Suggested Topics for an IS Introductory Course in Java

Azad I. Ali
Butler County Community College, Butler, PA, USA
Azad.Ali@bc3.edu

Frederick Kohun
Robert Morris University, Coraopolis, PA USA
Kohun@rmu.edu

Abstract

The purpose of this paper is to suggest topics that may be covered in an introductory programming using Java programming language at a typical Information Systems (IS) program. The paper begins by explaining in general terms three different methods that are used for teaching introductory courses in programming. It then clarifies some common characteristics that distinguish IS programming courses. Last, the paper recommends topics that may be covered for an introductory programming course using Java as the selected programming language in an IS program.

Keywords: Java Programming, IS Programming Course, Intro Programming, Programming Topics, IS2002 Programming Course

Introduction

As part of the curricular revision process, colleges and universities utilize a variety of criteria to help decide which programming language will be used in the first level (introductory) programming course. The selection of a programming language first begins by choosing an approach (e.g. object oriented versus procedural). After the selection of the programming approach comes the decision to select a specific programming language along with the topics to be covered. Furthermore, the type of program (e.g. information systems or computer science) influences the selection of the approach, the programming language or the specific topics to be covered in such a course. The perceived notion is that the Java programming language is similar to C or C++, which is usually taught in computer science departments. As a result, information systems programs frequently include both of these programming languages within their curriculum.

This paper suggests topics that may be covered in an introductory programming course in Java within an information systems (IS) program. To help understand the topics that are usually covered in IS programs; this paper is going to use the IS2002 standard curriculum. The IS2002 standard curriculum suggests courses to be included in a typical IS program along with topics to be covered in each of the suggested courses.

This paper begins by introducing three approaches to teaching programming courses: Procedural, Object-Oriented and Visual. It then briefly discusses the advantages and disadvantages of each approach as it pertains to teaching a first level programming course. Following that, it focuses on one particular approach that is applicable to IS programs by suggesting topics that are appropriate

for students in an introductory programming course. Finally, the paper tabulates and sequences the suggested topics for use in a first level programming course in Java.

Java as a First Programming Course

The goal of a first programming course is to provide the students with a foundation of specific skills such as problem solving, algorithm design and good programming practices (Schaller, 1997). This is compounded with the need to include competency—if not mastery—in the syntax of the programming language used. But with respect to the extent any particular programming language can fulfill these goals, comes the debate as to which programming language is best for a first programming course.

Before the proliferation of contemporary programming languages, the selection was simple. The BASIC programming language was the selection for a first programming course. There was simple reason cited for selecting BASIC for introductory course: BASIC is basic. The simplicity of writing the code and its applicability to a wide spectrum of problems and applications made it a good choice. Later, BASIC was substituted with PASCAL for the similar reasons: relative simplicity, structure and ease of use.

As more languages with distinctive features became available, the selection process for a first programming course language became more difficult. Language characteristics have emerged that require more than just simplicity of use. Such requirements may include range of application use, context for good programming practices and programming concepts. Further requirements emerged which were more specific to the field in which the programming course is taught (Prasad & Li, 2004).

Advocates of using Java in a first programming course cite certain characteristics of this programming language as a reason for its selection. Some of these characteristics include simplicity, modularity, modifiability, extensibility, flexibility, and other factors (Wang, 2003). Another factor that was commonly cited is that Java is "Platform Independent" (Gittleman, 2000). This essentially means that the same code can be executed across different set of hardware and software platforms.

Despite all these good factors in favor of the language, Java is not without criticism. The excessive use of semicolons and curly brackets are just two factors that confuse beginner programming students. Then Input/output is not handled well with Java because keyboard input is not easily supported in Java (Schaller, 1997). Add to that, the many abbreviations that are widely used in Java as well as the repeated use of the period to separate the classes from the objects, and methods. Additionally, the Java environment is not standard. The author of this paper, for example, has within the last five years, switched from using Visual J++ to Borland J and then to J Sharp. This switching was not by choice, but was driven instead by technological changes and market updates.

