IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4683

The Concepts of Class and Object as Presented in Selected Java Textbooks

Robert Joseph Skovira, PhD
Robert Morris University, PA
rjskovira@worldnet.att.net

INTRODUCTION

In object-oriented programming languages, and in particular Java, the most elemental concepts are those of class and object. Yet, in many textbooks that are the basis of teaching and learning Java, ideas are not clearly presented. Textbooks that are used to introduce students to object-oriented programming in Java, ought to be clear about the Java structures of class and object and how they are constructed and used. There is a problem with most textbooks purporting to be the learning platform for object-oriented programming in Java. The problem is the breadth and depth of the discussion and presentation of the class and object structures. These structures are concepts introduced and discussed, but the discussion is spare and sparse. There is a reliance on using the syntactical structures to enhance and extend the explanations of what classes and objects are, but the explicit coupling of syntax to concept is weak. The cognitive model of class and object is usually fragmented and not clearly drawn. Object-oriented textbooks in Java do not sufficiently link, in a descriptive or explanatory way, the conception with the syntax. A great burden of understanding of how things go together and work in regards to class and object rests upon the individual reader. Consequently, this essay is a study and analysis of Java textbook presentations of class and object concepts and how these ideas are modeled and implemented. In other words, the paper is a study of the cognitive models of class and object. The paper discusses various ways in which the class and object concepts are represented to students of the Java language.

OBJECT-ORIENTED PROGRAMMING

All of the textbooks say something about object-oriented (O-O) programming, or the object-oriented paradigm. They, in some fashion, try to set the context, but not very well. In most cases, the object-oriented paradigm is presented as a natural way of dealing with what most of the authors call "the real world." Programming in the O-O paradigm is modeling entities of real world environments.

One author writes that O-O programs are "models" of "real world system" The program consists of "objects" representing "entities," customers, vendors, reports, transactions, in the world (Hughes, 2002, 31). Another author states that object-oriented languages "model objects in the real world" and that classes are "representations" of things in the world (Cornelius, 2001, xiii-xiv). Another text states that another O-O feature is its "natural" way of perceiving and thinking about things. Problem solving is identifying "objects" in problematic situations and any necessary "actions" (Garside & Mariani, 2003, 29).

But, even so the O-O paradigm is not clearly conceptualized in the textbooks. For example, one author states that object orientation is about objects and sending and receiving "messages" (Morelli, 2000, 58). The same text describes object oriented programming in terms of the principles or rules such as divide and conquer, encapsulation, interface, information hiding, generality, and extensibility (Morelli, 2000, 8-9). Thus, a text states that "object definitions" and "instances of actual objects" are features of the O-O paradigm (Garside & Mariani, 2003, 35).

For several other textbooks, the O-O paradigm refers to how "data" and "procedures" are packaged in "objects" or "encapsulated," effectively "hiding" them (Shelly, Cashman & Starks, 2001, JI.10; Deitel & Deitel, 2002, 380). Finally, one text, usually used as a support text in trying to understand object-oriented programming, states that O-O paradigm is about joining functions and data to simplify a program (Holzner, 1998, 15).

GENERAL NOTION OF OBJECT

Some of the texts begin the discussion of what an "object" is by alluding to examples of experienced objects in the everyday world. Am object is thought of as a conceptual structure or a module of code. Objects are representational of things we deal with (Savitch, 2001, 211). But, one author sees an object in relation to activity and tasks. Classifying things or objects in the world is a natural way of thinking about the world and its things. Objects are grouped based on some differentiating attribute (Morelli, 2000, 60). Objects are inanimate things about which we have difficulty thinking that they send and receive message. However, objects are animate things in the world which do naturally communicate and interact with one another by sending and receiving messages (Morelli, 2000, 7). Another text, for example, states that, "Object is a broad term that stands for many things. For example, a student, a desk, a circle, and even a mortgage loan can all be viewed as objects. Certain properties define an object, and certain behaviors define what it does (Liang, 2000, 142)."

For at least one text, the idea of object is a useful device for modeling complexity in systems. "We can use the concept of objects to model quite complicated real-world systems that consists of many different kinds of objects and many instances thereof" (Garside & Mariani, 2003, 30).

