

## Introduction

---

### 0.1 The Subject of this Course

This is, first and foremost, a course about object-orientation: the course is structured around the principal concepts of object-orientation. Our main hope for this course is that, at the end of it, you will have a good understanding of the main principles and techniques of object-orientation and that you will understand the reasons for, and uses of, these principles and techniques.

Why are we so concerned that you learn object-oriented principles? To understand this you need to have an idea of the crisis that has faced, and still continues to face, the software industry. Real commercial software systems increase in size and complexity year on year. It is impossible to design a modern software system all in one go; it would be impossible to understand an unstructured system well enough to maintain it by, for example, making changes when an error is found or when supporting technologies change. Object-orientation has grown as a response to this problem.

Object-orientation is a portfolio of related techniques and concepts that help you to handle the complexity of software systems. The techniques and concepts encourage well-structured systems with reusable and reused components. The idea of reusability is central to modern program development: most development involves the reuse of code you have written before or code that has been supplied for you by such vendors as Sun Microsystems. To be able to reuse this code effectively entails an understanding of object-oriented principles.

One of the themes stressed throughout the course is the use of abstraction to help control the design of complicated systems. Object-orientated languages allow for greater possibilities for abstraction than procedural languages do. These techniques can be seen as generalisations of the notion of abstract data type. The abstractions studied in this course take the form of abstract views of procedures, abstract data types (which are abstractions of packages of data and procedures), design patterns (which are abstractions of techniques for solving problems), and frameworks (which are abstractions of solutions to large problems).

This is, secondly, a course about Java. We have chosen Java as the language on which to base this course, and much of the whole degree programme for several reasons:

- a. Java is a purer and cleaner implementation of object-oriented concepts than C++
- b. Java has more powerful facilities in the form of libraries of reusing code than any other language
- c. Java is platform independent
- d. Java is more commercially relevant than any other modern computer language, with the possible exception of C++.

The application domain for most of this course is graphical programming. There are several reasons we believe graphical programming is appropriate to this course and useful for you to study at this point in the programme. Some of these are:

- a. Most important object-oriented principles come up naturally in graphical programming
- b. Almost all modern computing systems of any complexity include a graphical interface
- c. Java's graphical library is particularly rich
- d. Graphical programming is fun and rewarding.

Modern computing systems generally are meant to be accessed by a multitude of users, some of whom will not be as comfortable with Java code as you. Most users, especially uninitiated ones, find it easier and more pleasant to interact with a system graphically, using buttons etc, than textually by typing into the system. Java has been designed with this in mind and has provided developers with a great deal of pre-defined classes to re-deploy in their applications. And the way that the Java graphics classes are put together follows object-oriented structuring principles. Moreover, graphical applications are very fruitfully seen as semi-independent objects on a screen. And, lastly, it is a particularly satisfying feeling to write a program and see a picture animated as a consequence.

---

## 0.2 How To Study This Course

This is an intermediate programming course. It is intended for students with some previous programming experience; ideally you will have Java experience up to the level of the first

year course in this programme. While only a very small amount of specific Java knowledge will be drawn upon without explanation, you are likely to find this course very difficult if this is your first exposure to Java and, particularly so, if this is your first experience of an object-oriented language of any sort.

Unlike other programming courses in this programme, this course follows a textbook very closely. The subject guide will repeatedly refer you to the textbook to read particular sections and to do exercises. It is very important that you do as many of the exercises as you can. Some exercises are suggested throughout the subject guide; you should do at least those, and preferably more. It is as impossible to learn to program without doing programming exercises as it would be to learn to ride a bicycle purely by reading a book about it. And, just as in learning to ride a bicycle, you should expect to fall sometimes. Nobody gets either bicycling or programming right straight off. The difference is that while most people eventually learn to ride a bicycle well enough to stay upright, learning to program is a continuous, never completed process: there will always be new things to make you fall. That can be frustrating, but it can also make learning to program a source of continuous small satisfactions. We very much hope that, at the end of the session, you will find this a fitting description of your experience of the course.

The purpose of the subject guide is to guide you through the material, adding explanations of topics in the text, adding examples, and occasional hints and solutions to exercises.

You will notice a substantial shift in the use of the subject guide in the middle of the course: there is quite a lot written in the guide to accompany the first three chapters of the book and comparatively little afterwards. There are two reasons for this. First, the first three chapters are introductory and you may feel the need for extra support at that stage. By the time you get to Chapter Four, you should be comfortable enough in the subject and in Laszlo to go through the book with only a small amount of guidance. And, secondly, the book is more detailed and slower in its treatment of material from Chapter Four on.

As well as the subject guide and the text, you should have a CD of additional materials. The CD contains the code that is referred to in the texts, and the programming environment—*BlueJ*—that you will be using throughout the course. Instructions for setting up *BlueJ* and your source code for the course are given in the Appendix.

