WebLearning Java Curriculum
**C++ Programming: Classes and Data Abstraction**

**Preceding Course:** C++ Programming: Structured Programming

**Following Course:** C++ Programming: Manipulating Objects

**Course Duration:** 4 Hours

**Audience:** Users with a knowledge of structured programming techniques

**Prerequisites:** Users should have some experience of C or another structured programming language

**Course Aim:** To provide users with the skills required to create and instantiate classes

**Learning Objectives:**

After taking this course, the user should be able to

- outline the elements of a class
- describe how class members can be accessed
- define and instantiate a class
- explain the use of inheritance in C++

**Course Incorporates:** Test, hands-on exercises
Object-Oriented Principles

Produced by WebLearning Courseware.

Course duration: 12 hours

Audience: Programmers

Prerequisites: Users should be programmers.

Course aim: To provide the user with a sound understanding of object-oriented principles

Learning objectives
After taking this course, the user should be able to

• explain the present demand for OOP
• list both the aims and the foundations of OOP
• describe what is meant by "data abstraction", "encapsulation", and "data hiding"
• describe how "classes" in OOP implement data abstraction
• explain how methods and messages provide for communication between classes while preserving the principles of data abstraction
• assess the advantages of inheritance
• assess the advantages of polymorphism and dynamic binding
• list the problems with traditional, procedural languages
• list the main promises of OOP and identify the new demands that it places on the designer/programmer
• list and describe some object-oriented languages and tools for development
• implement the development process

Topics covered

Ideals of programming
Changing software trends
Aims of OOP
Foundations of OOP: data abstraction; inheritance; polymorphism and dynamic binding
Classes and data abstraction
Classes and instances
Public interface
Methods and messages
Constructors and destructors
Life-cycle of an object
Collections of methods
Access to methods
Inheritance
Inheritance trees/hierarchies
Multiple inheritance
Advantages of inheritance

Inheritance v containment
Combining inheritance and containment
Polymorphism and dynamic binding
Benefits of polymorphism and dynamic binding
Problems with procedural languages
What OOP promises
Demands OOP places on designer/programmer
OOP languages
OOP tools for development
Iterative design cycle
Project definition
Project analysis
Project design
Project implementation
Project design exercise

DOS version available

Course incorporates: Test, Glossary
Object-Oriented Analysis - Objects and Classes

Produced by WebLearning Courseware.

Preceding course: Principles of OOP II
Following course: Object-Oriented Analysis - Dynamic Modeling

Course duration: 5 hours

Audience: Systems analysts

Prerequisites: Some experience of conventional systems analysis

Course aim: To provide the student with some knowledge of and experience in using object-oriented methods in the systems analysis process

Learning objectives
After taking this course, the student should be able to

- point out the benefits of OOAD by comparing other models with object-oriented models
- contrast conventional methods with OO methods
- draw up a list of objects in a problem domain and write object descriptions for them
- identify classes and class responsibilities
- identify class methods and attributes
- create an object-messaging diagram
- distinguish the three major types of class relationships (generalization/specialization relationships, aggregate relationships, association relationships)
- explain the concepts of inheritance and containment and identify classes that inherit from each other
- draw up a class hierarchy diagram

Topics covered
Software development models
Objects
Functions and objects
Challenging candidate objects
Object descriptions
Class hierarchies
Class responsibilities
Methods
Class attributes
Object diagrams
Messages
Class relationships
Generalization/specialization
Inheritance
Aggregate relationships

Association
Object schemas
Windows version available

Course incorporates: Test
Object-Oriented Analysis - Dynamic Modeling

Produced by WebLearning Courseware.

