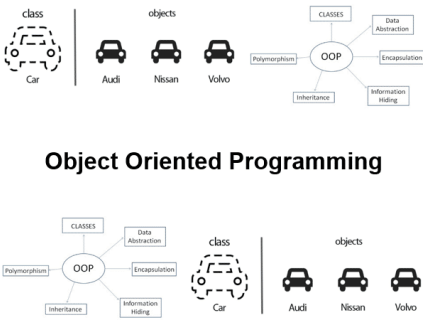
Object Georienteerd Programmeren

Duur: 3 dagen **Prijs: € 1999**

Open Rooster



Object Oriented Programming

Doelgroep Cursus Object Georiënteerd Programmeren

De cursus Object Georiënteerd Programmeren is bedoeld een ieder die object georiënteerd wil leren programmeren met classes en objects.

Voorkennis Cursus Object Georiënteerd Programmeren

Om aan deze cursus te kunnen deelnemen is ervaring met programmeren in een procedurele programmeer taal vereist.

Uitvoering Training Object Georiënteerd Programmeren

De theorie wordt behandeld op basis van presentatie slides. De theorie wordt verduidelijkt door middel van demo's. Na bespreking van een module is er de mogelijkheid te oefenen. De cursustijden zijn van 9.30 tot 16.30.

Certificering Cursus Object Georiënteerd Programmeren

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Object Georiënteerd Programmeren.

Inhoud Cursus Object Georienteerd Programmeren

In de cursus Object Georiënteerd Programmeren leren de deelnemers programmeren in een object georiënteerde taal zoals Java, C# of Python. De deelnemers kunnen zelf kiezen welke taal ze in de cursus willen gebruiken. Object Oriëntatie heeft zich bewezen als een vruchtbaar programmeer paradigma. De meeste moderne programmeer talen zijn tegenwoordig object georiënteerd en aan wat oudere talen zoals C of PHP is in de loop der tijd Object Oriëntatie toegevoegd.

Intro Object Oriëntatie

De cursus gaat van start met een overzicht over hoe Object Oriëntatie is voortgekomen uit andere software development paradigma's zoals structured en procedural programmeren.

Lowering of Semantic Gap

Een belangrijk voordeel van Object Oriëntatie is dat domein concepten direct in de software zijn terug te vinden. Uitgelegd wordt hoe door dit verlagen van de Semantic Gap de code begrijpelijker en beter onderhoudbaar wordt.

Classes en Objects

Vervolgens wordt ingegaan op concepten als Classes en Objects, Fields en Methods, Getters en Setters, Constructors en Destructors. De concepten zijn hetzelfde voor alle Object Georiënteerde talen, maar in de cursus wordt ook aandacht besteed aan verschillen op detail niveau.

Encapsulation

Eveneens komt het begrip Encapsulation aan de orde waarmee de interne data van classes wordt afgeschermd van de buitenwereld en waardoor wijzigingen in de implementatie zonder aanpassingen in de aanroepende code kunnen worden doorgevoerd.

Inheritance en Polymorfisme

De begrippen Inheritance en Polymorfisme worden eveneens behandeld. Door middel van Inheritance kunnen afgeleide classes de code uit de base class hergebruiken en daarmee duplicate code vermijden. Polymorfisme maakt het mogelijk base class methods een andere betekenis te geven in een afgeleide class. De runtime omgeving kan deze methods door dynamic binding dan automatisch vinden.

Design Patterns

Tenslotte wordt aandacht besteed aan Design Patterns in Object Georiënteerde software, waarmee standaard template oplossingen worden geleverd voor veel voorkomende problemen.

Modules Cursus Object Georienteerd Programmeren

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