

## Introduction

The purpose of this book is to teach you how to program in Java. Learning to program can be hard. Learning to program really well in a language as robust as Java can be really hard. Or, it can be fairly easy. Your attitude can make all the difference.

Java is grounded in object-oriented programming. Objects are a meta-cognitive concept. While that may sound like gobbledygook, understanding what meta-cognitive means is key to this book. The word cognitive refers to thinking. A cognitive process is simply how you think about something. When you play a game, you usually come up with a strategy to win the game (assuming that you *want* to win). That strategy is a cognitive process. Meta-cognitive is one step beyond cognitive, it is thinking about how to think. Tic-tac-toe is a game most children learn in school. It doesn't take long for most people to learn that there is no winning strategy for tic-tac-toe. That is, if both players make the proper moves, every game will end in a draw. The concept of un-winnable games is a meta-cognitive idea.

Piaget, a noted educational theorist, devised a list of stages in cognitive development. He observed children over time and concluded that most children start becoming meta-cognitive around 12 to 14 years old. That seems to be minimum age to learn how to program in Java because of the meta-cognitive nature of objects. While younger children can learn to program, it is our belief that only those who have entered the meta-cognitive stage of intellectual development should attempt to do so.

While meta-cognitive ability is essential to programming, educational researchers over the years have found that manipulatives can significantly improve learning. A manipulative is simply a physical thing that is essential to the instructional strategy (pedagogy). For example, some teachers use counting sticks to help young children learn their addition facts. Having a physical artifact helps many young children learn a fairly abstract body of knowledge. Similarly, in high school biology classes, dissections are an example of a manipulative used with older students.

For this book, the Lego™ Mindstorms™ RIS is the essential manipulative. Throughout this book, there is example code that uses the RIS. It is expected that the student will build robots and program them as part of the course of study. Without the activity of building and programming, the results of using this book is likely to be very poor. Further, the activities through the book build sequentially. For example, the first chapter has nothing to do with Java programming. It is about building a particular robot, Roverbot, and using the RIS programming environment to control it. This seems to have little to do with learning Java. However, it ensures that you have 1) a working IR link, 2) a working robot and 3) familiarity with robot commands that you will implement later in Java.

Learning to program requires developing skills in using programming tools and skills in learning how to think in programming terms. These are two different types of skills. Becoming proficient in using programming tools makes it much easier to do your own

programming. In the beginning there will be a lot of programming tools skill development exercises. You will be given correct, or nearly correct, programs to compile and run. Before you actually begin writing programs you will be correcting other people's programs. While this may seem paradoxical, how can you correct programs before you know how to program, this technique is known as *scaffolding* and will provide you with the tools for successfully programming robots.

One keyboarding technique you will be using is cut and paste. If you have ever used a word processor and a web browser, you are likely to be very familiar with this practice. Cutting and pasting can lead to plagiarism, the practice of claiming the work of others as your own. This is unethical and in school situations likely to result in disciplinary action. However, in programming it is common practice to search the web to find an example of what you want to do and include it in your own code. In my view, this is perfectly acceptable. In coding, as opposed to writing a report, there are frequently only a few "good" ways to accomplish a task and sometimes there is a "best" way. It makes no sense to write bad code when good code is freely available. This does not mean blindly copying code from the web is acceptable. It does mean analyzing your problem, using the web as a source of knowledge and using that knowledge base is appropriate.

