Publication date : 06/27/2024

Course : Object-oriented design

object-oriented software engineering

Practical course - 4d - 28h00 - Ref. COB

Price : 2100 € E.T.

How do you approach object-oriented design? How do you move from a functional approach to an Object-oriented approach? How to write an object-oriented program with real scalability and reusability? This course will provide you with a conceptual and practical mastery of Object-oriented design.

Teaching objectives

At the end of the training, the participant will be able to:

- Understand the principles and specifics of object-oriented design.
- Move from a functional to an object-oriented approach.
- Model object-oriented software using UML notation.
- Translate the UML model into an object language.
- Describe framework and component approaches.
- Learn how to implement Design Patterns.

Intended audience

Developers and project managers wishing to learn more about object-oriented design.

Prerequisites

Basic knowledge of application design and software development.

Practical details

Hands-on work

The points covered will be illustrated by numerous exercises specially chosen to highlight the principles and specific features of object-oriented design, from initial analysis through to implementation in an Object language.

Course schedule

PARTICIPANTS

Developers and project managers wishing to learn more about object-oriented design.

PREREQUISITES

Basic knowledge of application design and software development.

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

1 What can we expect from the Objet approach?

- Why use object technologies?
- The challenges of new computing: modularity (plug-ins), reusability, scalability.
- The use of component libraries. How does the Objet approach meet these challenges?
- How should you approach an Objet problem?
- Acquired knowledge from other areas of computer science and other disciplines.

2 Basic concepts of the Object-oriented approach

- Objects: a procedure/data duality.
- Classes as models of object structure and behavior, and instances as class representatives.
- Methods: procedures defined in classes and used by instances.
- Interaction between objects by sending messages. How are messages interpreted by objects?
- Inheritance. Inheritance and variable typing in strongly typed languages (C++, Java).

3 Diagrams and object representation using UML

- The main diagrams (class diagrams, sequence diagrams) and their use in object design.
- Object notation and representation tools: getting to grips with a commercially available modeler.

4 The main principles of object-oriented design

- What do we put in the form of an object? The principle of reification.
- Criteria for deciding what to put in Object form. Mistakes to avoid.
- How to structure object-oriented software? The principle of modularity and domain decomposition.
- How to structure a set of classes? Principles of abstraction and classification.
- How to think about interaction between objects? The principle of encapsulation and autonomy.
- Analyze complex systems in terms of communications. General approach. Mistakes to avoid.
- Criteria for "good" class hierarchies. Mistakes to avoid.

5 How do you approach Objet software?

- Development principles. From spiral development to incremental development.
- Identification of domain entities and description of interactions. Program reusability and scalability.
- Designing with objects doesn't mean using an object tool!
- Mistakes to avoid.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

6 From design to implementation

- Translate UML class diagrams into programming languages and databases.
- Principles of implementing object-oriented applications. The importance of distribution. Generalized client-server models.
- Today's major object platforms: Microsoft's .NET and SUN's JEE technologies.
- Comparison of their strengths and weaknesses.
- The importance of distribution. Class libraries. Programming and component languages.

7 The frameworks and components approach

- The software lifecycle problem.
- Scalability and maintenance issues call for a software approach that enables scalability.
- The framework and component approach, based on object-oriented thinking, is a response to this need.
- How can you quickly design and build applications using frameworks and reusable components?
- How do you integrate software components into an existing framework? How to build frameworks?
- Take an existing application and transform it into a framework, making it scalable.
- Major framework classes. Current component models.

8 Design Patterns

- How can experience be reused when designing and developing object applications?
- Design Patterns as software solutions to general recurring problems.
- The different types of Design Patterns. Examples of Design Patterns.
- Advantages and limitations of Design Patterns.
- How to use Design Patterns in practice Learn how to implement Design Patterns in practice.