Module Information

Module Identifier	**CS12320**
Module Title	**Programming Using an Object-Oriented Language**
Academic Year	**2014/2015**
Co-ordinator	**Mr Christopher William Loftus (mailto:cwl@aber.ac.uk?subject=CS12320)**
Semester	**Semester 2**
Pre-Requisite	**CS12020 (?m=CS12020)**
Other Staff	**Miss Helen Charlotte Miles (mailto:hem23@aber.ac.uk?subject=CS12320)** **Dr Harry George Strange (mailto:hgs08@aber.ac.uk?subject=CS12320)** **Mr Christopher William Loftus (mailto:cwl@aber.ac.uk?subject=CS12320)** **Mr Alassane Seck (mailto:als31@aber.ac.uk?subject=CS12320)** **Dr Lynda Ann Thomas (mailto:ltt@aber.ac.uk?subject=CS12320)** **Dr Wayne Aubrey (mailto:waa2@aber.ac.uk?subject=CS12320)** **Maximilian John Walker (mailto:mjw9@aber.ac.uk?subject=CS12320)** **Mr Michael Francis Clarke (mailto:mfc1@aber.ac.uk?subject=CS12320)**

Course Delivery

Delivery Type	Delivery length / details
Other	20 x 2hr workshops that include lecture content interspersed with practical material
Seminars / Tutorials	10 x 1hr tutorials focused on programming and design

Assessment

Assessment Type	Assessment length / details	Proportion
Semester Assessment	Individual assignment	50%
Semester Assessment	Up to 5 practical mini-assignments	40%
Semester Assessment	Active participation in tutorials	10%
Supplementary Assessment	One assignment	100%

Learning Outcomes

On successful completion of this module students should be able to:

1. Describe and explain the key differences between procedural and object oriented programming.

2. Find objects, classes and methods based on a problem statement, thereby showing aptitude to apply the concepts of abstraction and encapsulation.

3. Map UML use-case, class and sequence diagrams onto object-oriented code.

4. Develop a non-trivial object-oriented program that contains a graphical user interface, thereby demonstrating the ability to deal with simple event-driven programming.

5. Demonstrate the ability to apply the concepts of composition, inheritance and polymorphism.

6. Demonstrate in object-oriented code how to handle error conditions.

7. Demonstrate in object-oriented code how to store and retrieve data to and from files.

Aims

This module will build on CS12020 Introduction to Programming. It will explore the use of the object-oriented paradigm and its embodiment in the Java programming language. It will be taught in conjunction with CS10720 – Problems and Solutions, and will define and use UML when modeling requirements and design.

Brief description

This module will build on CS12020 Introduction to programming. In particular, it will explore the use of the object-oriented paradigm and its embodiment in the Java programming language. UML (Unified Modeling Language) notation will be defined and used as appropriate. It provides a foundation for Part 2 modules that use object-oriented languages, such as CS21120 - Data Structures and Algorithms, and CS22120 - The Software Development Life Cycle.

Content

This module's teaching pattern each week consists of two 2-hour lecturer-led teaching sessions in a large computer laboratory. This enables small presentations to be followed by related practical exercises and quizzes. Students also have a 1-hour small-group tutorial for team-based design and coding exercises.
Week 1 - Introductory workshops as a taster of many of the topics to be covered during the module: The idea of class and object. Storing data in instance variables. Methods. Java Virtual Machine and bytecode. Applications running from an integrated development environment tool. Review of concepts from semester one as used in Java: variables, conditional tests, loops.
Week 2 - Basic concepts. Exploration of objects and classes. The UML class diagram. Instance variables, methods and parameters, object diagrams. Relationships between classes and their representation in class diagrams. Mapping a simple procedural program from semester one to a Java program.
Week 3 - Review of basic concepts. Reading from the keyboard. Null references. Running programs from the command line. Javadoc comments. Naming conventions. Tutorial on the use of the Classes, Responsibilities and Collaborations technique.
Week 4 - Types and equality. Searching, loading and saving. Reading from and writing to files. Iteration over Java Collections. Java arrays. UML sequence diagrams.

Week 5 - Access modifiers. Packages and JAR files. The static modifier. Revisiting abstraction and encapsulation. Consolidation: design and implementation. The role of use-case diagrams and their relationship to class diagrams and implementation.
Weeks 7 and 8 - Focus on inheritance, polymorphism, interfaces and abstract classes. Overriding the equals method. Exception classes.
Weeks 9 and 10 - Graphical user interfaces. Event-driven programming. Separation of concerns.
Week 11 - review

Module Skills

Skills Type	Skills details
Application of Number	Inherent in the subject
Communication	Communication in a technical sense through UML diagrams
Improving own Learning and Performance	From feedback from staff and fellow students through peer assessment
Information Technology	Inherent in the subject
Personal Development and Career planning	The module will provide more information on what software engineers do
Problem solving	Solving design and coding problems
Research skills	Basic computer use
Subject Specific Skills	UML diagrams, code development skills, use of integrated development environments
Team work	Developed in tutorials

Reading List

Recommended Text
Sierra, Kathy. (2005.) *Head first Java /Kathy Sierra, Bert Bates.* 2nd ed. O'Reilly Primo search (http://primo.aber.ac.uk/primo_library/libweb/action/search.do?vl%28freeText0%29=Head+first+Java+%2FKathy+Sierra%2C+Bert+Bates.+Sierra%2C+Kathy.&fn=search&vid=ABERU_VU1)

Notes

This module is at CQFW (http://wales.gov.uk/topics/educationandskills/qualificationsinwales/creditqualificationsframework/?lang=en) Level 4