

**MODULE****OBJECT ORIENTED DESIGN &  
IMPLEMENTATION****CODE**

BSCH-2-2-09

**STAGE**

II

**NUMBER OF CREDITS**

4 semester credits / 6 quarter units

**STATUS**

CORE

**THEMES**

Software Development, Business Solutions &amp; Design

**ASSESSMENT**

Continuous Assessment	40%
Examination	60%

**Aims**

This module aims to introduce the fundamental concepts of object oriented program design and how to use the Object Modelling paradigm for constructing software systems from requirement specifications.

**Learning Outcomes**

Upon completion of this module a student should be able to:

- Develop the analytical skills necessary to apply abstract concepts in an object oriented manner.
- Express system solutions in a formal manner and implement the derived formalisation.
- An ability to select and apply appropriate design techniques to system solutions.
- Develop confidence in and awareness of the capabilities of object oriented development.
- Demonstrate an ability to abstract problem specifications and program design by producing good software designs.
- An ability to analyse a problem, produce high quality software designs using Universal Modelling Language (UML) notation and relate the software designs to the implementation.
- Identify problems associated with traditional methods of software specification, and explain how formal methods overcome these problems.
- Develop high quality software that is reliable, reusable and maintainable.

## Indicative Content

Topic	Description
<b>The Object Paradigm</b>	Classification: Objects and Object Types (Classes); Abstraction; Encapsulation: Data and Behaviour; Information Hiding: Access Specifiers; Inheritance and Polymorphism; Aggregation and Association; Software Reuse;
<b>Unified Modelling Language</b>	Rationale and history of UML; Use Case Analysis; Structural View: Class and Object diagrams; Behaviour View: Sequence, Collaboration, Statechart, Activity diagrams; Environment View;
<b>Methods for Object Oriented Analysis</b>	Object Behaviour Analysis Object Modelling Techniques Information Modelling Classes, Associations and Attributes Object Relationship Diagrams Dynamic Modelling States, Events Transitions Scenarios and Events Traces State Transition diagrams Functional Modelling Activity Diagrams
<b>Object Oriented Design</b>	Implementation Options; Object Oriented Methodologies; Use of iterative development; Introduction to Patterns and Frameworks;
<b>Object Oriented Programming</b>	Implementation of classes and objects; Static and Dynamic Objects; Testing and debugging in Java; Use of commercial libraries; Sample programs;
<b>Evaluation of the OOD approach</b>	The management perspective Pros and cons of the OO approach Changeover methods and difficulties Suitable application categories Current Issues Standardisation: OMG, COBRA, etc. OO-DBMS v RDBMS OO – OS



## Teaching and Learning Methods

This Module will be taught using a combination of lectures, demonstrations and tutorials, practical work using Object Oriented languages, such as Java, library and other research, study of texts and handouts and interactive teaching and a case study.

## Assessment Methods

Assessment will use both a continuous component and an end of semester examination. The continuous assessment component is used to develop practical skills of programming techniques and will be based both on the lab workbooks and graded assignments / in class tests. Students will be expected to develop efficient, well-documented code, meeting accepted quality standards.

## Primary Reading List

<b>Title</b>	<b>Author</b>	<b>Publisher</b>
Systems Analysis and Design Methods – fifth edition	Jeffrey L. Whitten, Lonnie D. Bently, Kevin C. Dittman	McGraw Hill, 2001
Information Systems Development: Methodologies, Techniques, and Tools	Avison and Fitzgearld	McGraw Hill, 2002

## Recommended Reading List

<b>Title</b>	<b>Author</b>	<b>Publisher</b>
Software Engineering – Theory and Practice	Shari, Lawrence and Pfleeger	Prentice Hall, 1999
An Introduction to Systems Analysis techniques	M. Lejk & David Deeks	Prentice Hall, 2000
Software Engineering	Sommerville. I	Addison-Wesley, 1998