**Preceding course:**  *Object-Oriented Analysis - Objects and Classes*

**Following course:**  *Object-Oriented Design*

**Course duration:**  6 hours

**Audience:**  Experienced systems analysts

**Prerequisites:**  Experience of procedural methods of analysis and design; the course *Object-Oriented Analysis - Classes and Objects*

**Course aim:**  To familiarize the student with dynamic modeling diagramming techniques

**Learning objectives**

After taking this course, the student should be able to

- describe the terms *state*, *event*, and *operation* and create a state-transition diagram
- differentiate between control conditions, triggers and clocks and show how they can be represented on an event diagram
- define a process and identify one from a text description
- create a process-dependency diagram
- explain what an object-flow diagram is and how to create one

**Topics covered**

- States, events and operations
- State-transition diagram notation
- Creating a state-transition diagram
- Control conditions, triggers and clocks
- Event diagram notation
- Creating an event diagram
- Identifying processes
- Process-dependency notation
- Creating an object-flow diagram

**Windows version available**

**Course incorporates: Test, Exercise topics**
Object-Oriented Design

Produced by WebLearning Courseware.

Preceding course:  Object-Oriented Analysis - Dynamic Modeling
Following course:  Several courses possible. See your curriculum planner.

Course duration:  3 hours
Audience:  Systems designers
Prerequisites:  Some experience of conventional systems design
Course aim:  To provide the student with some knowledge of and experience in using object-oriented methods in the systems design process, with particular reference to database and GUI applications

Learning objectives
After taking this course, the student should be able to

• develop and modify object hierarchies
• document a design so as to facilitate subsequent users
• validate a design against system specifications
• apply the theoretical concepts of OO design in real-world
database management situations
• design consistent, extensible, and flexible GUIs

Topics covered
Refining the object model
Developing objects
Inheritance and delegation
Modifying the object model
Data protection
Documenting the design
Design changes
Validation
Good design criteria
Object-oriented database management
OO/DBMS coupling
Pure OODBMSs
Object store
GUI design
ObjectWindow
NextStep

Windows version available
Course incorporates: Test
**OOAD with the UML: Rational Rose 98/2003 - Enhanced Features**

Course duration: 4 Hours  

**Audience:** Analysts, software engineers, application experts, and technical project managers using Rational Rose 98/2003 with the UML  

**Prerequisites:** Previous courses in the OOAD with the UML curriculum; a firm understanding of the UML and object-oriented analysis and design principles  

**Course aim:** To show some enhanced features of the Rational Rose 98/2003 tool, including team development  

**Learning objectives:**  
After taking this course, the student should be able to create, update, and save state transition diagrams, manipulate packages, create, update, and save component and deployment diagrams, understand the principles of team development in Rational Rose 98/2003, list and describe some of the enhanced features of Rational Rose 98/2003  

**Course incorporates:** Test, hands-on exercises
**OOAD with the UML: Analyzing the System**

Course duration: 4 Hours

**Audience:** OOD Analysts and designers

**Prerequisites:** Previous courses in the curriculum OOAD with the UML; a good understanding of object-oriented principles and the role of use-case and object models

**Course aim:** To show how to specify object interaction in the UML and to identify associations and inheritance between classes

**Learning objectives:**
- After taking this course, the student should be able to describe how to discover object interaction
- understand how to interpret sequence and collaboration diagrams in the UML
- describe the types of relationship that can exist between classes
- add names, roles, and navigation to associations in class diagram
- add multiplicity
- describe how to discover inheritance relationships between classes
- distinguish between inheritance and aggregation

**Course incorporates:** Test, hands-on exercises
**OOAD with the UML: Design and Implementation Issues**

Course duration: 4 Hours

**Audience:** OOAD analysts, designers, and programmers; system architects

**Prerequisites:** Previous courses in the curriculum OOAD with the UML; a good understanding of OOAD; a knowledge of OO programming considerations

**Course aim:** To describe the main principles of designing and implementing an object-oriented model

**Learning objectives:**
- After taking this course, the student should be able to list the characteristics of well-designed classes
- discuss the issues relating to designing relationships, attributes, operations, and inheritance
- specify attribute and operation design details in the UML
- design navigation for class associations

Course incorporates: Test
**OOAD with the UML: Designing the System**

Course duration: 4 Hours

**Audience:** OOAD analysts and designers

**Prerequisites:** Previous courses in the curriculum OOAD with the UML

**Course aim:** To introduce architectural analysis and design and the role of key mechanisms

**Learning objectives:**
- After taking this course, the student should be able to discuss the importance of a system architecture
- list the elements of the 4+1 architecture model
- use component and deployment diagrams
- discuss the importance of key mechanisms

