# Chapter 1 Introduction

In this book you will enter the exciting world of computer programming. Specifically, you will learn to thrive in the environment created by a special piece of software named "jBoxes." This software was designed and created to help give you a solid foundation for further learning about programming in Java and related material in computer science. You will learn to create relatively simple programs in this book, but that is not its main purpose.

The main goal of this book and the jBoxes software is to provide you with a *semantic model* for a Java-like language—a detailed, concrete conceptual picture of how the computer behaves when it performs the instructions you give it in a program. Once you have worked through this material, you will understand all the tools at your disposal in creating programs, and you will understand exactly how the computer is behaving, which is invaluable when you need to figure out why your programs are not working properly.

So, even if you have already had some exposure to programming, this material will be helpful to your deeper understanding. And, if you have never done any programming at all, it will provide you with a gentle, nurturing introduction to the basic concepts of programming.

In addition to developing a semantic model, you will also begin to learn how to program as you work through this book. Throughout the book you will be asked to create your own programs, but they will all be fairly simple, and closely related to the example presented in the book.

## Approaches to Learning

When you need to teach something to someone, you have to explain certain information, and you need to have a strategy for the order in which you present the pieces of this information. If you use a so-called "bottom-up" approach, you try to explain the simplest details first and gradually work yourself up to the big picture. This is the so-called "trust me" approach, where you ask the learner to trust that eventually all these details will build up to something useful and interesting.

For example, suppose you were trying to teach someone how to make things out of wood. Using the "trust me" approach, you would explain how to measure a desired length on a piece of wood, marking it with a pencil, and then how to use a saw to cut the board at that spot, and then how to drill a hole in a piece of wood, and then how to insert a screw through two pieces of wood, through the holes you had drilled, thus fastening them together. At that point, in some theoretical sense, you would have taught your student pretty much all the *facts* needed for this kind of carpentry, but they would probably still have trouble when you asked them to build, say, a doghouse. One reasonable thing to do at that point would be to show them a doghouse that someone else had built and ask them to make a similar doghouse. They could probably do that, and feel pretty good about themselves, but if you now asked them to build a radically different project, say some bleachers for a sports field, without showing them a version of the object, they would probably have

trouble, because you still would not have taught them anything about designing a new object.

This metaphor applies to your initial work in this book. For quite a while, you will be learning a zillion little details, without much idea of how they can be used to create a useful program. You just need to have faith that all these little pieces will come together.

Another approach to teaching is the so-called "just in time" approach. Using this approach, you present your students with a goal and ask them what they need to know or be able to do in order to accomplish that goal. Then you teach them those things.

With this approach, you would start teaching about building things with wood by saying, hey, let's design and build a shelter for a dog. After some discussion, the students would decide that a good shelter has to keep rain, sun, and snow off the dog, and after considerable thought, they would figure out that they needed to have a bunch of boards, side-by-side, forming a flat surface for the roof. Then they would say, but wait, at the lumber store all the boards are 8 or 10 or 12 feet long, and we need boards that are, like, three feet long. At that point, they would beg you, their teacher, to explain to them how to cut a long board into shorter pieces, and then you could show them how to use the measuring tape, pencil, and saw. The learning process would continue in this way, with the students realizing what they needed to know, and being able to put their new knowledge into context.

> **Exercise 1.1: Building Bleachers**
>
> As a demonstration of this idea, think about how you would build bleachers, and realize that the supposedly complete set of information about carpentry mentioned earlier is missing some crucial facts.

This "just in time" approach is probably more interesting, more motivating, and helps to develop design and problem solving abilities much better than the "trust me" approach. So, why does this book predominantly use the "trust me" approach? Well, imagine that instead of teaching humans to build things out of wood, you were working with a class of genetically-engineered super-intelligent dolphins. If you asked them to design and build a shelter for a dog, they would have some serious difficulties, such as, for example, not knowing what a dog is. So, fine, you could take them to the beach and show them some dogs strolling around, and then what? How long would it take them to realize that dogs like to be dry, and warm, but not too hot, and that therefore the shelter they were building would need a roof? You would have a lot of low-level detail to explain to these dolphins, and you might be tempted to present it in a bottom-up approach.

Similarly, faced with the goal of learning to program in Java, this book will begin by presenting, in a totally "trust me" way, a bunch of information about the world of jBoxes, just to get you "up to speed" on the strange new world of the programmable information processing machine. Then you'll be ready to start learning the deeper material, in the later chapters, following a "just in time" approach to a much greater extent.

This whole preceding discussion may have not made much sense, perhaps partly because you are so accustomed to being taught in a "trust me" way that you don't realize how really bad it is!

One thing that will make the "trust me" approach to first part of the book a lot more meaningful is that the jBoxes software is designed, as described later, to allow you to learn all about it through thoughtful observation. Actually, just by working with the jBoxes software and being especially observant and insightful, you could figure out for yourself all the facts that will be imparted to you in the "trust me" part of the book!