We suggest you devote your time to the course in roughly these proportions:

Chapter One 10%

Chapter Two 15%

Chapter Three 10%

Chapter Four 15%

Chapter Five 15%

Chapter Six 20%

Chapter Seven 15%

---

### 0.3 Topics

The major topics of the course are given below:

- Classes and Objects
- Methods
- Data Abstraction
- The Graphics2D API (Application Program Interface)
- Composition
- Inheritance
- Design Patterns
- Building Graphical User Interfaces

---

## 0.4 Assessment

### *Examination*

**Important:** the information and advice given in the following section is based on the examination structure used at the time this guide was written. However, the University can alter the format, style or requirements of an examination paper without notice. Because of this, we strongly advise you to check the instructions on the paper you actually sit.

There will be one three-hour unseen written examination paper for 2910212. This will contain two main sections, the first section corresponding to this guide. You will be asked to answer three questions from each of these sections.

The first section of the examination will not differ radically from the sample section A of the examination paper included in Appendix 1 of this guide. A sample of section B of the paper is included in volume 2 of the subject guide.

### *Coursework*

Two coursework assignments (issued separately each year) will be set for students to practice their program design and development skills using theoretical knowledge gained from the course.

The approach to solving a given problem by implementing an algorithm differs from person to person and from problem to problem, however it generally includes the following stages of work:

1. Analysis of the given problem
2. Decomposition into sub-problems
3. Derivation of a general plan to solve the sub-problems
4. Development of the program
5. Tests and possibly formal proofs of the correctness of the program
6. Comments or review of the limitations of the program.

At the end, a full document should be written which includes a section for each of the above stages of work.

---

## 0.5 Books

### *Course text*

The main text for this course is:

Laszlo, Michael *Object-Oriented Programming featuring Graphical Applications in Java*. (Boston; London: Addison Wesley, 2003) [ISBN 0201726270].

It is imperative that you have access to this book. Throughout the course, you will be referred to it for reading and exercises. It will be referred to variously as “Laszlo” and “the course text” in this guide. Note that any undirected reference to a book chapter or exercise is, in fact, a reference to Laszlo.

For most of the course, the structure and content of the book determines the structure and content of the guide. Indeed, in later chapters of the subject guide, care has been taken to try to synchronise the section numbers of the guide with that of the course text. For example, where possible, the section numbered 3.2 in the guide refers to the section numbered 3.2 in Laszlo.

### *Other books*

There are three other books that we have drawn upon while writing these notes. You may well find these books useful as well:

Budd, Timothy *Understanding Object-Oriented Programming With Java*. (Reading, Mass: Addison Wesley, 2000) [ISBN 0201612739]. Updated Edition (New Java 2 Coverage).

Barnes, David J. and Michael Kölling, *Objects First with Java: A Practical Introduction using BlueJ*. (Prentice Hall / Pearson Education, 2003) [ISBN 0-13-044929-6].

Knudsen, Jonathan *Java 2D Graphics*. (Sebastopol, Calif.: O’Reilly and Associates) [ISBN 1-56592-484-3].

Another book that we recommend is:

Eckel, Bruce *Thinking in Java*. (Upper Saddle River, NJ: Prentice Hall, 2000) second edition [ISBN 0130273635]. <http://www.bruceeckel.com/>

This book is freely downloadable from the Internet. To find a website near you, consult:

[www.mindview.net](http://www.mindview.net)

Almost any Java, object-orientation, or design pattern book would be useful. Some of these are listed in a booklist in section 0.7 below.

---

## 0.6 Compiling and Running Java Programs

The subject guide is written under the assumption that you will be using *BlueJ* to compile and run your programs. If you do not do so, you will find it difficult to follow some of the writing. The *BlueJ* provided on the CD runs under several operating systems, including, Windows, Linux and MacOS. If you have any problems with *BlueJ*, you should consult the tutorial that is included on the course CD.

---

## 0.7 What you will have achieved when you have completed this subject guide

Having completed this subject guide you will understand how object-orientation is used as a way of developing computer applications that are robust and reusable. You should also know how to deploy object-oriented development techniques in Java, and how to write Java applications that have substantial graphical facets. The learning outcomes at the end of each chapter summarise the examinable topics.

---

## 0.8 Some suggested Books

Deitel and Deitel *Java: How to Program*. (Prentice Hall International, 2000) third edition [ISBN 0-13-012507-5].

Flanagan, David *Java in a Nutshell*. (O'Reilly, 1999) third edition [ISBN 1-56592-487-8].

Hubbard, John R. *Schaums: Outlines Programming with Java*. (McGrawHill, 1998).

Jia, Xiapong *Object-Oriented Software Development Using Java: Principles, Patterns, and Frameworks*. (Addison Wesley, 2000).

Lambert, Kenneth A. and Martin Osborne *Java: A Framework for Programming and Problem Solving*. (Brookes Cole, 2002).

Liang, Y. Daniel *Introduction to Java Programming with JBuilder 4*. (Prentice Hall, 2001) second edition [ISBN 0-13-033364-6].

Liang, Y. Daniel *Rapid Prototyping*. (Prentice Hall, 2001) [ISBN 0-13-033364-6].

Richter, Charles *Designing Flexible Object-Oriented Systems with UML*. (Indianapolis, IN: Macmillan Technical Publishing, c.1999) [ISBN 1578700981].

Vlissides, John *Pattern hatching: design patterns applied*. (Harlow: Addison Wesley, 1998) [ISBN 0201432935].