# teacher'S guide

# Lesson 01 – Introduction to Object-Oriented Programming (OOP)

**OBJECTIVES:** The student will learn the basics of classes and objects, the basis of

Object-Oriented Programming (OOP).

The student will learn how to invoke the methods of objects by sending messages to objects.

The student will see how a program is created using objects.

The student will compile and run a simple program that uses objects.

**ACTIVITIES/TIME:** **Note:** The allocation of time in these lessons total about 24 weeks to cover the A level material in one year. This allows for 4-5 weeks for the APCS Marine Biology Simulation Case Study (see Appendix) and review for the exam. The AB material is also written for a year; to complete both the A and AB material in one year you should reference an alternative schedule in the Appendix.

One week

**MATERIALS:** Student Lesson A1: *Introduction to Object-Oriented Programming*

Handout A1.1, *DrawingTool Class Specifications*

Lab Assignment A1.1, *DrawHouse*

Worksheet A1.1, *Object-Oriented Programming*

Teacher's Guide, Lesson A1: *Introduction to Object-Oriented Programming*

Lab Assignment A1.1 - Answers, *DrawHouse.java*

Worksheet A1.1 – *Answer Sheet*

Code from Lesson A1 - *DrawSquare.java*

**REFERENCES:** **Java Tutorial on Object-Oriented Programming Concepts**  
<http://www.java.sun.com/docs/books/tutorial/java/concepts>

**Introduction to Computer Science Using Java**

<http://chortle.ccsu.edu/CS151/cs151java.html>   
  
  
**Focus on Java**  
<http://java.about.com/>

With great resources for beginners, including a tutorial, Java glossary, Java keywords, and recommendations for Java editors.

**INSTRUCTOR**

**NOTES:** This lesson has been allocated for one week. This is to allow you to introduce the course and get your students acclimated to the lab setup. It’s a good idea to have students practice getting onto their computers and saving files, logging off, getting back in, etc. If your class does not take place in a computer lab, you should discuss how to install Java so your students will be able to setup their own computers correctly.

Lesson A22, *General Computer Knowledge* covers basic facts about computers: hardware vs. software, primary vs. secondary memory, operating systems, programming languages, source code and object code. If you feel that it is necessary, you may introduce this material first. It is suggested that you wait until the students are a little bit familiar with programming and computers so that they will understand and appreciate this information more. However, this chapter can be taught at any time.

Lesson A1 introduces the student to the world of Object-Oriented Programming (OOP). Role-playing is a good technique to help the students understand objects and messages. This technique is effective for topics throughout the course, especially the Case Study. There are quite a few free role-plays referenced from AP Central. It is a good idea to start your students off with easy role-plays and get them used to them. For example, the teacher or a student can play the role of an object, such as a DrawingTool. The other students can send the object “messages” that are used to draw a figure on the blackboard. Software professionals use this kind of role-playing to understand the objects of a system they are designing. An analogy that helps students understand the difference between objects and classes is the blueprint. A class is like a blueprint. From a single blueprint, many houses (objects) can be built. Each house can have different attributes such as address, style, or color.

It is very important to have the students learn the correct terminology and vocabulary from the very first day of class. These words will be used during the entire course, and setting a standard list very early on will help the students to remember the definitions. Many programmers across different specialized fields and using different programming languages may use different words for essentially the same thing. At the beginning of their programming education, it is important for students to learn a single verbiage.

Lesson A1 begins immediately with creating and running a program so you will need to help the students with installation if necessary. See the Installation Setup Guide in the Appendix.

The HTML Web-formatted lesson files can be used as presentation material or given to the students as an online reference. It is very useful, if you have the facilities, to work with the students as a whole to demonstrate how to open a project and type code in, compile it, and run it. If the students are shown directly how to work with their IDE and what well-structured code looks like, then they are much more likely to be able to produce solid code as well. The DrawSquare project is an excellent place to have the whole class work together. With two files as a starting point - the class and the driver that contains the main - work through every line and show the students exactly what is going on. Strongly reinforce right at the start that main methods should be left out of normal classes and only used in driver classes.

The first few weeks are also a good time to get students thinking about solving problems - not just computer programming problems, but all kinds of problems. Puzzle-solving skills can be reinforced by referring students to brainteaser puzzles such as can be found in newspapers or in puzzle books. They can be assigned for early homework and group assignments to improve students' general skills, and help them to solve more complex programming problems. Online discussion boards are also an excellent resource for selecting discussion topics on issues and problems that most students face. Current events related to technology and computers are very good ways to get students interested and to understand real world applications of what they are learning.

**WORKSHEET**

**NOTES:** Worksheet A1.1, *Object-Oriented Programming*, gives students practice problems using DrawingTool and SketchPad, allowing them to use objects and methods in a simple way. The practice of using several methods for drawing (forward, turnRight, turnLeft, etc.) is used in question # 1 and in the beginning of question # 2. The more efficient *move* method is then introduced in question # 2. With questions # 1 and # 2 (and in future worksheets), students are asked to determine output given certain code segments in Java. Encourage students to resist the temptation to use copy-paste-compile to obtain answers. Following the logical flow of the code to obtain output will help them much more in the long run! Questions # 3-5 ask students to write code that uses DrawingTool and incorporates familiar geometric concepts. It is suggested that students type in the code after they have written what they think the code will do.