Practical Objects

Hands-on Introduction to Object-Oriented Concepts & Programming

Object-orientation is now considered a mainstream approach to software development. Current trends include component-based development, distributed object computing, design patterns, object modelling — most popularly using UML — and agile development methods — such as Extreme Programming. The principles on which object-orientation is based are drawn from good software engineering theory and practice, balancing the practical aspects of day-to-day development with longer term goals such as management of complexity. As such, object-oriented development supports different project styles and sizes.

It can be hard for programmers to develop a working understanding of OO without hands-on experience; explanations of terminology are useful but cannot reasonably change how a programmer approaches development. The *Practical Objects* course aims to introduce OO concepts, terminology and practices to software development professionals. The course is based around lectures, discussion and hands-on programming exercises. To avoid the detail and distraction of a language such as Java, C# or C++, the course uses a scripting language to reduce the language introduction overhead and encourage a practical, hands-on appreciation of OO.

Objectives

- Explain the terminology of object-orientation
- Understand the principles and practices on which object-oriented development is based
- Become familiar with modern trends in OO development, such as component-based development, design patterns, refactoring and object modelling
- Experience object-oriented programming first hand

Audience

The course is suitable for software developers who wish to gain a practical understanding of OO programming. No prior knowledge of OO concepts or OO programming is assumed. However, experience of programming in a modern programming language is required — the course is an introduction to OO programming but not an introduction to programming.

Content

Object Fundamentals Objects · Encapsulation · Methods and messages · Classes · Object instantiation and lifetimes · Polymorphism · Inheritance · Overriding

Development Process Requirements and user stories · Modelling and UML · Patterns · Prototyping · Testing · Refactoring · Iterative and incremental development · Agile methods

Object-Oriented Programming Static versus dynamic typing · Java · C++ · C# · Scripting · Libraries and frameworks · Defining classes and inheritance, and using polymorphism

Structuring Objects Object relationships · Delegation · Design by contract · CRC cards · Command versus query methods · Collections · Values, entities and services as objects

Relating Classes Inheritance and substitutability · Pluggability · Abstract classes and methods · Dependency management · Pure interfaces · Multiple inheritance

Design Patterns Recurring design ideas · Recursive whole–part structures · Callback and notification · Transparent forwarding · Encapsulated object creation

Refactoring Personal hygiene for code · Extracting methods · Splitting classes · Generalising class hierarchies · Replacing explicit selection with polymorphism

Object Technologies Persistence · Component-based development · Distributed object computing · Multi-tier architectures · COM(+) · .NET · CORBA · Object request brokers · J2EE

Additional Details

Duration 3 days

Setup Projection facilities for a laptop · One PC per delegate or pair of delegates · Whiteboard and/or flip chart · Reference cards (150mm × 100mm)

Contact Kevlin Henney · kevlin@curbralan.com · Curbralan Limited · +44 117 942 2990