|  |  |
| --- | --- |
| |  | | --- | | **Objects First with Java A Practical Introduction using BlueJ**  *This document describes the goals, contents and main pedagogical concepts of this book. It is largely identical with the preface in the book.*  *Contents of this description:*  [Java](http://www.bluej.org/objects-first/description.html#java) [BlueJ](http://www.bluej.org/objects-first/description.html#bluej) [Real objects first](http://www.bluej.org/objects-first/description.html#real) [An iterative approach](http://www.bluej.org/objects-first/description.html#iterative) [No complete language coverage](http://www.bluej.org/objects-first/description.html#coverage) [Project-driven approach](http://www.bluej.org/objects-first/description.html#project)  [Concept sequence rather than language constructs](http://www.bluej.org/objects-first/description.html#concept) [Third edition](http://www.bluej.org/objects-first/description.html#edition)  [Chapter sequence](http://www.bluej.org/objects-first/sequence.html)  This book is an introduction to object-oriented programming for beginners. The main focus of the book is general object-oriented and programming concepts from a software engineering perspective.  While the first chapters are written for students with no programming experience, later chapters are suitable for more advanced or professional programmers as well. In particular, programmers with experience in a non-object-oriented language who wish to migrate their skills into object orientation should also be able to benefit from the book.  We use two tools throughout the book to enable the concepts introduced to be put into practice: These tools are the Java programming language and the Java development environment BlueJ.  **Java**  Java was chosen because of a combination of two aspects: the language design and its popularity. The Java programming language itself provides a very clean implementation of most of the important object-oriented concepts, and serves well as an introductory teaching language. Its popularity ensures an immense pool of support resources.  In any subject area, having a variety of sources of information available is very helpful, for teachers and students alike. For Java in particular, countless books, tutorials, exercises, compilers, environments, and quizzes already exist, in many different kinds and styles. Many of them are online and many are available free of charge. The large amount and good quality of support material makes Java an excellent choice as an introduction to object-oriented programming.  With so much Java material already available, is there still room for more to be said about it? We think there is, and the second tool we use is one of the reasons ...  **BlueJ**  The second tool, BlueJ, deserves more comment. This book is unique in its completely integrated use of the BlueJ environment.  BlueJ is a Java development environment that is being developed and maintained at the University of SDouthern Denmark, Deakin University, Australia, and the University of Kent at Canterbury, UK, explicitly as an environment for teaching introductory object-oriented programming. It is better suited to introductory teaching than other environments for a variety of reasons:   * The user interface is much simpler. Beginning students can typically use the BlueJ environment in a competent manner after 20 minutes of introduction. From then on, instruction can concentrate on the important concepts at hand - object orientation and Java - and no time needs to be wasted talking about environments, file systems, class paths, DOS commands or DLL conflicts. * The environment supports important teaching tools not available in other environments. One of them is visualization of class structure. BlueJ automatically displays a UML-like diagram representing the classes and relationships in a project. Visualizing these important concepts is a great help to both teachers and students. It is hard to grasp the concept of an object when all you ever see on the screen is lines of code! The diagram notation is a simple subset of UML, again tailored to the needs of beginning students. This makes it easy to understand, but also allows migration to full UML in later courses. * One of the most important strengths of the BlueJ environment is the user's ability to create directly objects of any class, and then to interact with their methods. This creates the opportunity for direct experimentation with objects, for little overhead in the environment. Students can almost 'feel' what it means to create an object, call a method, pass a parameter or receive a return value. They can try out a method immediately after it has been written, without the need to write test drivers. This facility is an invaluable aid in understanding the underlying concepts and language details.   BlueJ is a full Java environment. It is not a cut down, simplified version of Java for teaching. It runs on top of Sun Microsystems' Java Development Kit, and makes use of the standard compiler and virtual machine. This ensures that it always conforms to the official and most up-to-date Java specification.  The authors of this book have several years of teaching experience with the BlueJ environment (and many more years without it before that). We both have experienced how the use of BlueJ has increased the involvement, understanding and activity of students in our courses. One of the authors is also a developer of the BlueJ system.  **Real objects first**  One of the reasons for choosing BlueJ was that it allows an approach where teachers truly deal with the important concepts first. 'Objects early' has been a battle cry for many textbook authors and teachers for some time. Unfortunately, the Java language does not make this noble goal very easy. Numerous hurdles of syntax and detail have to be overcome before the first experience with a living object arises. The minimal Java program to create and call an object typically includes:   * writing a class; * writing a main method, including concepts such as static methods, parameters, and arrays in the signature; * a statement to create the object ('new'); * an assignment to a variable; * the variable declaration, including variable type; * a method call, using dot notation; * possibly a parameter list.   As a result, textbooks typically either   * have to work their way through this forbidding list, and only reach objects somewhere around chapter 4; or * use a "Hello, world"-style program with a single static main method as the first example, thus not creating any objects at all.   With BlueJ, this is not a problem. A student can create an object and call its methods as the very first activity! Because users can create and interact with objects directly, concepts such as classes, objects, methods and parameters can easily be discussed in a concrete manner before looking at the first line of Java syntax. Instead of explaining more about this here, we suggest that the curious reader dip into Chapter 1 - things will quickly become clear then.  **An iterative approach**  Another important aspect of this book is that it follows an iterative style. In the computing education community, a well-known educational design pattern exists that states that important concepts should be taught early and often (The "Early Bird" pattern, in J. Bergin: " *Fourteen Pedagogical Patterns for Teaching Computer Science*", Proceedings of the Fifth European Conference on Pattern Languages of Programs (EuroPLop 2000), Irsee, Germany, July 2000). It is very tempting for textbook authors to try and say everything about a topic at the point where it is introduced. For example, it is common, when introducing types, to give a full list of built-in data types, or to discuss all available kinds of loops when introducing the concept of a loop.  These two approaches conflict: we cannot concentrate on discussing important concepts first, and at the same time provide complete coverage of all topics encountered. Our experience with textbooks is that much of the detail is initially distracting, and has the effect of drowning the important points, thus making them harder to grasp.  In this book, we touch on all of the important topics several times, both within the same chapter and across different chapters. Concepts are usually introduced at a level of detail necessary for understanding and applying the task at hand. They are revisited later in a different context, and understanding deepens as the reader continues through the chapters. This approach also helps to deal with the frequent occurrence of mutual dependencies between concepts.  Some teachers may not be familiar with an iterative approach. Looking at the first few chapters, teachers used to a more sequential introduction will be surprised about the number of concepts touched on this early. It may seem like a steep learning curve.  It is important to understand that this is not the end of the story. Students are not expected to understand everything about these concepts immediately. Instead, these fundamental concepts will be revisited again and again throughout the book, allowing students to get a deeper and deeper understanding over time. Since their knowledge level changes as they work their way forward, revisiting important topics later allows them to gain a deeper understanding overall.  We have tried this approach with students many times. It seems that students have fewer problems dealing with it than some long time teachers. And remember: a steep learning curve is not a problem as long as you ensure that your students can climb it!  **No complete language coverage**  Related to our iterative approach is the decision not to try to provide complete coverage of the Java language within the book.  The main focus of this book is to convey object-oriented programming principles in general, not Java language details in particular. Students studying with this book may be working as software professionals for the next 30 or 40 years of their life - is a fairly safe bet that the majority of their work will not be in Java. Every serious textbook must of course attempt to prepare them for something more fundamental than the language flavor of the day.  On the other hand, many Java details are important for actually doing the practical work. In this book, we cover Java constructs in as much detail as is necessary to illustrate the concepts at hand and implement the practical work. Some constructs specific to Java have been deliberately left out of the discussion.  We are aware that some instructors will choose to cover some topics that we do not discuss in detail. That is expected and necessary. However, instead of trying to cover every possible topic ourselves (and thus blowing the size of this book out to 1500 pages), we deal with it using hooks. Hooks are pointers, often in the form of questions that raise the topic and give references to an appendix or outside material. These hooks ensure that a relevant topic is brought up at an appropriate time, and leaves it up to the reader or the teacher to decide to what level of detail that topic should be covered. Thus, hooks serve as a reminder of the existence of the topic and a placeholder indicating a point in the sequence where discussion can be inserted.  