Object-Oriented Design and Programming in LabVIEW™ Course Manual

Course Software Version 2010
November 2010 Edition
Part Number 325616A-01

Worldwide Technical Support and Product Information
ni.com

National Instruments Corporate Headquarters
11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices
Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 41309277, Japan 0120-527196, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 328 90 10, Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

For further support information, refer to the *Additional Information and Resources* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.

Contents

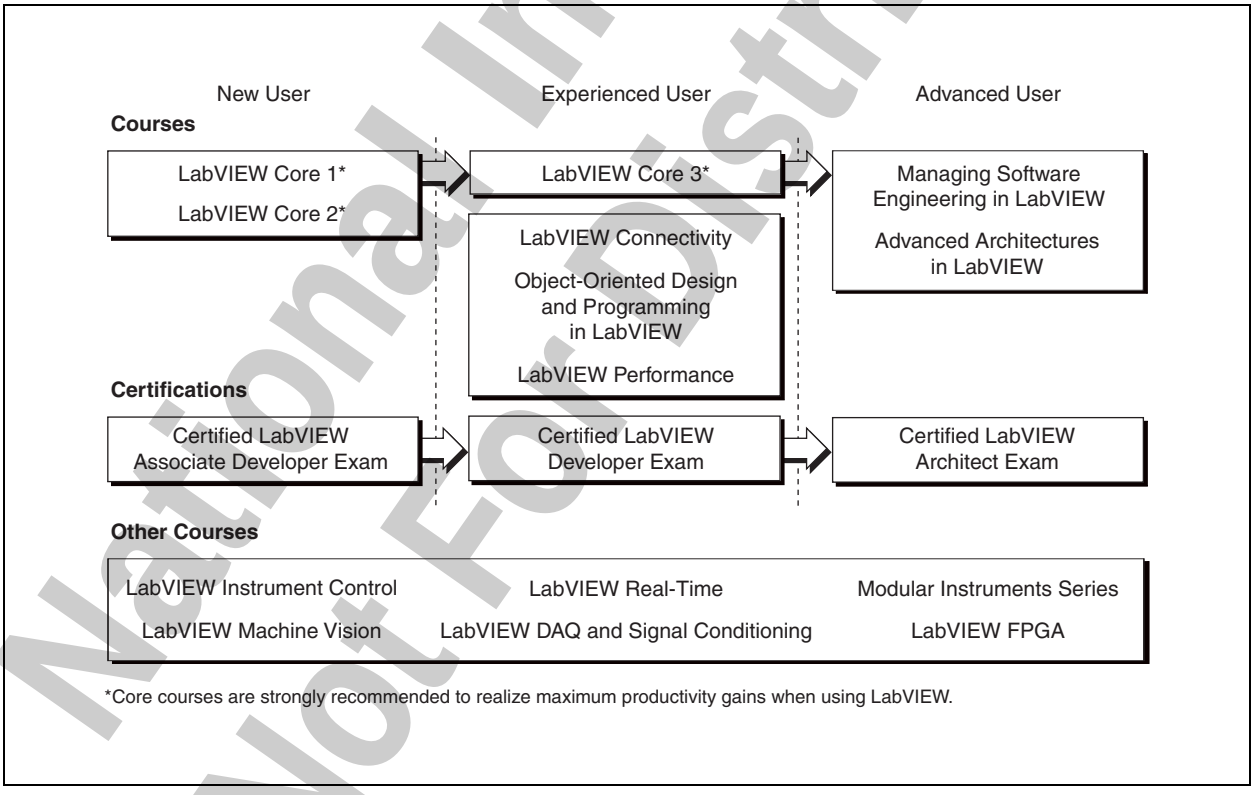
Student Guide

Thank you for purchasing the *Object-Oriented Design and Programming in LabVIEW* course kit. This course manual and the accompanying software are used in the two-day, hands-on *Object-Oriented Design and Programming with LabVIEW* course.

You can apply the full purchase price of this course kit toward the corresponding course registration fee if you register within 90 days of purchasing the kit. Visit `ni.com/training` to register for a course and to access course schedules, syllabi, and training center location information.

A. NI Certification

The Object-Oriented Design and Programming in LabVIEW course is part of a series of courses designed to build your proficiency with LabVIEW and help you prepare for exams to become an NI Certified LabVIEW Developer and NI Certified LabVIEW Architect. The following illustration shows the courses that are part of the LabVIEW training series. Refer to `ni.com/training` for more information about NI Certification.



B. Course Description

This course assumes that you have taken the *LabVIEW Core 3* course or have equivalent experience.