## What is Programming?

The term "programming," as it is used here, simply means giving instructions to some sort of machine that can then perform, or *execute* those instructions. For example, when you punch buttons on a microwave oven, selecting a power level and a time length, and then hit the start button, you are programming the appliance to behave independently for a time, working away on its own to accomplish some task. One reason this book will use the "trust me" approach at the beginning is that, like the dolphins, no matter how brilliant you are, you may be totally unfamiliar with the miraculous type of machine that can be given instructions and then work independently.

A computer is actually an amazing machine, known by computer scientists as a "universal machine," because you can program it to perform, under its own guidance, any task that it is physically capable of performing. If you think about it, it is really amazing that you can buy one machine, a computer, and use it one minute to produce a document and a minute later to play a game. It's as if you could go to the store and buy a "kitchen appliance" that was a single piece of equipment sitting in your kitchen that could wash your dishes, cook your food, and keep your leftovers cold. This is very unlikely in the physical world, but in the world of information processing that you are entering, it is quite natural.

The jBoxes machine is especially nice for introducing programming because it is a visible environment, in the sense that you can see everything that it is doing, if you want to. Later, when you go on in your software development career to work with a Java machine, you will find out that you can't see anything that is happening when a program is executing except for the results of instructions that tell the machine to display some information. Note that even though jBoxes and Java are apparently quite different, they are really very similar. In order to provide visibility, jBoxes is diagrammatic rather than textual, but a lot of the instructions you write in various parts of a jBoxes diagram are exactly the same as the instructions in a corresponding Java program, and the semantic model of jBoxes is exactly a simplified version of the semantic model of Java. In other words, everything you learn about jBoxes will immediately apply when you move on to learning about Java.

The language in which instructions can be given to a particular machine is known as a *programming language*. Java is a currently popular programming language, but literally thousands of other programming languages have been developed and used. The language that a particular machine actually understands is known as its *machine language*. Every computer has a single machine language which is the language built into its central processing unit. To use a language other than its own machine language on a computer, you need to translate the program into an equivalent machine language program using a program, written in machine language, that is known as a *compiler*, or translator. Or, you

can execute a special program, written in machine language, known as an *interpreter*, that looks at the program symbolically and simulates execution of the steps it is requesting.

In the last chapter of this book, you will learn exactly how all this works in the context of the Java machine.

The programming environment known as jBoxes is an interpreter for diagrams. When you program in jBoxes, you make diagrams, with embedded symbols, that represent instructions to the jBoxes machine. The jBoxes machine is simulated by running a machine language program known, for example on a Windows machine, as `jboxes.exe`.

## Installing jBoxes

Since the entire purpose of this book and the jBoxes software is to give you a solid, detailed foundation for future work in software development, the process of installing the jBoxes software will not be done automatically. Instead, you will be required to copy files manually, so that you will know exactly what is going on—nothing mysterious and inexplicable will be done behind your back.

To install the jBoxes software on a Windows machine, simply go to the folder on the CD named `jBoxes` and copy it into some suitable folder on your hard disk.

> **Exercise 1.2: Viewing the Installed Files**
>
> Examine the files in the folder named `jBoxes` that is now on your disk. You will find a file named `jboxes.exe` which is the main machine language program for the jBoxes system. You will also find a rather strange file named `glut32.dll`. This file contains a collection of graphics utility routines, again written in machine language. Finally, notice that the folder contains a lot of files with the suffix `.box`. These are the various jBoxes universes that you will be working with through the rest of this book.

### Starting jBoxes

To start the jBoxes system, simply go to the file named `jboxes.exe` and double-click on it.

The jBoxes program will begin executing and will ask you for the name of the universe you want to work with. Just type the name—without the `.box` part—and hit enter.

> **Exercise 1.3: Testing Your Installation**
>
> To test whether you have things set up correctly, and to see jBoxes in action for the first time, start the program on the file `counting.box` (remember that you do not type the `.box` part of the universe name.
>
> Once the window showing part of the universe opens up (you may need to click on the icon on the task bar, or do something else for your operating system to get the window to show in front), you will see part of the diagram for the jBoxes universe. When you see this diagram, you will know that you have the software working.

Now stop jBoxes in the proper way by pressing the F6 key. This saves any changes you may have made in the universe and then exits the program. Later, when you are actually making changes, you can, and definitely should, hit the F5 key frequently to save your work.

If you ever realize that you want to exit the system without saving any changes you have made—typically when you realize that you have accidentally messed things up and want to not make those mistakes permanent—you can left-click on the "X" in the upper right corner of the jBoxes window. Of course, if you leave jBoxes this way when you meant to save your work first, you may deeply regret it—jBoxes does not ask you if you want to save, it just quits.