delivered as part of a Group Award, it is recommended that it should be taught and assessed within the subject area of the Group Award to which it contributes.

Unit specification: statement of standards

Unit title: Software Development: Object Oriented Programming

Unit code: HP2L 48

The sections of the Unit stating the Outcomes, Knowledge and/or Skills, and Evidence Requirements are mandatory.

Where evidence for Outcomes is assessed on a sample basis, the whole of the content listed in the Knowledge and/or Skills section must be taught and available for assessment. Candidates should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Investigate object oriented programming techniques and apply them to a design.

Knowledge and/or Skills

- Object oriented concepts and terms
- Object oriented programming techniques
- Objects and classes
- Attributes and methods
- Parameter passing
- Abstraction, encapsulation and information-hiding
- Inheritance
- Polymorphism
- Association
- Aggregation and collection
- Coupling and cohesion

Outcome 2

Implement a solution from an object oriented design using object oriented techniques.

Knowledge and/or Skills

- Declaring and initialising variables
- Using operators
- Implementing control structures
- Defining data structures
- Accessing and manipulating data structures
- Using parameter passing
- Creating Classes
- Creating instances of classes
- Creating relationships between classes
- Creating Constructor methods
- Overloading methods
- Use of exceptions
- Use of standard object libraries
- Documenting code

Outcome 3

Test the completed product.

Knowledge and/or Skills

- Implementing a test plan using a defined strategy
- Maintaining test documentation
- Evaluating results of test runs
- Amending code as necessary

Evidence Requirements for the Unit

As an alternative to traditional assessment methods (eg paper-based), candidates can provide a digital record of evidence to demonstrate Knowledge and/or Skills. Suggested approaches are outlined in the Support Notes, Guidance on the assessment of this Unit.

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills by showing that they can investigate object oriented programming techniques and apply them appropriately to a design.

A candidate's response can be judged to be satisfactory where the evidence produced shows the candidate has successfully investigated and applied appropriate object oriented programming techniques.

Candidates will be required to implement a given Object Oriented Design. This must be of sufficient complexity to cover the knowledge and skills for each Outcome as outlined in the Support Notes, Guidance on the assessment of this Unit. The program must consist of at least four classes, and at least one 'one-to-many' association must be implemented. There must also be correct use of encapsulation and inheritance.

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills by showing that they can produce completed test documentation recording both the expected results of the test data and the actual results. The test data should be sufficient to adequately test the implemented solution in scope and range.

The candidate will be expected to record and evaluate the results of the test runs. Where there are discrepancies between the expected results and the actual results, the coding must be amended and corrected accordingly.

Assessment of these Outcomes should be conducted under open-book supervised conditions.

Assessors must assure themselves of the authenticity of each candidate's submission.

Unit specification: support notes

Unit title: Software Development: Object Oriented Programming

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

While the exact time allocated to this Unit is at the discretion of the centre, the notional design length is 80 hours.

Guidance on the content and context for this Unit

This Unit is intended as a two credit introduction to object oriented programming. Candidates will acquire knowledge of the concepts and principles of object oriented software development. Candidates will then be required to implement object oriented programming skills through the creation of object oriented solutions to problems.

In contrast to existing object oriented programming Units, this Unit is not an introductory programming Unit. It is an introduction to programming in an **object oriented** manner. However, this Unit assumes prior knowledge of programming fundamentals. In addition, there is less of a focus on documentation than in other programming Units. This Unit is designed to be delivered in conjunction with *Systems Development: Object Oriented Analysis and Design* (HP2M 48), which adequately assesses the ability to produce design documentation. It is also possible to incorporate some aspects of event driven programming. Object oriented and event driven programming are not mutually exclusive, and almost all object oriented languages can be used in an event driven way. A program can both be structured in an object oriented manner and enable user interaction through an event driven interface. An understanding of basic event driven techniques is therefore required for designing programs which have a graphical user interface

The intention is for this Unit to act as a follow-on to *Developing Software: Introduction* (HP1R 47). Candidates should already be familiar with the basic programming building blocks, allowing this Unit to focus on object oriented programming concepts such as classes and inheritance. The Unit could also focus on creating graphical programs rather than console-based programs, enabling the candidates to create simple GUI applications to match Unit purpose. The teaching and learning approaches may include problem-based learning, teamwork, and potentially an element of competition (eg ‘solve in an hour’ code challenges) to engage candidates and encourage self-directed study.

