Lesson plan for Niño and Hosch’s
*Programming & Object Oriented Design using Java*

This lesson plan matches the chapters of *Programming & Object Oriented Design using Java*, by Jaime Niño and Frederick A. Hosch, with activities in *ProgramLive*. Each unit of the lesson plan corresponds to a chapter. Different authors of programming texts introduce material in different orders and emphasize different concepts, so the match between Niño-Hosch and *ProgramLive* is not exact.

Below, we give an overview of each unit together with a checklist for the activities in it. Check each one off as you complete it. But first:

- An activity or lab that is labeled “optional” is in *ProgramLive* but not in Niño-Hosch.
- Niño-Hosch do not discuss the running of Java programs on a computer, leaving that detail to an instructor. *ProgramLive* has tutorials on the Interactive Development Environments Visual Cafe (lesson 18) and CodeWarrior (lesson 19), and Appendix B of this Companion discusses the use of the UNIX command-line system for compiling and running Java programs.
- It is possible to skip lesson 0 of *ProgramLive* on how to use the livetext, but you will save time if you spend half an hour on it. In addition to the activities, there are a plethora of instructional tools such as the glossary, index, exercises, and labs.

- Introduction to livetexts
- Activities
- The lesson book page
- Global features
- Page controls
- Dealing With Java programs
- Learning effectively

Unit 1. Introduction

This chapter of Niño-Hosch goes into great detail on some issues that are only moderately covered, or not at all, in *ProgramLive*’s introduction. Object-oriented systems, for example, are not covered in the introduction of *ProgramLive*. 
Niño-Hosch, Secs. 1.1–1.7, p. 1–22. What is computer science, what is a software system, object-oriented systems, a model of a computer system, software tools, errors in the programming process

Hardware and software  
This provides a brief look at hardware and software, a more in-depth treatment of the computer and memory, and a brief introduction to software tools such as programming languages and compilers or interpreters for them.

Unit 2. Data abstraction: introductory concepts

Niño-Hosch, Sec. 2.1, p. 27. Values and types

Variables and types (activity 3)  
Values, types, variables, and assignments to them are discussed.

Niño-Hosch, Sec. 2.2, p. 28. Primitive types in Java

Overview of primitive types  
Use lesson pages 6-2–6-6 as a reference for information on primitive types.

Niño-Hosch, Secs. 2.3–2.4, p. 30–39. Objects and Classes

Niño-Hosch introduces objects and classes in an abstract way, without reference to corresponding constructs of Java, while ProgramLive introduces objects and classes and the corresponding Java features at the same time. Because of these fundamentally different approaches, it does not make sense to list ProgramLive activities corresponding to these sections.

Unit 3. Basic Java structural components

Niño-Hosch, Sec. 3.1, p. 47. Syntactic structure of a system definition in Java

A do-nothing Java program (activity 1)  
Pay special attention to ProgramLive’s filing cabinet metaphor to help you understand what a class is.

Packages  

Niño-Hosch, Sec. 3.2, p. 49. Identifiers

Components of a Java program (activities 3 and 4)  

Naming conventions (last activity)  
Also, read the basic guidelines and at the top of the page.

Niño-Hosch, Sec. 3.3, p. 53. Literals
Lesson plan for Niño & Hosch’s Programming & Object-Oriented Design

- Do the first activity of these Lesson pages: 6-2 (Integral types), 118
  6-3 (A minimalist view of floating-point), 120
  6-5 (Type char), 121
  6-6 (Type boolean), 123
  5-3 (Strings), 106

- Niño-Hosch, Sec. 3.4, p. 55. Lexical structure

- Components of a Java program (activities 1 and 2)  
  PL Lesson 1-3, page 47

Unit 4. Specification of a simple class

This chapter of Niño-Hosch provides an overview of the process of specifying a class, using three examples. The process results in a Java class in which the bodies of methods have not yet been written. Along the way, Java syntax for instance variables, instance methods (constructors, functions, and procedures), and method calls is introduced, but not in full detail.