However, given all the factors mentioned above, Java can still be a good first programming language. The editor within Java can offset many of the difficulties mentioned. Above; it provides various underlying, coloring and matching of indentation and bracketing that aids in program design and debugging. Furthermore, the wide range of applications for which Java can be used, especially with the applets and the World Wide Web, make it a good candidate for a first programming course (Malik & Nair, 2003).

Programming Approaches

A first programming course teaches the students the fundamentals of programming languages and usually covers certain specific topics such as programming statements, iteration statements or

loops, selection statements, arithmetic operations and others. Each textbook introduces the above mentioned topics from the context of one of three approaches. While multiple approaches may be discussed at any given time in the text, the topics are presented from frame of one dominant approach (Bradley, 2002; Kamin, 2002; Liang, 2003; Wang, 2003).

The three distinct approaches that frame a programming language are: Procedural, Visual and Object Oriented. The following three sections explain each of the approaches mentioned earlier. A brief description of the approach will be introduced then the advantages and disadvantages of using them will be further discussed.

The Procedural Approach

This approach divides a program into procedures or modules. "Modularization" or "Top-down design" are alternative names that are commonly mentioned for this approach. Among the programming languages that are more adoptive to the procedural approach is COBOL although other programming languages can follow this approach also.

Using the procedural approach, different procedures or modules constitute the building blocks of the program (Weisert, 1997). Functions are created in this approach to complete small tasks and they communicate with other functions through arguments, parameters and return values ("Functional Programming", 2004). Teaching this approach in a beginner-programming course usually begins by displaying simple output on the console using commands in Java. It continues building on the display of output for a number of chapters (Liang, 2004). Advanced topics in object-oriented and visual terminologies follow.

The advantage of using this approach lies in its simplicity: It is easier to use, easier to debug and to modify (Deitel & Deitel, 2003). There are not many terms that need to be introduced at the beginning, thus the teaching can jump right into the syntax and logic and may be able to cover more of the subjects related to the language. But this approach has its disadvantages. The current trend is a shift to newer methods of explaining programming including the object-oriented methodology. Besides, this approach is not strong in supporting the concept of "usability" that is mentioned as one of the main advantages of using Java (Wang, 2003).

Object-Oriented Approach

With this approach, the design of the program is centered on objects, attributes for the objects, and relationships among objects (Gittleman, 2000). Textbooks that follow this approach and which are used in programming course in Java begin by explaining OOP concepts such as classes, objects, and inheritance. All subsequent programs and examples are built on these topics. So, students get to use this approach from the beginning and to the end (Kamin, 2002, Wang, 2003).

The advantage of using this approach is that it gets the students to learn about consistent methods of program development that is becoming more widely used in the industry (Chen, 2004). Other advantages of using this approach include modifiability (easy to modify), extensibility (easy to add features to) and reusability (objects can be used in other programs) (Wang, 2003). The main disadvantage of using this approach for first level programming courses stems from its relative difficulty: It gets to explain a lot of terms that may not be easily explained through using simple examples. This will delay the start of getting into the syntax and as a result may not be able to cover the same materials that are normally need to be covered in introductory programming course (Schaller, 1997).

The Visual Approach

The visual approach explains the programming concepts through the display of visual objects such as text boxes, radio buttons and command buttons. In other words, it uses Graphical User Interface (GUI) objects from the beginning (Bradley, Case, & Anita, 2002). Any programming example that shows input data is explained using GUI objects. At the same time, output from programs is displayed using the same GUI objects (e.g. panels).

Although this may appear to be preferred approach because it teaches objects that are commonly used in today's Window's environment, there is a down side. Using this approach for a beginning programming course may lead to spending more time explaining how to build a user interface rather than spending it to learn programming concepts. This may delay getting into the explanation of programming syntax. Besides, drawing user interface is not easily handled in Java (Schaller, 1997).

