Object-Oriented Design
Topic Paper #11

Alex Laird
CS-3210-01
2/20/09
Survey of Programming Languages
Spring 2009
Computer Science, (319) 360-8771
alexdlaird@cedarville.edu

Grading Rubric		Max	Earn
	On Time/Format	1	
	Correct	5	
	Clear	2	
	Concise	2	
	Total	10 pts	

Peer Reviewer

ABSTRACT
This paper describes structural concepts of object-oriented design: abstraction, encapsulation, inheritance, and polymorphism.

Keywords
Object-Oriented, Design, Abstraction, Encapsulation, Polymorphism, Inheritance

1. INTRODUCTION
The concept of object-oriented design was first introduced with the Simula programming language in the 1960s. However, the first programming language to be called "object-oriented" was Smalltalk in 1969. Since then, the concepts of object-oriented programming, including data abstraction, encapsulation, polymorphism, and inheritance mainly, have been used to deepen the usability of powerful, modern languages.

2. OBJECT-ORIENTATION
Up until the concept of object-oriented programming, programs had been seen as purely structural, completing a list of instructions and the considering itself finished. Introducing objects into the equation, more specifically, giving programs a way to methodically store data within themselves and defining discrete, reusable portions of code, gave programming languages a vast new horizon on which they could build.

In object-oriented design, a program is a collection of objects that are a list of tasks (subroutines) which need to be performed before the program comes to a close. Each object has the ability to send data, receive data, and perform specific operations within itself [1].

3. ABSTRACTION
Abstraction is a concept that everyone does regularly; separating larger tasks into smaller, more manageable ones. In object-oriented programming, this can be achieved, for example, through casting an object to a related type, perhaps for more functionality *or* for protective means to hide functionality [3].

4. ENCAPSULATION AND POLYMORPHISM
"All object-oriented languages that are based on classes depend on the class as the fundamental unit of modularity" [2]. To encapsulate is to seal or to keep hidden to inner workings of the class; an object has specific operations that can be used to manipulate messages or information contained within the object, but the rest of the program does not necessarily need to be aware of how these functions work, it only needs to know that they functionality is there.

Similarly, related objects may have similar functionality, but one object may more specifically define the functionality. For instance, many things may be able to speak(), but derived classes may implement this different; a cat would meow() and a dog would bark().

Polymorphism and encapsulation work hand-in-hand, as you can hide a more specific method by calling the parents method instead [2].

5. INHERITANCE
Subclasses, which are things that inherit characteristics from a parent class, are what define inheritance. Inheritance says that a child class "is a …" of the parent class. For instance, a collie is a dog, therefore collie would be a subclass of dog and would inherit all the characteristics of a dog.

Inheritance is known as one of the most useful and most problematic features of object-oriented programming. One of the previously larger debates (though it has diminished a bit with more recent languages) is whether multiple-inheritance should be allowed or not. Many languages these days, such as Java, choose to disallow multiple inheritance directly (though there are ways to work around this) [3].

6. CONCLUSIONS
Objects have redefined programming and made it much more versatile. The additional baggage that comes with object-orientation is little in comparison to the massive amounts of benefits that have come from object-oriented programming.

7. REFERENCES
[1] Sebesta, Robert W., 2008. Concepts of Programming Languages. Addison-Wesley, Boston. ISBN: 978-0-321-49362-0

[2] Schärli, N., Black, A. P., and Ducasse, S. 2004. Object-oriented encapsulation for dynamically typed languages. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Vancouver, BC, Canada, October 24 - 28, 2004). OOPSLA '04. ACM, New York, NY, 130-149. DOI= http://doi.acm.org/10.1145/1028976.1028988

[3] Taivalsaari, A. 1996. On the notion of inheritance. *ACM Comput. Surv.* 28, 3 (Sep. 1996), 438-479. DOI= http://doi.acm.org/10.1145/243439.243441

Alex Laird, Cedarville University, Cedarville, Ohio, 45314