SPECIAL NOTION OF OBJECT

From an instructor's, and student's, points of view, the penultimate idea to grapple with in O-O programming is the special notion of object. This idea is dealt with in all of the texts, as would be expected. But, as perhaps, not expected, the idea is not as well presented as it ought to be, in my estimation. One text states that an object is "module" "encapsulating" a program's behavior (Morelli, 2000, 7). Another text states that an "object" has "data" and "actions" (Savitch, 2001, 210). An object is a complex entity (Savitch, 2001, 213) that has callable operations (Garside & Mariani, 2003, 30). Further attempt at clarification is when a text states that an object has a "state," i.e., its data and procedures (Garside & Mariani, 2003, 29). The ultimate clarifying note is that an object is a "noun" (Shelly, Cashman & Starks, 2001, JI.12).

One of the most repeated explanatory sentences in all the textbooks is the one that simply states that an object is an instance of a class (Liang, 2000, 144; Morelli, 2000, 64; Liang , 2000, 143; Cornelius , 2001, 43, 50; Bishop, 1998, 23).

The most intriguing explanation is to be found in two texts. This is that an object is a "black box" (Garside & Mariani, 2003, 30; Shelly, Cashman & Starks, 2001, JI.12).

Another explanatory attempt views an object as a modeling piece, perhaps like a Lego block. The interesting thing here is that the explanation leads to the notion of a class as a category of objects sharing behavior. A class defines shared behavior. So, instead of dealing with the idea of what an object is, we move to what defines it. The text states that objects are "model elements" (Anow & Weirs, 2000, 4).

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

CLASSES AS CONSTRUCTS AND DEFINITIONS

One text states that classes are "constructs" defining objects by specifying variables and methods (Liang, 2000, 143; Holzner, 1998, 15; Bishop, 1998, 80; Morelli, 2000, 63, 65).

Other texts discuss class as a definition encapsulating an object's information and behaviors (Morelli, 2000, 61, 78 ; Schildt, 2001, 130; Garside & Mariani, 2003, 40; Savitch, 2001, 211-212).

One text states that a class is a "category" of objects, a way of classifying common properties and behaviors (Shelly, Cashman & Starks, 2001, JI,12).

One text suggests that a class is set of elements or declarations about information and processes (Anow & Weirs, 2000, 4; Hughes, 2002, 250).

Other texts discuss the notion of class as a way of defining "types" of things, the things beings objects as instances of the class (Cornelius, 2001, 44, xix, 23; Schildt, 2001, 130). This idea of a class-defining-new-type of entity is an way of extending objects and the scope of Java (Adams, Nyhoff & Nyhoff, 2001, 70).

One text states that a class is an "abstract entity" or an "abstraction" (Morelli, 2000, 77; Garside & Mariani, 2003, 43).

EXPLANATORY METAPHORS

An interesting aspect of the various texts studied here is that they all try to describe the function, and perhaps, nature, of a class by certain metaphors. These metaphors are blueprint, template, model, pattern, recipe, and cookie cutter. While we all commonly understand what these metaphors say, they still do not bring the instructor or the student any closer to the notion of class.

One metaphor for describing what a class is blueprint (Liang, 2000, 143). Another author uses the same metaphor of blueprint and creates a synonym in the form of the template metaphor (Morelli, 2000, 63; Garside & Mariani, 2003, 43). Another author uses the template metaphor (Schildt, 2001, 130). Still, another text manages to use template, blueprint, and extend the metaphoric range to include pattern, and model. (Adams, Nyhoff & Nyhoff, 2001, 70). Some flip places; the main metaphor is template, followed by blueprint (Morelli, 2000, 61). Another metaphor is the recipe (Liang, 2000, 143). A text uses the model metaphor to describe a class; here a class is a representation something. (Anow & Weirs, 2000, 2, 57). Another author uses a cookie cutter to cookie metaphor (Holzner, 1998, 14-15).