ProgramLive uses a different order: (1) methods and their specification are studied in full, (2) classes are thoroughly explored, and (3) an example of class specification is given. This difference in approach results in some problems in matching ProgramLive to Niño-Hosch.

There is a difference in terminology. ProgramLive considers three kinds of methods, the latter two being traditional in mathematics and older programming languages:

- constructor: a method for initializing fields of a class;
- function: a function call is an expression, and the call produces a result;
- procedure: a procedure call is a statement (or command), and the call does not produce a result.

Niño-Hosch uses method and function interchangeably and uses query for a function call and command for a procedure call.

- Niño-Hosch, Sec. 4.1–4.4, p. 59–84. Object specifications and Specifying a class

- Classes and objects (first three activities)  
  PL Lesson 3-3, page 79

- Methods (first three activities)  
  PL Lesson 2-1, page 61
  These activities relate methods to recipes and view a method as a black box—only its specification can be seen.

- Do Lab PGL-2 (Understanding method calls) of this lesson.  
  PL Lesson 2

- Do Lab PGL-2 (Drawing objects) of this lesson.  
  PL Lesson 3

- Functions versus procedures (only the first activity)  
  PL Lesson 2-4, page 70
  A function must return a result; a procedure cannot.
Unit 5. Implementing a simple class

We now look at the implementations of methods. This requires us to introduce declarations of variables and to discuss certain statements and expressions.

- Niño-Hosch, Sec. 5.1, p. 91. Implementing data descriptions
  - Variables and types (only activity 5)  
  - A class as a type (only the last activity)
  - Hiding instance variables (only the last three activities)
  - Constants (only the second activity)
  - Naming conventions (only activities 3 and 4)
  - Describing instance variables (only the first activity)

- Niño-Hosch, Sec. 5.2.1, p. 97. Method implementation: simple queries
  This section defines “statement”, says that the body of a method is a sequence of statements, and illustrates the use of a return statement in a function body, with the value of a simple variable being returned. ProgramLive introduces all these notions in different places, and we cannot point to any place that would make sense at this time.

- Niño-Hosch, Sec. 5.2.2, p. 100. Arithmetic expressions
  - Expressions (only activities 5 and 6)
  - Operations on type int (only activity 2)
Lesson plan for Niño & Hosch’s *Programming & Object-Oriented Design*

- Do Lab PGL-1 (Integral types) of this lesson.  
  *PL Lesson 6*
- Operations on type **double** (only activity 2)  
  *PL Lesson 6-3, page 120*
- Casting (only activities 3 and 4)  
  *PL Lesson 6-2, page 118*
- Casting (read only the end of the page)  
  *PL Lesson 6-3, page 120*
- Do Lab PGL-2 (Casting among integral types) of this lesson.  
  *PL Lesson 6*
- Look in the *ProgramLive* Glossary under *precedence* for information about operator precedence.
- Catenation of **Strings** (only activity 3)  
  *PL Lesson 5-3, page 106*
- Niño-Hosch, Sec. 5.2.3, p. 105. Method implementation: simple commands
  Niño-Hosch discusses only the assignment statement in this section.
- The assignment statement (activities 1 and 2)  
  *PL Lesson 1-4, page 51*
  Read also the discussion of the statement-comment after these two activities.
- Do Lab PGL-1 (The assignment statement) of this lesson.  
  *PL Lesson 1*
- Niño-Hosch, Sec. 5.2.4, p. 107. Using parameters
  *PL Lesson 2-2, page 64*
- Niño-Hosch, Sec. 5.2.5, p. 112. Invoking a method
  Invoking a method:  
  - activities 4, 5 on procedure calls, Lesson page 2-1, 61
  - activity 2 on function calls, Lesson page 2-4, 70
  - activities 1–3 on nonstatic methods, Lesson page 3-6, 87
- Niño-Hosch, Sec. 5.2.6, p. 121–124. Local variables
  Local variables (only the first activity)  
  *PL Lesson 2-3, page 67*
- Statement-comments (only activities 3–5)  
  *PL Lesson 13-4, page 190*
  The “statement-comment” is not a Java construct but a convention for allowing more “abstraction” in the documentation of a program. It is not discussed in Niño-Hosch. These activities explain why the statement-comment is so useful.
- Niño-Hosch, Sec. 5.3, p. 121. Testing an implementation
  Introduction to testing and debugging  
  *PL Lesson 14-1, page 193*
  In addition, you might want to look at GUI *JLiveWindow*, which is discussed in activity 3 of Lesson page 1-5 (see *Companion* page 56), as a driver to test some programs.
Unit 6. Conditions

