Before You Begin

Please read the following topics before you begin the course.

This Course is for Hands-On Students!

This course was designed for 9th-12th grade students by experienced software professionals. We have created a program that will give students a solid, practical foundation in the field of computer programming. Lessons begin with basic concepts, including sample code, and move quickly to hands-on implementation. Students will enjoy writing their own programs as they progress through the course.

Each chapter includes at least one activity that will allow the student to demonstrate the concepts that they have learned. Beginning in Chapter 2, these activities will have students writing and running their very own Java programs. These activities will start small and move up in complexity as the course progresses. At the end of the course, students will have the opportunity to apply what they have learned by completing a final project: a board game based on an ancient chess-like Viking game!

What Do Students Need to Start?

Students will need to meet the following requirements in order to successfully complete the course:

- Students should have the pre-requisite computer skills listed in the Student Textbook
- Students should have a Windows-based or Mac-based computer that meets the minimum hardware and software requirements listed in the Student Textbook
- Students will need an Internet connection to complete the download and installation of the (free) Java and Eclipse software during the Chapters One and Chapter Three. An Internet connection is optional for the rest of the course.

What are "Bonus" Lessons?

There are three "bonus" lessons referenced in Chapters Two, Four, and Nine. Casual students can skip these lessons without impact to the rest of the course. Students who wish to take the "AP* Computer Science A" Exam must complete all bonus lessons and other tasks outlined in our "Preparing for the AP Exam" documentation. See the "Bonus Material" in the Student and Solution Menus for more information!

* "AP" and "College Board" are registered trademarks of the College Board, which was not involved in the production of, and does not endorse, this product.

Why Did We Choose the "Java" Programming Language?

There are several major programming languages a student can learn, including the Visual Basic and C# languages we teach in other KidCoder™ and TeenCoder™ curriculum. The following advantages apply to the Java language your student is learning now:

- Java is widely used in academics and businesses today, with a large active community of programmers.
- Java is modern and powerful, yet easy to use.
- Java is free to install on your computer!
- High-quality development software is freely available.
- Java is cross-platform, meaning it will work on Windows, Apple Mac OS, and other platforms.
- Java can be used for many interesting tasks, such as writing applications for Android™ mobile phones as you will see in our second-semester *TeenCoder: Android Programming* course!

What Do Teachers Need to Do?

The course is a self-study curriculum where students can simply read the textbook, complete the activities, and refer to this Solution Guide to clear up any questions about the activities. Teachers may choose to have greater involvement if they wish to administer the course for a more formal grade or school credit. Some students, especially younger students, may need additional assistance.

Grading a student's progress is done in two ways. First, each chapter has a hands-on activity which should be completed by the student. These activities are the focal point of the course and we recommend assigning at least 50% of the course grade to the activities. Guidance on grading activities is given below. The remainder of the course grade can be derived from the 10-question multiple choice tests for every chapter.

Please see the section on "Evaluating Student Progress" for more details!

How Do I Get Help?

If you have questions or concerns about any of the activities, solutions, or tests, please contact us according to the instructions on our website. We will provide responses to your questions as soon as possible. Please also check the Errata section of our website for any reported corrections or clarifications to this edition.

Evaluating Student Progress

If teachers wish to grade student progress, they will be responsible for administering tests and evaluating student programs.

Administering Tests

Chapter tests can be found in a "Tests" directory underneath the root folder where you installed the Solution Files. You can access the tests from the Solution Menu or by browsing directly to the "TeenCoder\Java Programming\Tests" folder using Windows Explorer or Mac OS Finder. Tests are in Adobe Acrobat .PDF format for easy printing on a home printer. You will also find a complete answer key for all tests in another .PDF file. Test questions are multiple-choice.

How Do I Evaluate Student Programs?

If you do not have a background in programming or computers, you may be wondering how to evaluate your student's progress. But don't worry! This course is designed primarily for student self-study; your level of involvement depends on your interest in the material. This Solution Guide includes easy-to-understand activity solutions. If your student is stuck on any activity you can review the provided solution with them. Each solution is written so that a novice computer user can understand and evaluate a student's progress.

Evaluating a student's program is like grading an art project. The process can be very subjective. To make it easier, follow these guidelines:

- Have your student build and run the program on a computer. Check to make sure that the program performs all the tasks as outlined in the activity requirements.
- Have your student turn in a printed copy of the source code.
- Check this printed copy for the key elements that are mentioned in the activity solution. Note that all solutions will be slightly different as there are many ways to achieve the same ends through code.
- Finally, have the student walk through the printed program and the computer program with you. If they can explain how the program works to you, they understand it well enough to have passed the activity.

While each student's program will look slightly different than the provided solution, the program outputs and behavior should meet the requirements specified in the activity. We clearly point out all of the elements to look for in every activity description.

Activity Solutions

The next chapters contain complete solutions for each Student Textbook activity. Each activity solution will generally contain:

- A picture of the screen the student will produce
- The complete source code the student is responsible for writing.

When installing the "Solution Files" from the setup program, your computer will also receive fully coded activity solution projects. If you have installed the Eclipse software on your computer (just as done during the student's first activity) then you can review the solutions using the same Integrated Development Environment (IDE) as the student. The student will spend most of their hands-on time within this Java coding software, so teachers may also wish to become familiar with the essentials of the development environment.

The activity solutions contain an Eclipse "workspace" holding all of the projects. When you launch Eclipse, simply select the "TeenCoder\Java Programming\Activity Solutions" directory as your workspace. Eclipse should then display all of the activity solutions in the IDE. Refer to Chapter Three in the Student Textbook for a description of how to use Eclipse to build and run a program.

If you do not have the Eclipse software installed, you may still view the solution files using any text editor! Simply open Windows Notepad, Mac TextEdit, or your preferred text editor and navigate to the "\TeenCoder\Java Programming\Activity Solutions\<project>" directory to find the activity solution folder. Open the *.java (e.g. "RaceCar.java") source file in the \src subdirectory into your text editor for review.