

Table of Contents

Preface	vi
Laboratory 1—I/O and Exceptions	1
The focus of this laboratory is processing data from files and from the console. Both require introduction of exception handling, exceptions, and the try-catch mechanism. Most of the attention is centered on file processing in Java.	
Laboratory 2—Bits, Bytes, and Pieces.	19
Binary representation of data and Java’s bitwise operations are introduced. Hexadecimal constants are applied to the examination of bytes within an integer. Classes that allow bit-by-bit access to files are developed.	
Laboratory 3—Java GUI Programming	28
The AWT and basic graphics classes are introduced and used in simple applications. This lab not only represents a set of objects that illustrate inheritance, but it also provides the opportunity to bring event-driven programming front and center. Students develop a simple graphics program that visually records positions of mouse clicks. This provides the foundation for the next laboratory.	
Laboratory 4—Solitaire Games	43
Building on ideas of event-driven programming, this laboratory works through the design and creation of classes for a FreeCell solitaire game. This requires two sessions in the laboratory to complete, but results in a fully functional version of the game. It illustrates that careful design of the set of classes and their characteristics can make a large program manageable.	
Laboratory 5—Klondike	56
With the addition of one class and the modification of another, we illustrate reuse of a large collection of classes by writing the Klondike solitaire game. The post-laboratory converts this game into an applet, showing how easy it is to convert an application to an applet.	

Laboratory 6—Two-Dimensional Arrays	68
The most commonly used multidimensional array is the two-dimensional version. After examining common operations on such arrays and using them with files, the post-laboratory simulates Conway's Game of Life.	
Laboratory 7—Linked Lists	75
The list abstract data type is introduced, along with dynamic memory allocation. A simple linked list class for integers is implemented. A later laboratory develops a general-purpose linked list class.	
Laboratory 8—Iteration	87
The ability to access elements of a collection is a common need. Using a class that is a set of integers, an iterator is developed that illustrates the fundamental operations. The IntSet class that is used illustrates the basic concept and operations of the set abstract data type.	
Laboratory 9—Hashing Basics	100
A more efficient technique for information access uses a hash table. Chained hashing is presented and the development and testing of hash functions is the central topic of the laboratory. The post-laboratory develops a simple spelling-checker program whose dictionary operations use a hash table.	
Laboratory 10—Stack Implementations	119
The stack abstract data type is introduced, together with common applications. Both the array and linked list implementations are included.	
Laboratory 11—Queue Implementations	128
The queue abstract data type is introduced, together with common applications. Both the array and linked list implementations are included. The post-laboratory develops a simulation of multiple waiting lines and servers to test whether a single line with multiple servers is better than multiple servers, each with its own line.	
Laboratory 12—Binary Trees I.	136
The basics of binary trees are introduced. The laboratory uses a tree to implement a simple guessing game.	
Laboratory 13—Binary Trees II	144
Tree traversals and binary search trees are central topics in this laboratory. The post-laboratory examines complete trees and balanced trees to find trees of minimal height.	
Laboratory 14—Heap Applications.	157
A heap is a complete tree with a special property at each node. An efficient array implementation is possible, which allows the laboratory to implement the heap sort and compare it with the quick sort. The post-laboratory introduces priority queues that can be implemented using a heap.	
Laboratory 15—File Compression	168
Huffman encoding is used to implement simple file compression. The laboratory requires two sessions to develop the priority queue used by the algorithm and to complete the method that performs the compression. Two classes from Laboratory 2 are used to read and write bits. The post-laboratory implements the expansion method.	

Appendix A—The LabPkg Package	187
Appendix B—IDE Information	194
Appendix C—Debugging Tools	198
Index	200