A *conditional statement* is used to choose one of two alternatives. The choice depends on the value of a *condition*, which is a *boolean expression* (an expression that yields *true* or *false*). Boolean expressions are also used also in making assertions about a program.

☐ Niño-Hosch, Sec. 6.1, p. 131. Conditional statements

☐ Assertions in programs

☐ Do Lab PGL-6 (Relations) of this lesson.

☐ Do Lab PGL-1 (Assertions) of this lesson.

☐ Conditional statements (only activities 3 and 4)

☐ Blocks, or compound statements (only activity 6)

☐ Do Lab PGL-2 (The if-statement) of this lesson.

☐ Do Lab PGL-1 (The if-else-statement) of this lesson.

☐ Niño-Hosch, Sec. 6.2, p. 141. Boolean expressions

☐ Boolean expressions (only activity 5)

☐ Boolean expressions

☐ Do Lab PGL-4 (Boolean expressions) of this lesson.

☐ String equality

Unit 7. Programming by contract

This chapter discusses a programming style in which “the invocation of a method is viewed as a contract between a client and server, with each having explicitly stated responsibilities”. *ProgramLive* also takes this view, but it doesn’t devote a lesson specifically to it. Instead, the view pervades *ProgramLive*, and almost no method is given without providing a precise specification as a comment. Here are the activities that deal with this topic.

☐ Assertions (only activity 3)

☐ Guidelines for writing methods (only activities 1–2)

☐ Top-down programming
Unit 8. Testing a class

☑ Niño-Hosch, Sec. 8.1, pp. 183. Testing
☐ Introduction to testing and debugging  \textit{PL} Lesson 14-1, page 193
☑ Niño-Hosch, Sec. 8.2–8.3, pp. 185–187. Testing a class implementation and Building a test system
☐ Testing strategies  \textit{PL} Lesson 14-2, page 194
☐ Selecting tests cases and checking them  \textit{PL} Lesson 14-3, page 195
☐ Debugging  \textit{PL} Lesson 14-4, page 196

Unit 9. Relations

This material is not covered in \textit{ProgramLive}.

Unit 10. Putting a system together

☑ Top-down programming  \textit{PL} Lesson 2-5, page 72
This lesson page introduces the notion of stepwise refinement, or top-down programming, as an aid to writing a method (or set of methods).

☑ Object-oriented design  \textit{PL} Lesson 3-8, page 93
This lesson page discusses the design of an object-oriented program, using an example of a simple clock game.

Unit 11. Software quality

☑ Good programming paractices  \textit{PL} Lesson 13-1, page 185
\textit{ProgramLive} has no equivalent discussion of software quality. Lesson 13, “Programming style”, provides some of these thoughts.

Unit 12. Lists and iteration

Niño-Hosch first specifies a class \texttt{StudentList} and then discusses loops in terms of this class, using as examples only loops that process a list; \textit{ProgramLive} provides a more thorough study of loops with more varied examples. Niño-Hosch delays discussion of loop invariants until a later chapter but never uses them; \textit{ProgramLive} uses loop invariants right from the start and develops and presents all loops in terms of loop invariants.

