IT-1025

Alexandria Macko-Millard

Object-Oriented Programming

In the field of computer science, programming is the process of designing and bringing that design to life using a language that a computer understands. Over the course of a few decades, programming languages began offering more and more libraries and utilities ensuring that developers can write less lines of code to achieve a desired outcome in their programming endeavors. Initially, all code was procedural, meaning that the code was executed in the order it was written but over time the desire to reuse code to reduce redundancy resulted in a new method of programming, object-oriented programming. Object-Oriented programming is made of four pillars: abstraction, encapsulation, inheritance, and polymorphism.

Abstraction is a process that takes place in the design level of development and can be defined as identifying which details are necessary and visible to the user, hiding any irrelevant information. Abstraction can be implemented using Abstract Classes, which is a class inherited by child classes that uses the methods and properties of the Abstract Class. This ties in very closely with Encapsulation since Encapsulation is “the mechanism by which Abstraction is implemented” (C#Corner). With encapsulation, the developer prevents access to implementation details by enclosing code in an object, package, or method while using access specifiers to “define the scope and visibility of a class member” (Tutorialspoint). Using the Microsoft language C#, the access specifiers are Public, Private, Protected, Internal, and Protected Internal. In procedural programming, all information is immediately available to the user. For example, if you wanted to make coffee, your procedural program would outline each

step of making the coffee like getting water, crushing the beans, adding a filter, etc. With abstraction, the developer would simply call a method called “MakeCoffee()” which would take care of all the specific steps of the coffee making process.

Inheritance is the ability to create a parent-child relationship between classes. A parent class, or base class, is a template containing properties and methods that the child class inherits. An example of this would be a “Person” class that contains the person’s name, height, weight, birthdate and other personal details. A child class of “Person” could be “Employee” since an employee is a person but with additional properties such as Employee ID and Salary. The parent-child relationship creates a tree-like class hierarchy meaning that the parent-class will be at the top of the tree and multiple child-classes with their own sub-classes would all inherit from the parent class (Stackify). Class modifiers (as mentioned in encapsulation) are used to either allow or deny access to any parent class properties. If you had information in the parent class that you did not want your child classes to access, you could make the proerty “private” instead of “public”.

Polymorphism is the ability of an object to take many different forms. There are different kinds of polymorphism and the type used in the developer’s program depends on “when the object takes its form (compile-time or dynamic) and what part (method or object) of the object is transforming” (bmc blogs). There are four different types of polymorphism: Subtype polymorphism (runtime), Parametric polymorphism (Overloading), Ad hoc polymorphism (Compile-Time), and Coercion polymorphism (Casting). The first type, subtype polymorphism is the most common kind of polymorphism. With subtype polymorphism, one class can be used to refer to different subtypes. An example would be using a “Person” class to

generically refer to an “Employee” or “Patron.” Both an employee or patron would have person fields of first name, last name, birthday, etc so referencing these subclasses interchangeably with the “Person” parent class works. Parametric polymorphism, also known as overloading, is the ability of one function to handle multiple data types. Using parametric polymorphism, a method may take in a list of objects and manipulate them regardless of the type of the object. Ad hoc polymorphism is the ability to create multiple methods with the same name and different data types, allowing the program at runtime to decide which method to used based on the data type of the parameter passed to the method. If you had two methods with the name “GetInfo” with one method taking a string as a parameter and one method taking an integer as a parameter, the program will decide which GetInfo() to call based off of the parameter being an integer or a string.

In summary, we have discussed the four main components of object-oriented programming, abstraction, encapsulation, inheritance, and polymorphism. Using the pillars of object-oriented programming, a developer can create a robust program using very few lines of code compared to a procedural approach.

Works Cited

1. 23, October, and Jonathan Johnson. “Polymorphism in Programming.” *BMC Blogs*, 23 Oct. 2020, https://www.bmc.com/blogs/polymorphism-programming/#.
2. JanssenThorben “OOP Concept for Beginners: What Is Inheritance?” *Stackify*, 30 Mar. 2021, https://stackify.com/oop-concept-inheritance/.
3. *C# - Encapsulation*, https://www.tutorialspoint.com/csharp/csharp_encapsulation.htm.
4. “Abstraction and Encapsulation.” *C# Corner*, https://www.c-sharpcorner.com/blogs/abstraction-and-encapsulation1.
5. KnowledgeHut. “Abstraction in C# Tutorial with Examples.” *Knowledgehut*, 1 Mar. 2019, https://www.knowledgehut.com/tutorials/csharp/csharp-abstraction#:~:text=Abstraction%20in%20C%23-,Abstraction%20is%20an%20important%20part%20of%20object%20oriented%20programming.,base%20classes%20with%20partial%20implementation.