CONCLUSION

The burden of explaining the natures, and not merely the syntax, of class and object, and their relationship is the instructor's in almost all cases to bring things together coherently. The syntax for creating a class which defines objects is straight forward. But, it takes awhile to realize that not all classes produce objects. Thus, the sense of the relationship between class and object shows up in the discussion and understanding of how instance variables and class variables function within a program. That is, what an instance variable is and what a class variable is refers to what they do or can do in a program. The same holds for the discussion of instance methods and class methods. Understanding how these two kinds of methods work, and are allowed to work, shows up in understanding the relationship of class and object. It also shows up in the understanding that not all classes produce instances, are used to produce instances, or objects. This explanatory burden extends to the discussion of abstract classes and to interfaces as they appear in the object-oriented world of Java. And, this leads us to a consideration of the analysis and design of systems in an object-oriented manner. While we may experience, at the level of detail, things and stuff we can turn into objects, we must ultimately think, or program, the experienced objects as classes. We must do a classification turn, and create the conceptual versions of the actual things, the actual objects. This classifying turn is not discussed at all in any of the textbooks reviewed in this essay.

REFERENCES

Adams, Joel, Nyhoff, Larry R. and Nyhoff, Jeffrey. (2001). *Java: An introduction to computing*. Upper Saddle River, NJ: Prentice Hall.

Anow, David M. and Weirs, Gerald. (2000). *Introduction to programming using Java: An object-oriented approach.* Reading, MA: Addison Wesley Longman.

Bishop, Judy M. (1998). *Java gently*. Harlow, England: Addison Wesley Longman.

Cornelius, Barry. (2001). *Understanding Java*. Harlow, England: Pearson Education.

Deitel, H.M. and Deitel, P.J. (2002). *Java: How to program*, 4e. Upper Saddle River, NJ: Prentice Hall.

Garside, Roger and Mariani, John. (2003). *Java: First contact*, 2e. Pacific Grove, CA: Brooks/Cole Thomson Learning.

Holzner, Steven. (1998). *Java 1.2: In record time.* San Francisco: Sybex.

Hughes, David. (2002). *Fundamentals of computer science using Java*. Boston: Jones and Bartlett.

Liang, Y. Daniel. (2000). *Introduction to Java programming with Microsoft Visual J++ 6*. Upper Saddle River, NJ: Prentice Hall.

Morelli, Ralph. (2000). *Java, Java, Java: Object-oriented problem solving approach*. Upper Saddle River, NJ: Prentice Hall.

Savitch, Walter. (2001). *Java: An introduction to computer science and programming*, 2e. Upper Saddle River, NJ: Prentice Hall.

Schildt, Hebert. (2001). *Java 2: The complete reference,* 4e. Berkeley: Osborne/McGraw-Hill.

Shelly, Gary B., Cashman, Thomas J. and Starks, Joy L. (2001). *Java programming: Complete concepts and techniques*. Boston: Course Technology.

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/proceeding-paper/concepts-class-object-presented-selected/32250

Related Content

WSN Management Self-Silence Design and Data Analysis for Neural Network Based Infrastructure
Nilayam Kumar Kamilaand Sunil Dhal (2017). *International Journal of Rough Sets and Data Analysis (pp. 82-100).*
www.irma-international.org/article/wsn-management-self-silence-design-and-data-analysis-for-neural-network-based-infrastructure/186860

Digital Technologies for Teaching and Learning at the BoP: A Managerial Perspective
Alessia Pisoni, Alessandra Cortiand Rafaela Gjergji (2021). *Handbook of Research on Analyzing IT Opportunities for Inclusive Digital Learning (pp. 272-292).*
www.irma-international.org/chapter/digital-technologies-for-teaching-and-learning-at-the-bop/278964

Better Use Case Diagrams by Using Work System Snapshots
Narasimha Bollojuand Steven Alter (2016). *International Journal of Information Technologies and Systems Approach (pp. 1-22).*
www.irma-international.org/article/better-use-case-diagrams-by-using-work-system-snapshots/152882

The Role of Systems Engineering in the Development of Information Systems
Miroljub Kljajicand John V. Farr (2008). *International Journal of Information Technologies and Systems Approach (pp. 49-61).*
www.irma-international.org/article/role-systems-engineering-development-information/2533

The Effect of Innovative Communication Technologies in Higher Education
Stavros Kiriakidis, Efstathios Kefallonitisand Androniki Kavoura (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 3827-3838).*
www.irma-international.org/chapter/the-effect-of-innovative-communication-technologies-in-higher-education/184092