Undergraduate Topics in Computer Science

'Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275-300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at http://www.springer.com/series/7592

Kingsley Sage

Concise Guide to Object-Oriented Programming

An Accessible Approach Using Java

Springer

Kingsley Sage
School of Engineering and Informatics
University of Sussex
Falmer, East Sussex, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-13303-0 ISBN 978-3-030-13304-7 (eBook)
https://doi.org/10.1007/978-3-030-13304-7

Library of Congress Control Number: 2019931822

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The twenty-first century continues to experience the relentless expansion of the IT revolution into our daily lives. We consume services, do our shopping on-line, listen to music streams and watch movies on demand. The impact of social media has had a profound impact on our society and has changed fundamentally the way we obtain and consume news, information and ideas. There is little sign of a slowdown in this dramatic shift in our relationship with technology. Vast research budgets are being applied to the development of autonomous vehicles, and in applying Artificial Intelligence to change the way we live. But it has also changed the demand for skills within our workforce. The demand for manual skills is in decline, and the demand for IT and programming skills is rising at an unprecedented rate.

In comparison to the industrialists of the nineteenth and twentieth centuries, the twenty-first-century entrepreneurs are experts in IT, programming, software design and development, and developing practical applications using concepts such as Artificial Intelligence for our daily lives. With this profound paradigm shift has come a need for the workforce of many industrialised nations to evolve. Governments recognise the need for a huge increase in the workforce with programming skills. In the United Kingdom, and in many other industrialised nations, core coding skills are now a part of the secondary school curriculum. Learning to program is no longer considered to be just a part of the traditional journey of the Computer Science undergraduate, but a broader skill that underpins an IT literate workforce for the modern age.

What is the Purpose of This Book?

When I was first approached to write this book, it was suggested that its purpose was to provide an accessible introduction to coding and the world of Object Oriented Programming (OOP). Standard texts on the subject often fall between those that provide only a very lightweight treatment of the subject ("a little knowledge can be a frustrating thing"), and those that run to 500 pages or more that are rather better suited as reference texts or as support on a lengthy period of study in depth. The challenge for this book is to provide an accessible introduction to the world of

coding and OOP in a way that is helpful to the first-time coder and allows them to develop and to understand their knowledge and skills in a way that is relevant and practical. The examples developed for this book are intended to show how OOP skills can be used to create applications and programs that have everyday value, rather than examples that have been synthesised solely to demonstrate an academic point.

The reader should be able to use this book to develop a solid appreciation of OOP and how to code. The programming language used throughout is Java. Java has been chosen as it can be used across all computing platforms, because it has a commercial skill that has a clear on-going value derived from its adoption as a core language for smartphone applications on the Android platform, and as the language at the heart of the Java EE 8 Jakarta Enterprise scale framework. The book focusses on the core Java language and does not consider smartphone or EE 8 coding, as these require skills over and above what this book is about. However, a knowledge of core Java coding and some of the related issues also discussed in this book would form an appropriate pre-requisite for the further study of these topics.

Although this book uses Java as its illustrative programming language, many of the ideas may be translated directly into other OO languages such as C++, C# and others. Throughout this book, programming in Java is demonstrated using the BlueJ Integrated Development Environment (IDE). BlueJ is a well-established IDE for learning BlueJ and is widely used in schools and Universities. Eclipse is the closest product to an industry standard for the development of Java, but it is often found too complex for the task of teaching and learning.

Who is This Book Aimed at?

As someone with over 20 years of teaching experience from level 3 through to postgraduate, from traditional University teaching to adult education, I have never been able to identify satisfactorily what defines the ability of an individual to learn to program. Suffice to say, all that is really needed is an interest in the subject and time. The aim of this book is to provide an accessible entry into the world of Object Oriented Programming (OOP).

The book does not assume any prior knowledge of coding, or any prior knowledge of software engineering or OO, not does it require any prior exposure to mathematics. Whilst such prior knowledge is not unhelpful, it is not essential to learn to program. Instead, this book takes a more everyday experience to the subject, drawing on examples from everyday experience to explain what OO is and why it is relevant in the modern programming experience. As such, the book is aimed at those who are coming to OO programming for the first time. It is therefore likely to be useful as a one-semester book introducing the topic to those new to the study of computer science at the undergraduate and postgraduate levels, and those who are just learning for the purpose of self-improvement or professional development. Whilst the book is aimed at those with no prior coding experience, it does

explore broader topics surrounding coding. This with some prior knowledge may opt to skip some of the early chapters. That does not impact the usefulness of this book in terms of learning to code in Java.

