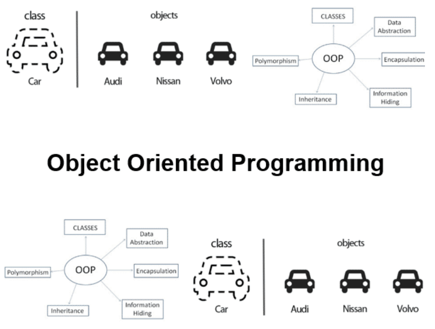
www.spiraltrain.nl

SpiralTrain
developer courses

Object Oriented Programming

Duration: 3 days Price: € 1850

Open Schedule



Audience Course Object Oriented Programming

The course Object Oriented Programming is intended for anyone who wants to learn object oriented programming with classes and objects.

Prerequisites Course Object Oriented Programming

In order to participate in this course experience with programming in a procedural programming language is required.

Realization Training Object Oriented Programming

The theory is discussed on the basis of presentation slides. The theory is explained further through demos. After discussing a module there is the possibility to practice. Course times are from 9.30 to 16.30.

Certification Course Object Oriented Programming

After successful completion of the course the participants receive an official certificate Object Oriented Programming.

Content Course Object Oriented Programming

In the course Object Oriented Programming participants learn to program in an object oriented language such as Java, C# or Python. The participants can choose which language they want to use in the course. Object Orientation has proven to be a fertile programming paradigm. Most modern programming languages today are object oriented and some older languages such as C or PHP have added Object Orientation later.

Intro Object Orientation

The course starts with an overview of how Object Orientation evolved from other software development paradigms such as structured and procedural programming.

Lowering of Semantic Gap

An important advantage of Object Orientation is that domain concepts can be found directly in the software. It is explained how this lowering of the Semantic Gap makes the code more understandable and maintainable.

Classes and Objects

Subsequently concepts such as Classes and Objects, Fields and Methods, Getters and Setters, Constructors and Destructors are discussed. The concepts are the same for all Object Oriented languages, but in the course attention is also paid to differences at the detail level.

Encapsulation

Also treated is the concept of Encapsulation with which the internal data of classes is shielded from the outside world so that changes in the implementation can be made without modifications to the calling code.

Inheritance and Polymorphism

The concepts of Inheritance and Polymorphism are also part of the course program. By means of Inheritance derived classes can reuse the code from the base class and thus avoid duplication of code. Polymorphism makes it possible to give base class methods a different meaning in a derived class. The runtime environment can then automatically find these methods through dynamic binding.

Design Patterns

Finally attention is paid to Design Patterns in Object Oriented software, which provide standard template solutions for common problems.

SpiralTrain BV
Standerdmolen 10, 2e verdieping
3995 AA Houten

info@spiraltrain.nl
www.spiraltrain.nl
Tel.: +31 (0) 30 – 737 0661

Locations
Houten, Amsterdam, Rotterdam, Eindhoven, Zwolle, Online

Modules Course Object Oriented Programming

Module 1 : Intro Object Orientation	Module 2 : Classes and Objects	Module 3 : Encapsulation
OO Origins Abstraction Levels Domain Analysis Unstructured Programming Procedural Programming Object Oriented Programming OO Benefits Reusability Lowering Semantic Gap Higher Abstraction Objects as Domain Concepts Objects as Program Concepts	Classes are Types Objects are Instances Fields Methods Creating Objects Object Initialization Constructors Using Objects Getters and Setters Destructors Current Object this or self	Encapsulation Benefits Information Hiding Access Specifiers private and public Implementation Changes Validity Checks Ensuring Data Validity Class Variables static Data Class Methods static Methods static Initializers
Module 4 : Inheritance	**Module 5 : Polymorphism**	**Module 6 : Design Patterns**
Deriving Classes Class Hierarchies Hiding Instance Variables Overriding Methods Overloading Methods Constructor Chaining Accessing Base Class protected Members super or base Multiple Inheritance	Call Overridden Functions Virtual Functions Role of v-table Polymorphism Benefits Abstract Classes Incomplete Base Classes Concrete Classes Interfaces Interface Implementation Dynamic Binding	What are Design Patterns? Common Problems Pattern Solutions Singleton Pattern private Constructors Creation Functions Adapter Pattern Adapting an Interface Observer Pattern Publish and Subscribe