Finally, all the algorithms that Niño-Hosch discusses in this chapter, like searching a list for a value, are discussed in \textit{ProgramLive’s} Lesson 8 on arrays.

Therefore, it is impossible to give a useful correspondence between sections of this chapter of Niño-Hosch and activities in \textit{ProgramLive}. In place of this, we give below a checklist for \textit{ProgramLive’s} lesson on iteration.
Do Lab PGL-1 (Executing a while loop) of this lesson.  

Knowing how a loop is executed is not enough; you have to know how to understand what happens within a loop, and you have to be able to explain a loop to others. This requires the notion of a loop invariant. Study this material carefully.

Several examples of loops

Do Lab PGL-2 (Developing loops from invariants) of this lesson.  

(Optional) Do Lab PGL-3 (Developing loops . . .II) of this lesson.

Conventions for indentation

Read the footnote near the bottom of the lesson page on conventions for indentation and for loop invariants.

Loop schemata

Do Lab PGL-4 (Using loop schemata) of this lesson.

The for loop

(Optional) Do Lab PGL-4 (Translating whiles . . .) of this lesson.

Making progress and stopping

Miscellaneous points about loops

The first two activities illustrate an important use of the statement-comment. You need not study the information about the do-while loop, the continue statement, and the break statement. In general, these constructs are not needed at this point of your programming career. If you see them in a program and want to find out about them, look in the ProgramLive glossary or index.

Unit 13. Sorting and searching

The corresponding lesson pages of ProgramLive treat arrays and not lists.


Finding the minimum value (activity 3)  
This algorithm is used within selection sort.

Inserting into a sorted segment (activity 4)  
This algorithm is used within insertion sort.

Selection sort and insertion sort  
(ProgramLive does not do bubble sort.)

Niño-Hosch, Sec. 13.3, pp. 314. Binary search
Lesson plan for Niño & Hosch’s Programming & Object-Oriented Design

Binary search (last activity)  
PL Lesson 8-5, page 156

Linear search (first two activities)  
PL Lesson 8-5, page 156

Niño-Hosch, Sec. 13.4, pp. 320. Verifying correctness: using a loop invariant

Loop invariants (activities 2-3)  
PL Lesson 7-1, page 127

Developing loop invariants (activity 1)  
PL Lesson 7-2, page 132

Making progress and stopping  
PL Lesson 7-5, page 138

Unit 14. Abstraction and inheritance

This chapter introduces the subclass and all the features that go with it. Object-oriented programming without subclasses would be possible, but not half as useful as with subclasses.


Subclasses  
PL Lesson 4-1, page 97

Do Lab PGL-1 (Drawing objects) of this lesson.  
PL Lesson 4

Object-oriented design with subclasses  
PL Lesson 4-4, page 103
Skip the details about implementing the methods.

Niño-Hosch, Sec. 14.2.2, pp. 336. Abstract classes and abstract methods

Abstract classes  
PL Lesson 4-5, page 104

Niño-Hosch, Sec. 14.2.3, pp. 338. Constructors and subclasses

Writing a constructor for a subclass (activity 1)  
PL Lesson 4-2, page 99

Do Lab PGL-2 (Writing constructors) of this lesson.  
PL Lesson 4


Overriding an inherited method (activity 2)  
PL Lesson 4-2, page 99

Overloading method names (activity 4)  
PL Lesson 3-1, page 75

Niño-Hosch, Sec. 14.4, pp. 344. Subclasses and contract
No specific activity in ProgramLive corresponds to this section.

Niño-Hosch, Sec. 14.5, pp. 346. Using inheritance
No specific activity in ProgramLive corresponds to this section.