This Unit forms part of the SQA Advanced Diploma in Computing: Software Development Group Award and should be delivered within the context of the Group Award. It would be suitable for candidates who are proposing to follow a career in systems design or software development. It is not suitable as a stand-alone Unit, but could be taken after successful completion of *Developing Software: Introduction* (HP1R 47), or fit into the second year of the SQA Advanced Diploma in Computing: Software Development structure.

As a non-introductory programming Unit, the context and examples used can be relatively complex in terms of the scope and programming fundamentals required. However, they should be relatively simple in terms of object orientation. Using examples of relatively simple games (for example noughts and crosses, battleships, pac-man, pong, space invaders) would hopefully make the learning process more enjoyable and meaningful for candidates, and would permit the candidate to concentrate more on the programming techniques involved, rather than trying to understand the initial problem.

By the end of the Unit, the candidate should have achieved a good foundation in the skills required for developing reliable, robust and efficient object oriented program designs for simple applications.

It is likely that the Unit will be delivered in the second year of a full-time SQA Advanced Diploma in Computing: Software Development Group Award.

The Unit is capable of being delivered on its own, but is usually expected to be delivered in conjunction with *Systems Development: Object Oriented Analysis and Design* (HP2M 48).

This Unit covers some of the skills described for a pre-entry/Junior technician role in the National Occupational Standards — IT and Telecoms (2009). The main areas covered correspond to discipline 5.2 Software Development and discipline 5.3 IT/ Technology Solution Testing. There are also ample opportunities within the Unit to address a range of skills at both foundation and intermediate level that are described in the National Occupational Standards for IT Users v3. The most likely areas to be covered would be using the Internet and IT Software Fundamentals.

Guidance on the delivery of this Unit

Outcomes 1–3

1 Investigate object oriented programming techniques and apply them to a design.
2 Implement a solution from an object oriented design using object oriented techniques.
3 Test the completed product.

This Unit is designed as a follow-on to *Developing Software: Introduction* (HP1R 47). As such, it assumes knowledge of fundamental programming concepts, but it would be advisable to briefly refresh these basic topics (variables, operators, iteration, selection, arrays) and show how to implement these using the syntax of the object oriented programming language you have chosen to use. It is recommended that no more than eight hours should be allocated to these topics, leaving the vast majority of the time to be allocated to object oriented programming techniques. The following topics should be covered:

- Defining data structures
- Accessing and manipulating data structures
- Using parameter passing
- Creating Classes
- Creating instances of classes
- Creating relationships between classes
- Creating Constructor methods
- Overloading methods
- Use of exceptions
- Use of standard object libraries
- Use of pre-defined interface components
- Documenting code

The implementation language chosen is not specified. At the time of writing, the most appropriate languages to use would be Java or C#. Both these languages are widely used in industry and have a similar, C-type syntax. Java is a good teaching language with a large object library, and the creation of user interfaces is supported through AWT and swing. C# allows for the creation of interfaces through simple drag and drop, and the XNA development kit can be used in association with C# for the development of fairly complex computer games, if centres wished to enable candidates to develop games. Candidates should be introduced to the standard object libraries and shown how to find what is available (eg via the Sun Java Docs for Java, or the MSDN libraries for C# and C++) and how to make use of appropriate standard objects (eg via import or using statements).

It might well be beneficial to introduce candidates to particular technical skills for event driven programming, dependent upon the implementation language chosen. For example, if Java is used then candidates will need to be taught about the awt and swing packages, whereas if C# is used the candidates could be taught how to set up user interfaces through code, or be introduced to the toolbox of user interface components and their various properties. The GUI libraries supplied with modern OO languages are an excellent way of introducing the idea of classes and objects. The hierarchies can aid in the understanding of inheritance and encapsulation. In addition, the interactive event methods can be used to introduce the concept of events in software development.