Course incorporates: Test
**OOAD with the UML: Exploring System Behavior**

**Course duration:** 4 Hours

**Audience:** Analysts and designers using OOAD for the first time; software engineers; anyone involved with capturing requirements for large software systems

**Prerequisites:** The course OOAD with the UML: Fundamentals; an understanding of basic software life-cycle principles

**Course aim:** To explain how to capture a system's requirements with use cases

**Learning objectives:**
After taking this course, the student should be able to
- explain what a use case is
- explain what an actor is
- describe the process of exploring system behavior through the systematic identification of use cases and actors
- outline the purpose of problem statements
- illustrate use cases and actors in use-case models using UML notation
- explain how you generate a flow of events from a use case

**Course incorporates:** Test, hands-on exercises
**OOAD with the UML: Finding Classes**

Course duration: 4 Hours

**Audience:** Analysts and designers wishing to use OOAD for the first time

**Prerequisites:** Previous courses in the curriculum OOAD with the UML

**Course aim:** To outline techniques for finding classes from an initial set of system requirements

**Learning objectives:**
- After taking this course, the student should be able to distinguish between objects and classes
- list the characteristics of a good class
- identify candidate classes from a use-case flow of events
- describe boundary, entity, and control class stereotypes
- group classes into boundary, entity, or control stereotypes
- draw simple class diagrams in the UML
- use the Class-Responsibilities-Collaborators (CRC) card technique to refine class information

**Course incorporates:** Test, hands-on exercises
**Course duration:** 4 Hours

**Audience:** Analysts and designers interested in using an object-oriented process to design better software; analysts, programmers, and software managers interested in the UML

**Prerequisites:** Knowledge of object-oriented programming concepts, such as encapsulation and inheritance, and of software engineering principles, while not essential, would be an advantage

**Course aim:** To introduce basic OOAD principles, the Rational Objectory Process, and the role of the UML

**Learning objectives:**  
After taking this course, the student should be able to  
distinguish between analysis and design  
explain the importance of having a software life-cycle process  
list the advantages of using object orientation  
describe the role of the UML in analysis and design  
list the phases and process components of the Rational Objectory Process

**Course incorporates:** Test, hands-on exercises
**Course duration:** 4 Hours  

**Audience:** Analysts, software engineers, application experts, and technical project managers using Rational Rose 98 with the UML  

**Prerequisites:** Previous courses in the OOAD with the UML curriculum; a firm understanding of the UML and object-oriented analysis and design principles  

**Course aim:** To show the main features of Rational Rose 98 and how to create and maintain use-case, class, and interaction diagrams  

**Learning objectives:**  
After taking this course, the student should be able to  
- list the main features of the Rational Rose 98 tool  
- use the user-interface, and set options  
- create, update, and save use-case diagrams  
- create, update, and save class diagrams  
- add operations and attribute details to class diagrams  
- create, update, and save interaction diagrams  

**Course incorporates:** Test, hands-on exercises
**Course duration**: 4 Hours  
**Audience**: OOAD analysts and designers; OO programmers needing to read and interpret diagrams in the UML  
**Prerequisites**: Previous courses in the curriculum OOAD with the UML  
**Course aim**: To introduce operations and attributes, state and activity diagrams, and to show the importance of a review of the class model  
**Learning objectives**:  
After taking this course, the student should be able to  
- find and document operations and attributes for classes  
- describe the purpose of state and activity diagrams  
- read and interpret state diagrams in the UML  
- read and interpret activity diagrams in the UML  
- describe how a model can be reviewed for consistency and quality  