Niño-Hosch, Sec. 14.6, pp. 350. Feature accessibility
No specific activity in ProgramLive corresponds to this section.
Lesson plan for Niño & Hosch’s *Programming & Object-Oriented Design*

- Calling an overridden method (activity 3)  
  *PL Lesson 4-2, page 99*
- Keywords *this* and *super* (activity 4)  
  *PL Lesson 4-2, page 99*
- Modifier *protected* (activity 6)  
  *PL Lesson 4-2, page 99*
- The class hierarchy (activity 7)  
  *PL Lesson 4-2, page 99*

  In *ProgramLive*, the scope rules for each kind of variable or method are treated at the place where the variable or method is defined. Look up “scope” in *ProgramLive*’s Glossary for a summary and in *ProgramLive*’s Index in order to get to the activities where more detail is given.

**Unit 15. Modeling with abstraction**

This chapter introduces the *interface*, which is a Java construct.

- Niño-Hosch, Sec. 15.1, pp. 371. Abstract classes and interfaces
  
  - Abstract classes  
    *PL Lesson 4-5, page 104*
  - Interfaces  
    *PL Lesson 12-1, page 179*
  - The interface as a type  
    *PL Lesson 12-2, page 180*
  - Interface *Comparable* (as an example)  
    *PL Lesson 12-3, page 182*

- Niño-Hosch, Sec. 15.2, pp. 380. Extension and composition
  This material is not discussed in *ProgramLive*.

- Niño-Hosch, Sec. 15.3, pp. 386. Extension and state
  This material is not discussed in *ProgramLive*.

**Unit 16. Organizing lists**

No part of *ProgramLive* corresponds to this chapter.

**Unit 17. Recursion**

What can be done with iteration can be done with recursion, and vice versa. Recursion may be a bit slower (in execution), but its use allows many programs to be expressed more simply. Some extremely beautiful *functional* rely solely on recursion—they don’t have iteration, or even an assignment statement.

- Niño-Hosch, Sec. 17.1, pp. 407. Recursion and iteration
  
  - Recursion  
    *PL Lesson 15-1, page 197*
Lesson plan for Niño & Hosch’s Programming & Object-Oriented Design

Execution of calls on recursive methods

ProgramLive goes into more detail on how recursive calls are executed. ProgramLive also shows how to eliminate tail recursion.

Do Lab PGL-1 (Writing recursive String methods) of this lesson.

Do Lab PGL-2 (Writing recursive integer methods) of this lesson.

Niño-Hosch, Sec. 17.2, pp. 416. Example: the towers puzzle

Some interesting recursive methods

Instead of towers of Hanoi, ProgramLive gives three other examples.

Niño-Hosch, Sec. 17.3, pp. 422. Quicksort

Partitioning an array segment (activity 5)

Quicksort

The basic algorithm may require \( n^2 \) space; ProgramLive shows how to reduce this to logarithmic space.

Niño-Hosch, Sec. 17.4, pp. 430. An inefficient algorithm

This example is not covered in ProgramLive.

Niño-Hosch, Sec. 17.5, pp. 431. Indirect recursion

This topic is not covered in ProgramLive.

Niño-Hosch, Sec. 17.6, pp. 433. Object recursion

This topic is not covered in ProgramLive.

Unit 18. Failures and exceptions

Niño-Hosch, Secs. 18.1–18.2, pp. 443-444. Failures and the Java exception mechanism

Output of thrown Exceptions and Errors

Niño-Hosch, Sec. 18.2.1, pp. 445. Exceptions as objects

The throwable object

Niño-Hosch, Sec. 18.2.2, pp. 446. Catching exceptions

Catching a thrown Exception (activities 1–3)

Niño-Hosch, Sec. 18.2.3, p. 449. Propagated exceptions

Propagation of a thrown object (activity 4)

The throw statement

Niño-Hosch, Sec. 18.2.4, pp. 449. Checked and unchecked exceptions
Checked exceptions and the throws clause  

Niño-Hosch, Sec. 18.3, pp. 450. Dealing with failure: using exceptions  

Hints on using exceptions  

Unit 19. Building the user interface  

Niño-Hosch, Sec. 19.1, pp. 464. The system interface  

Niño-Hosch, Sec. 19.2, pp. 468. An introduction to Swing  

ProgramLive does not concentrate on the Swing classes but also briefly discusses the awt classes.  