In order to understand practical object oriented programming techniques and structures, there are several theoretical object oriented concepts which candidates should understand. The following topics should be covered:

- Objects and classes
- Attributes and methods
- Parameter passing
- Abstraction, encapsulation and information-hiding
- Inheritance
- Polymorphism
- Association
- Aggregation
- Coupling
- Cohesion

It is recommended that theory and practice are not separated but are intertwined as much as possible. Candidates should be given several practical exercises for each topic as programming is a skill which is learnt and improved upon through practice. It is recommended that the practical exercises increase in complexity throughout the course of the Unit as new techniques are learnt. The practice of providing examples which show the new technique or skill in isolation is useful for initial understanding of how a technique may be implemented, but is not useful for the problem solving and program design process. Candidates should begin learning how to incorporate new techniques into larger programs from an early stage in the Unit so that there is suitable opportunity for integration of concepts.

Internal documentation of code should be carried out using current commercial standards. At the time of writing, this could include using basic JavaDoc for Java, or basic XML documentation for C#. Naming conventions, indentation, and version control should also be stressed when discussing code structure and documentation.

Developing the ability to analyse a project design brief and implement a suitable object oriented solution is a skill that is acquired through practice rather than one that can be explicitly taught. This Outcome should be embedded throughout the Unit, and will be acquired through continual practice of the problem solving and program design process at each stage of Unit. Although there is no way to easily teach this skill, there are a variety of helpful techniques which can be taught. The object oriented software development methodology and Unified Modelling Language should be introduced, if they are not being covered in another Unit on the course. It would also be beneficial to introduce candidates to Case-Based Reasoning. This is the technique of looking back at previous problems and their solutions and using similar cases as the basis for the design of an implementation for the new problem. For example, candidates could be shown the problem of writing a noughts and crosses game, and as a class the solution to this problem could be designed and implemented. Candidates could then be asked to create a Connect 4 game, using the noughts and crosses game as a basis for their design due to the many similarities between the two games.

The GUI libraries supplied with modern OO languages are an excellent way of introducing the idea of classes and objects. If this approach is adopted, it would also be beneficial to introduce candidates to HCI design techniques such as storyboarding. This could alternatively be delivered via other Units in the framework, such as*: Human Computer Interaction* (HR8C 47), which could run concurrently.

Guidance on the assessment of this Unit

This Unit should be assessed by means of a single project covering all three Outcomes. Ideally, the assessment project would involve producing an application or a library. Two or more projects should be offered to give candidates a degree of choice. The Evidence Requirements can be met through one large program that covers all knowledge and skills, or via a portfolio of two or more smaller programs.

Assessment Guidelines

Outcome 1

In order to assess Outcome 1, it is recommended that the program demonstrates the following:

- Abstraction, Encapsulation and information-hiding used where appropriate.
- Inheritance used if appropriate to the solution.
- Polymorphism used only if appropriate to the solution.
- All class-wide variables are private to prevent content coupling.
- Class-wide variables are kept to a minimum to ensure a minimum of common coupling.
- Data coupling is used (using parameter passing) in preference to content or common coupling.
- Program does not contain a lot of unnecessary data coupling.
- Classes are highly cohesive.

This Outcome will be assessed along with Outcomes 2 and 3 in a single project. The development of a program which contains a sampled selection of object oriented programming techniques will provide evidence of an understanding of the concepts covered in this learning Outcome.

It is recommended that the project has a minimum of two deadlines: one for the program implementation and one for testing the implemented solution, to stagger the assessment load and allow for feedback on the implementation before the final testing is completed.

It is recommended that candidates are given the option to choose from a selection of projects. Suitable projects would include 2D puzzle games such as Sokoban or Tetris-style games. Non-game-related projects such as an address book application would also be suitable projects. You may also allow the candidate to suggest their own project, as long as it will be sufficiently complex to meet all Evidence Requirements, whilst not being too complex to implement in the time allocated. The project specifications should allow the candidates enough flexibility to customise their program. The project should not be too regimented in terms of physical appearance and sound features, but must be precise in its operational requirements. For example, the appearance of the interface should not be specified. However, the manner in which the user of the program must be able to interact with it, such as keyboard input, and the functionality of the program, must be clear.