Specific Characteristics of IS Programming Courses

In order to decide on which approach is more suitable for IS majors some understanding of the background of the major may be helpful. A study noted that Computer Science majors are more rooted in math and engineering while the Information System majors require more analytical and organization ability (Kohun & Wood, 2004). Given that IS programs are less rooted in math, the use of the procedural approach for IS programming courses seems appropriate. This is substantiated when a survey conducted to investigate the difference between IS and CS programming students found out that IS students have more difficulties than computer science majors with program design (Prasad, 2004). This is especially true when additional terms and concepts are introduced to the design process. In such courses, the programming codes are explained in more detail, spaced out, and more examples related to business environment are given. This is in contrast to other computer majors where codes are combined and emphasis is placed on examples focused on performing complex mathematical operations. Again, this reinforces the use of the procedural approach for IS programs (Schaller, 1997.

IS2002 Guidelines

The Association of Computing Machinery (ACM) with the Association for Information Systems (AIS) and the Association for Information Technology Professionals (AITP) developed a model curriculum for undergraduate degree programs in Information Systems (Association for Computing Machinery [ACM], 2002). The main objective of this model curriculum is to help IS faculty and administrators design programs to produce graduating students that are prepared for the job market in the Information Systems field.

IS2002 listed a number of courses for IS students to take. The course selection ranges from theory to networking to programming and others. These courses are numbered with a prefix of IS2002 added it to it to distinguish that it is a course suggested by IS2002. The first programming course suggested in this curriculum is IS2002.5 "Programming, Data, File and Object Structures". The IS2002.5 lists different topics to teach. The remainder of this section explains topics listed in the IS2002.5 course as they pertain to Java programming language.

Data Structures and Representation, Character, Records And Files

Data structure and representation explain the various data types that range from simple ones such as Byte, Char, and Integer data types to the more complicated such as short, long, float and double. The basic data types can be explained early in the course so to give examples that are mean-

ingful. Thus explaining variables that utilize simple data types may be essential at the beginning. More complicated data types such as Short, Long or Float can be explained later and also within the subject or Precision of Data when discussing the different methods of converting data from one data type to another one.

Precision of Data

Precision of data deals with data presentation, truncation and converting from one data type to another. These topics require certain knowledge of basic data representation. It may also require the knowledge of other data types such as long, and short, their storage of data, their range of values and their methods of display. Knowledge of these data types is essential before the students can learn about precision of data. Various examples can be given regarding truncation and moving data from one data type to another. Another topic that will be helpful to understand the subject of data precision is the conversion from a longer data format to shorter one and vice versa (Narrow conversion versus Wide conversion).

Information Representation, Organization, and Storage

The distinction between information representation and data representation and precision of data is not quite obvious. The information representation and organization could mean presenting data in field, record and table formats or it could mean different file organization methods. But in either case, these topics are not handled easily in Java and if they are to be explained in an introductory programming course, they can be postponed to the end of the course—only after the students have mastered the basic concepts of program logic and control structures.

Algorithm Development

Algorithm development can be covered in general terms. The same algorithm can be developed for various programming languages, though different language use specific terms that may be used for each language. Having a general knowledge of algorithm development at the beginning of the course helps the students understand the basics of program development. However, when developing an algorithm in Java, the emphasis must be placed on the terminology specific to Java such as classes, objects, methods, instantiation and related terminology. Teaching algorithmic logic concurrently with specific concepts of Java, although necessary, but it may be overwhelming to the students taking a first programming course. The suggested strategy here is to teach algorithm development in general terms at the beginning of the semester and then get into the specific terms of Java later on.

Programming Control Structures

The three control structures of programming (Sequence, Selection and Iteration) are common subjects in all entry level programming courses. The depth of coverage may depend on the language itself. For each language comes the specifics regarding syntax of the programming language, the method by which they execute the code, and other details. The following explains the control structures in Java programming language.