Components and containers  

Niño-Hosch, Sec. 19.3, pp. 475. Creating components  

Niño-Hosch, Sec. 19.4, pp. 482. Events: programming the user interface  

Listening to a GUI  

Niño-Hosch, Sec. 19.5, pp. 491. Some class features  

Appendix D of this Companion contains the specifications of many methods of many of the Swing and awt classes. Reference this appendix often when writing Java programs that deal with GUIs.  

Listening to a GUI  

Components and containers  

Windows and frames  

Unit 20. Designing the GUI front end  

This material is not covered in ProgramLive.  

Unit 21. Computational complexity  

This material is generally not covered in a first course. There are a few instances in ProgramLive where analysis of execution time is discussed — e.g. with binary search, linear search, and quicksort— but ProgramLive does not contain a full discussion of analysis of execution time.
Unit 22. Implementing lists: array implementations

- Niño-Hosch, Sec. 22.1, p. 557. Arrays
  Niño-Hosch covers arrays mainly to show how to implement their Lists of Sec. 12.1. For example, they do not discuss array initializers and uses of arrays for other than Lists. ProgramLive gives a more thorough, traditional coverage of arrays.

- Introduction to arrays  
  PL Lesson 8-1, page 143

- Talking about array segments  
  PL Lesson 8-2, page 146

- Do Lab PGL-1 Using arrays) of this lesson.  
  PL Lesson 8

- Some programs that use arrays  
  PL Lesson 8-3, page 148

- Arrays and classes (activities 1–3)  
  PL Lesson 8-4, page 153

- Some basic array algorithms  
  PL Lesson 8-5, page 156

- Niño-Hosch, Sec. 22.2, p. 562. An array-based list implementation
  Class StudentReport (activities 2)  
  PL Lesson 8-4, page 153
  Activity 2 implements the list of students in an array.

- Niño-Hosch, Sec. 22.3, p. 573. Dynamic arrays
  A class that implements dynamic arrays (activities 5–6)  
  PL Lesson 8-4, page 153

- Class Vector  
  PL Lesson 5-4, page 110

Unit 23. Implementing lists: linked implementations

This topic is not covered in ProgramLive.

Unit 24. Organizing list implementations

- Niño-Hosch, Sec. 24.1, p. 603. A library structure
  This example is specific to Niño-Hosch.

- Niño-Hosch, Sec. 24.2, p. 608. Iterators
  Interfaces Enumeration and Iterator  
  PL Lesson 12-4, page 183

- Niño-Hosch, Sec. 24.3–24.4, p. 617–625. Iterators as arguments and Comparing implementations
  This material is specific to Niño-Hosch.

- Niño-Hosch, Sec. 24.5, p. 625. The java.util Collection hierarchy
  This material is not covered in in ProgramLive.
Unit 25. Dispensers and dictionaries

This material is not covered in ProgramLive.

Unit 26. Appendix A: Stream I/O

- Niño-Hosch, Sec. a.1, p. 654. OOO library classes
  - Input/output (activities 2 and 3)  
  - Activity 2 introduces a class JLiveRead for reading values from the keyboard, which provides the functionality of Niño-Hosch’s class BasicFileReader. This is basically what is needed in the beginning of an introductory course. Activity 3 discusses a simple GUI, JLiveWindow, for small amounts of I/O.

- Niño-Hosch, Sec. a.2, p. 657. The java.io library
  - Reading from the keyboard and files  
  - Writing to the Java console and files

Unit 27. Appendix B: Applets

- Applets  
- Html and applet commands  
- Examples of applets