What's in the Book?

Chapter 1 starts with an overview of what programming and coding is all about. It includes some useful historical perspective on the development of programming languages and the core ideas that underpin all programming languages. It introduces the idea of a computing machine and concepts such as a compiler. This section is helpful to those who have no prior experience of computing as it helps subsequent understanding of some of the core coding processes and terminology. The chapter then continues to discuss how the need for OOP arose in the period from the end of the 1970s to the present day, and a discussion of why it is considered important to help us solve modern-day programming problems.

Chapter 2 provides a short introduction to programming in Java using BlueJ. It is intended to provide just enough knowledge and skills to create and execute a single-class Java program under BlueJ. This is significant as it then facilitates discussion of the core principles of procedural and structured programming, such as loops and conditional statements. Those with prior experience of coding using languages such as C and Python may opt to skip this chapter, as they would undoubtedly be familiar with much of the content. I chose to organise the book this way as the basic procedural and structured coding constructions are common to almost all programming (or at least those that owe their syntactic ancestry to C), and getting these constructions understood at this stage allows for a more specific focus later on the principles of OO.

Chapter 3 gets into the details of what OO really is and how it can be applied to solve modern programming challenges. We start with a discussion of what classes and objects are, and how the construction and execution of an OO program parallels the way that human organisations such as a large office operate. Such analogies are invaluable in appreciating the true benefits of the OO paradigm. In this chapter, we develop a set of small multi-class Java applications and consider the cornerstone issues in OO design of class cohesion and coupling.

Chapter 4 considers a range of Java library objects and packages such as the `String` and the `ArrayList`, and introduces the idea of the Application Programming Interface (API). This enables the reader to start building more complex applications involving simple linear collections of objects. These ideas are developed using a set of simple programs that can be enhanced in many different ways as an exercise for the reader.

Chapter 5 delves further into the OO paradigm and considers how OO design forms an essential part of producing a useful solution to a problem. The chapter introduces the idea of class polymorphism (super and sub-classes) and how this can be used to create a program with a structure that more closely mirrors an underlying

domain. The chapter also looks further into the idea of selecting classes that are suited to solving specific problem and so also has elements of software engineering principles and practice.

Chapter 6 considers what to do when code encounters an error condition. Software systems are not immune to errors either at the coding or at the run time phases, and modern software systems need to be built in a robust manner so that they behave in a predictable manner when something goes wrong. The exception handling mechanism is introduced, along with steps on laying out a program to assist in debugging it. This chapter also considers practical measures that are adopted in defensive coding.

Chapter 7 digs deeper into the work of arrays and collections, notably fixed length arrays, the `HashMap` and `HashSet`, and shows how different collection types can be used to effectively model different real-world collections of data. This chapter also includes some background on the underlying ideas for these collection types, such as the hash table.

Chapter 8 provides an introduction to building a Graphical User Interface (GUI) using Swing. Although some may consider Swing a relatively old library for the development of a GUI, the key ideas are relevant across a range of other libraries such as JavaFX, and Swing forms more of a core element of the Java landscape. The development of GUIs is a large topic in its own right, so this chapter can only ever serve as an introduction. In this chapter, we also consider the concept of a design pattern, specifically the idea of Model View Controller (MVC) architecture, and how a Java application can be constructed in a well-recognisable design configuration.

In the final Chap. 9, two complete applications are presented, from conceptual design to implementation to help cement the ideas presented in the previous chapters. One is a text-based application with no Graphical User Interface (GUI). The other is a small GUI-based application to give a sense of how to build a GUI on top of an underlying application.

All the code examples used in this book and the two example projects described in Chap. 9 are available as on-line resource accompanying this book.

It is my hope that this book will inspire the reader to learn more about the world of OO and coding. As such, it represents the start of a learning journey. As with all endeavours, clarity will improve with time and effort. Few will write an award-winning book at their first attempt. Few artists will paint their defining masterpiece at the outset of their career. Programming is no exception and your skills will improve with effort, time, reflection and experience. But every learning journey has to start somewhere. For many, the story starts with the codebreakers of Bletchley Park in the United Kingdom during WWII, but we shall start our story in early nineteenth-century France …

Falmer, UK Kingsley Sage
January 2019

Contents

About the Author

Dr. Kingsley Sage is a Senior Teaching Fellow in Computing Sciences in the Department of Informatics at the University of Sussex, Brighton, UK, and a Senior Fellow of the Higher Education Academy (SFHEA). He has more than 20 years of teaching experience, from the level of further/continuing education through to postgraduate-level teaching, in both traditional university teaching and adult education.