Sequence control structure

This control structure usually discusses the statements that are executed one after another. It is the most basic control structure and is usually covered early at programming courses. But the simple control structures cannot be explained within the programming context unless there are statements that allow user input; perform processing, calculations and manipulations; and then displays and formats the output. Within the context of sequence control structures certain topics can be included such as the following:

1- Displaying prompts on the screen using Java statements.

2- Prompting users to enter data such as characters, numbers.

3- Arithmetic operations

Selection control structure

The selection control structure can be implemented in two ways in Java: either using the *if…else* to select between two options or using the *switch* statement to select between more than two options. Added to this general description of selection statements in Java are the additional topics of the branching out and the nested selection statement. Because these statements involve more knowledge of branching and executing, the suggestion here is to cover it after covering the sequence control structure.

The iteration control structure

In Java there are three methods of implementing iterations (or loops): *For, While and Do-while* loops. Each of the three methods has its appropriate place and usage. The suggestion is that all three methods of implementing looping in Java be explained in one chapter (module) and be done after completing the chapter (module) that explains control structure, data formats, arithmetic operations and selection structure.

Program Correctness, Verification and Validation

Ensuring program correctness can range from the simple task of checking program syntax to other topics that involves using the debugger. Syntax checking is an early on subject in learning programming languages and it needs to be covered at the beginning of the course. Java editors can be helpful in recognizing syntax errors; they provide features that underline the exact location of syntax errors and they provide helpful messages that identify syntax errors when submitting programs for compilation or building.

Program debugging can be a useful tool to ensure program correctness. But in order to better utilize the various features in the debugger, knowledge of data presentation and various control structures would be deemed essential. Thus the suggestion here is to teach the debugging tools later in the course.

An important program verification procedure used within Java programming language is the use of *Try … Catch* statements. This newer method deals with error handling more systematically. This feature handles avoiding runtime errors and can be incorporated within the same topic discussing the debugger and runtime errors.

File Structures and Representation

File handling is more complicated in Java than in other languages. The simplest file format (which is sequential file organization) is handled in Java by reading a line (or a stream) of data, then breaking it into components using the Tokenizer statement. It may also include error handling using the *Try…Catch* statements. These subjects require substantial knowledge of the programming language before it can be understood. Thus if these subjects are introduced in an entry level Java course, it ought to be covered at the end.

Programming in Traditional Environments that Incorporate Event-Driven, OO Design

Understanding Object-Oriented terms, topics and terminology is essential to fully understanding the Java programming language because Java is built around methods, objects, instantiations and others. But the recommendation here is to first teach the features that are basic to all programming languages such as program syntax, arithmetic operations and control structures before indulging in the specifics of the Object Oriented terms. Thus the suggestion here is to teach a chapter (module) on these topics later in the course.

Regarding event driven design, the recommended approach here is to use the Panels topic in Java or to teach the display of various visual objects on the screen and then let the program respond to events such as the clicking a button or the checking a radio button. But this requires the use of applets in Java. An appropriate place to teach this is at the end of the course. This can include examples that are used to create and display various elements on a form, responses to user actions on the form, and then further program processing.

Recommendations

Based on the description introduced in this study, the characteristics of the Java programming language, and the guidelines developed for the programming course in IS2002, this paper recommended topics to teach in an introductory programming course in Java. The recommendations here divided the topics into chapters (modules) and then additional topics within each chapter (module). Table 1 below lists three columns: the first column lists the suggested chapter (module) number. The second column lists a suggested chapter (module) title that will be indicative of the main theme of the chapter (module). And the third column lists individual topics suggested for the chapter (module).