Individual teachers can decide to use the book as it is, following our suggested sequence, or to branch out into sidetracks suggested by the hooks in the text.  Each chapter also includes several questions suggesting discussion material related to the topic, but not discussed in this book. We fully expect teachers to discuss some of these questions in class, or students to research the answers as homework exercises.  **Project-driven approach**  The introduction of material in the book is project driven. The book discusses numerous programming projects and provides many exercises. Instead of introducing a new construct and then providing an exercise to apply this construct to solve a task, we first provide a goal and a problem. Analyzing the problem at hand determines what kinds of solutions we need. As a consequence, language constructs are introduced as they are needed to solve the problems before us.  Almost all chapters provide at least two discussion examples. These are projects that are discussed in detail to illustrate the important concepts of each chapter. Using two very different examples supports the iterative approach: each concept is revisited in a different context after it is introduced.  In designing this book, we have tried to use a large number and wide variety of different example projects. This will hopefully serve to capture the reader's interest, but it also helps to illustrate the variety of different contexts in which the concepts can be applied. Finding good example projects is hard. We hope that our projects serve to give teachers good starting points and many ideas for a wide variety of interesting assignments.  The implementation for all our projects is written very carefully, so that many peripheral issues may be studied by reading the projects' source code. We are strong believers in the benefit of learning by reading and imitating good examples. For this to work, however, one must make sure that the examples students read are well written and worth imitating. We have tried to do this.  All projects are designed as open-ended problems. While one or more versions of each problem are discussed in detail in the book, the projects are designed so that further extensions and improvements can be done as student projects. Complete source code for all projects is included. A list of projects discussed in this book is provided in the book.  **Concept sequence rather than language constructs**  One other aspect that distinguishes this book from many others is that it is structured along fundamental software development tasks and not necessarily according to the particular Java language constructs. One indicator of this is the chapter headings. In this book, you will not find many of the traditional chapters titles, such as 'Primitive data types' or 'Control structures'. Structuring by fundamental development tasks allows us to give a much more general introduction that is not driven by intricacies of the particular programming language utilized. We also believe that it is easier for students to follow the motivation of the introduction, and that it makes it much more interesting.  As a result of this approach, it is less straightforward to use the book as a reference book. Introductory textbooks and reference books have different, partly competing, goals. To a certain extent, a book can try to be both, but compromises have to be made at certain points. Our book is clearly designed as a textbook, and wherever a conflict occurred, the textbook style took precedence over its use as a reference book.  We have, however, provided support for use as a reference book by listing the Java constructs introduced in each chapter in the chapter introduction.  For a brief description of each chapter and the sequence of projects, [look here](http://www.bluej.org/objects-first/sequence.html).  **Third edition**  This is the third edition of this book. Several things have been changed from previous editions. The second edition saw the introduction of JUnit and a chapter on GUI programming. In this edition, the most obvious change is the use of Java 5 as the implementation language. Java 5 introduced new language constructs, such as generic classes and enumeration types, and almost all of our code examples have been changed to make use of these new features. The discussions in the text have, of course, also been rewritten to take account of this. Overall, however, the concept and style of this book remain unchanged.  Feedback we received from readers of prior editions was overwhelmingly positive, and many people have helped in making this book better by sending in comments and suggestions, finding errors and telling us about them, contributing material to the book’s web site, contributing to the discussions on the mailing list, or translating the book into foreign languages.  Overall, however, the book seems to be ‘working.’ So this third edition is an attempt at improvements in the same style, rather than a radical change.  [Next: A short description of each chapter](http://www.bluej.org/objects-first/sequence.html) |   [Back to main page](http://www.bluej.org/objects-first/index.html) |