The course is divided into lessons, each covering a topic or a set of topics. Each lesson consists of the following parts:

- An introduction that describes what you will learn
- A discussion of the topics
- A set of exercises that reinforces the topics presented in the discussion

  Some lessons include optional exercises or challenge steps to complete if time permits
- A summary that outlines important concepts and skills taught in the lesson

Note For course manual updates and corrections, refer to `ni.com/info` and enter the Info Code `lvoop`.

C. What You Need to Get Started

Before you use this manual, make sure you have the following items:

- ❑ Windows XP or later
- ❑ LabVIEW Professional Development System 2010 or later
- ❑ *Object-Oriented Design and Programming in LabVIEW* CD, which contains the following folders and files:

Folder Name	Description
`Exercises`	Contains all the VIs and support files needed to complete the exercises in this course
`Solutions`	Contains completed versions of the VIs you build in the exercises for this course

D. Installing the Course Software

Complete the following steps to install the course software.

1. Insert the course CD in your computer. The **Object-Oriented Design and Programming in LabVIEW Course Setup** dialog box appears.
2. Click **Install the Course Material**.
3. Follow the onscreen instructions to complete installation and setup.

Exercise files are located in the `<Exercises>\Object-Oriented Design and Programming in LabVIEW` folder.

Tip Folder names in angle brackets, such as `<Exercises>`, refer to folders on the root directory of your computer.

Repairing or Removing Course Material

You can repair or remove the course material using the Add or Remove Programs feature on the Windows Control Panel. Repair the course manual to overwrite existing course material with the original, unedited versions of the files. Remove the course material if you no longer need the files on your computer.

E. Course Goals

This course prepares you to:

- Determine the appropriateness of using an object-oriented approach to solve the problem
- Design an application using object-oriented design principles
- Implement a basic class hierarchy using LabVIEW classes
- Modify an existing application written in G to replace common patterns with objects

This course does *not* present the following topic:

- Developing a complete application for any student in the class; refer to the NI Example Finder, available by selecting **Help»Find Examples**, for example VIs you can use and incorporate into VIs you create

F. Course Conventions

The following conventions are used in this course manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **Options»Settings»General** directs you to pull down the **Options** menu, select the **Settings** item, and select **General** from the last dialog box.
	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a note, which alerts you to important information.
	This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
bold	Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you enter from the keyboard, sections of code, programming examples, and syntax examples. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`<Exercises>/.../`	This short-hand denotes the `Object-Oriented Design and Programming in LabVIEW` directory located in the `Exercises` directory.
`<Solutions>/.../`	This short-hand denotes the `Object-Oriented Design and Programming in LabVIEW` directory located in the `Solutions` directory.

1

Introduction to Object-Oriented Programming

This lesson provides an overview of object-oriented programming and describes the elements required to develop an object-oriented application. You will learn how to evaluate a project to determine if an object-oriented approach is appropriate for the application you are designing.

Topics

A. What is a Class?

B. What is Object-Oriented Design?

C. What is Object-Oriented Design?

D. What is Object-Oriented Programming?

A. What is a Class?

The idea of object-oriented programming is to focus first on the parts of your system that represent things and then to organize your code by assigning tasks to the appropriate thing. These things are called classes. A *class* is a collection of data and the functions that interact with that data. Classes are nouns. They are things in a program that take action or are acted upon. When you look at the description of a new programming assignment, you can identify potential classes by looking for nouns.

Begin by focusing on classes that represent tangible things, because these are the easiest to recognize in a design. Examples of these types of classes classes include person, car, assembly line, or sound card.

Over time, you will begin to recognize that many intangible things can be represented in a computer program as classes. Some less tangible classes still reflect aspects of the real world, such as sound wave, flight path, bank account, or signal filter. Some classes are computer-specific concepts such as string, path, timestamp, or error report. Architects of large object-oriented systems recognize that classes are effective at representing more abstract concepts such as relationship, style, or geometric transformation.

The one thing these nouns, or classes, have in common is that they are all things. They have a set of attributes and a set of actions that they perform. A class defines a new data type and you will use these new data types when writing your software.

Any value, attribute, or quantity in a computer program can be considered as part of some class. It is the job of that class to define the storage, modifications, and rules for that data. Do not make the mistake of believing that all classes are complex. In some programming languages “32-bit unsigned integer” is a class. It fits the definition: a well defined block of data with a defined set of operations that you can do on that data. It is distinct from the class of “double precision floating point number,” which stores different data and has a different set of allowed operations.

Many of the classes you create for an application will be small and simple, and sometimes it might seem that creating the class serves no benefit compared to using raw memory or primitive data types directly. It is possible to create too many classes and make an application overly complex. The goal of this course is to show you how a large software project can be more easily tackled by breaking it down into many well defined classes, some of them quite small and simple. If you start to think of everything as a potential class, you should find that scaling a design down is easier than building it up.