**Course incorporates**: Test, hands-on exercises
**Principles of OOP: I**

**Following course:** Principles of OOP: II  
Course duration: 4 Hours

**Audience:** Programmers

**Pre-requisites:** Some programming experience

**Course aim:** To introduce the user to object-oriented principles

**Learning objectives:**
After taking this course, the user should be able to
- explain the present demand for OOP
- list the aims and the foundations of OOP
- describe what is meant by "data abstraction", "encapsulation", and "data hiding"
- describe how "classes" in OOP implement data abstraction
- explain how methods and messages provide for communication between classes while preserving the principles of data abstraction

Course incorporates: Test
Principles of OOP: II

Preceding course: Principles of OOP: I 

**Following course:** Object-Oriented Analysis - Classes and Objects

Course duration: 4 Hours

**Audience:** Programmers

**Pre-requisites:** Some programming experience

**Course aim:** To provide the user with a sound understanding of object-oriented principles

Learning objectives:

- After taking this course, the user should be able to access the advantages of inheritance
- access the advantages of polymorphism and dynamic binding
- list the problems with traditional, procedural languages
- list the main promises of OOP and identify the demands that it places on the designer/programmer
- list and describe some object-oriented languages and tools for the development
- implement the development process

Course incorporates: Test
Microsoft Windows Architecture: Database Technologies
Course duration: 4 Hours
**Audience:** IS managers, system solution developers, and programmers with no prior Windows programming experience
**Prerequisites:** Familiarity with the Windows environment
**Course aim:** To examine the technologies available for creating database solutions
**Learning objectives:**
- After taking this course, the student should be able to explain how databases interact, share data, and keep data up to date
- describe the features and components of Microsoft's database products
- explain how Open Database Connectivity (ODBC) enhances interactions with databases
- introduce Microsoft's OLE database technology

Course incorporates: Test
Microsoft Solution Architectures: Conceptual and Logical Design

Course duration: 4 Hours

**Audience:** A wide range of information technology (IT) professionals, including IT managers, system analysts, developers, consultants, and those wishing to learn more about designing solutions

**Prerequisites:** Familiarity with analysis, design, and programming techniques

**Course aim:** To explain the process of conceptual and logical design and consider the technical architecture choices

Learning objectives:
- After taking this course, the user should be able to
  - describe the conceptual design phase and the process involved
  - describe the logical design phase and the process involved
  - explain how to define the technical architecture
  - outline the database design process

Units in Microsoft Solution Architectures: Conceptual and Logical Design:
- Conceptual design
- Defining the technical architecture
- Logical design
- The database life cycle

**Course incorporates:** Test, hands-on exercises
Microsoft Solution Architectures: Physical Design and User Interface

Course duration: 4 Hours

**Audience:** A wide range of information technology (IT) professionals, including IT managers, system analysts, developers, consultants, and those wishing to learn more about designing solutions

**Prerequisites:** Familiarity with analysis, design, and programming techniques

**Course aim:** To describe the physical design process, examine the issues surrounding user interface design, and consider methods of providing user Help

**Learning objectives:**
After taking this course, the user should be able to
- describe the physical design phase
- identify various GUI controls
- choose the appropriate control for a given task
- explain how to design user interfaces according to Microsoft Windows standards

**Units in Microsoft Solution Architectures: Physical Design and User Interface:**
- Physical design
- Implementing the data tier
- Implementing the business tier
- Implementing the presentation tier

**Course incorporates:** Test, hands-on exercises
**Microsoft Solution Architectures: Analysis and Design Techniques**

Course duration: 4 Hours

**Audience:** A wide range of information technology (IT) professionals, including IT managers, system analysts, developers, consultants, and those wishing to learn more about designing solutions

**Prerequisites:** Familiarity with analysis, design, and programming techniques

**Course aim:** To describe the techniques used by analysts and designers

Learning objectives:
After taking this course, the user should be able to
· create entity relationship diagrams
· perform normalization on a set of relations
· describe scenario diagrams for the Context model, the Workflow Process model, the Task Sequence model, and the Physical Environment model
· draw data flow diagrams

Units in Microsoft Solution Architectures: Analysis and Design Techniques:
Entity relationship diagrams
Normalization
Scenario models
Data flow diagrams

**Course incorporates:** Test, hands-on exercises