Table 1 – Suggested Topics to be Covered in Beginner Programming Course in Java

Chapter # (Module #)	Title	Topics
1	Intro to Programming	Information Representation and Storage Algorithm Development
2	Java Programming Concepts	Syntax, Logic Simple Statements
3	Arithmetic operations	Numeric Operations Arithmetic Operations
4	Selection Statements	Simple Selection, If …Else Nested if Statement, Switch statement
5	Iteration – Looping	For Statement While Statement Do-While Statement
6	Data Precision in Java	Data Conversion
7	Arrays	Declaring Arrays Copying and moving data
8	Object-Oriented Programming	Methods and Classes Objects and Constructors
9	Event Driven Programming	GUI and Java Layout Manager Event, listeners and handling events
10	Java Applets	Applet Class, Creating Java Applet Combining with HTML Files
11	File Processing	I/O Stream, File Streams Text input and output

References

Association for Computing Machinery [ACM]. (2002). *IS2002 model curriculum for undergraduate degree in information systems.* Retrieved December 10, 2003 from http://www.acm.org/dl

Bradley J., Case M., & Anita C. (2002). *Programming with Java.* Boston, MA: McGraw-Hill.

Chen, K. C. (2004). Comparison of object-oriented and procedural computer languages: Case study of C++ programming languages. *Issues in Information Systems, 4*, (1).

Deitel, P. & Deitel, H. (2003). *C++ How to program.* Upper Saddle River, NJ: Prentice Hall.

Functional Programming. (2004). Retrieved November 23, 2004 from http://c2.com/cgi/wiki?functionalprogramming

Gittleman, A. (2000). *Objects to components with the Java platform*. El Granada, CA: Scott Jones.

Kamin S., Mickunas D., & Reingold E. (2002). *An introduction to computer science using Java*. Boston: McGraw Hill.

Kohun, F. G., & Wood, D. F. (2004). The ABET CAC accreditation: Is accreditation right for information systems? *Issues in Information Systems, 4* (2).

Liang, D. Y. (2003). *Introduction to Java programming*. Upper Saddle River, NJ: Prentice Hall.

Malik, D. S. & Nair, P. S. (2003). *Java programming: From problem analysis to program design.* Boston: Thompson Course Technology.

Prasad C. & Li, X. (2004). Teaching introductory programming to information systems and computing majors: Is there a difference? Retrieved November 2, 2004 from ACM Digital Library http://www.acm.org/dl

Schaller, N. C. (1997). Using Java in computer science education – Panel discussion. Rochester Institute of Technology. Retrieved February 28, 2005 from http://www.cs.rit.edu/~ncs/uppsala97/

Wallingford, E. (1996). Toward a first programming course based on object-oriented patterns. *Technical Symposium on Computer Science Education*, 27-31.

Wange, P. S. (2003). *Java with object-oriented programming.* Pacific Grove, CA: Thomson Brooks/Cole.

Weisert, C. (1997). Learning to program: It starts with procedural. *Information Disciplines*. Retrieved November 11, 2004 from http://www.idinews.com/procedural.pdf

Biographies

Azad Ali, D.Sc., Associate Professor of Computer Information Systems at Butler County Community College in Butler, Pennsylvania has 22 years of combined experience in areas of financial and information systems. He holds a bachelor degree in Business Administration form the University of Baghdad, an M.B. A. from Indiana University of Pennsylvania, an M.P.A. from the University of Pittsburgh, and a Doctorate of Science in Communications and Information Systems form Robert Morris University. Dr. Ali research interests include object oriented languages, web programming tools, and curriculum design. His community service and academic expertise gets him in the news on Pittsburgh television and in the newspapers.

Frederick G. Kohun, Ph.D., Associate Dean and Professor in the School of Communications and Information Systems at Robert Morris University in Pittsburgh, has more that 29 years experience as a professor and administrator in the information systems field. He holds a bachelor degree in economics from Georgetown University, graduate degrees in economics and information science, from the University of Pittsburgh, and a Ph.D. in applied history in technology from Carnegie Mellon University. He had a leadership role in the design and implementation of eight technology based academic programs at the undergraduate and graduate level including a doctoral program. Most recently, he was involved in the first round of ABET-CAC information systems accreditation.