Outcome 2

This Outcome will be assessed along with Outcomes 1 and 3 in a single project.

In order to assess Outcome 2, it is recommended that the program demonstrates the following:

- Implement a working solution which meets the requirements of the given brief.
- Declare and initialise variables.
- Correctly use arithmetic and/ or logical operators.
- Implement a range of control structures.
- Access and manipulate a data structure.
- Create a minimum of four classes, which contain attributes, methods, and a constructor method.
- Create a minimum of three objects from the classes, with appropriate initial attribute values set through the Constructor methods.
- Implement at least one overloaded method (this may be the constructor).
- Link the classes appropriately through association, aggregation, or inheritance relationships.
- Pass parameters correctly both within and between objects.
- Define the access type (public, private, or protected) for methods, attributes and classes, and the access modifiers chosen are appropriate.
- Make use of pre-defined Classes and/ or Methods from the standard object library.
- Appropriately handle errors with exceptions or pre-validation.
- Implement code commented appropriately throughout.

It is recommended that polymorphism is assessed, either in the project or in a separate program.

Outcome 3

This Outcome will be assessed along with Outcomes 1 and 2 in a single project.

It is recommended that a test plan be provided for the candidate. Alternately, the test plan could be developed as a group classroom activity. This would let candidates see how to set about developing a test plan, which would help them when it comes to the Graded Unit project.

Online and Distance Learning

If this Unit is delivered by open or distance learning methods, additional planning and resources may be required for candidate support, assessment and quality assurance. A combination of new and traditional authentication tools may have to be devised for assessment and re-assessment purposes.

Opportunities for the use of e-assessment

E-assessment may be appropriate for some assessments in this Unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all candidate evidence and that conditions of assessment as specified in the Evidence Requirements are met, regardless of the mode of gathering evidence. Further advice is available in *SQA Guidelines on Online Assessment for Further Education (AA1641, March 2003).*

Opportunities for developing Core Skills

There may be opportunities to gather evidence towards Core Skills in this Unit, although there is no automatic certification of Core Skills or Core Skills components.

Candidates develop the *Problem Solving* Core Skill at SCQF level 6 through identifying how to implement the software solution based on a design which does not provide detailed instructions on how to program a solution. The candidates could be required to write a short report detailing their problem solving process, ie how they identified the best programming structures to use for the major program functionality, in order to fulfil this requirement.

Equality and inclusion

This unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website www.sqa.org.uk/assessmentarrangements.

History of changes to Unit

Version	Description of change	Date

SQA acknowledges the valuable contribution that Scotland's colleges have made to the development of SQA Advanced Qualifications.

FURTHER INFORMATION: Call SQA's Customer Contact Centre on 44 (0) 141 500 5030 or 0345 279 1000. Alternatively, complete our Centre Feedback Form.

General information for candidates

Unit title: Software Development: Object Oriented Programming

This Unit is designed to cover the object oriented programming skills required for a career in software development. It is a non-introductory Unit and assumes prior knowledge and proficiency in basic programming concepts and techniques. *Developing Software: Introduction* (HP1R 47) is a recommended pre-requisite for this Unit.

In this Unit you will acquire knowledge of the concepts, principles, and techniques of object oriented software development necessary to enable you to design and develop object oriented software.

This will involve the following areas of learning:

- Using the features of an object oriented programming language, you will implement a software solution based on a given design. Your understanding and grasp of object oriented concepts and programming techniques will be reinforced throughout with practical exercises.
- Using a test plan, you will test your software to ensure it works correctly and meets the user requirements. You will be required to amend any errors in your code in order to achieve a robust, reliable and efficient working program.

In completing the above you will have achieved the following Outcomes within this Unit:

1 Investigate object oriented programming techniques and apply them to a design.
2 Implement a solution from an object oriented design using object oriented techniques.
3 Test the completed product.