What are the four pillars of Object-Oriented Programming? Explain each pillar.

The four pillars of Object-Oriented Programming are Abstraction, Encapsulation, Inheritance, and Polymorphism.

Abstraction is the process of finding repeating code and creating a reusable function to be used in multiple places. This allows the programmer to make code easier to read and write. Instead of having to change many lines of code to through an entire codebase, you can use abstraction to create small digestible functions that are easy to understand and easy to update.

Encapsulation is the process of hiding information or limiting what parts of the code are accessible. This could be done by adding information that isn't returned but is still held in an object, by specifically creating private functions or variables to a class, or by closures(making data inaccessible via local scope rather than global). You can also take long lines of code and separating it into smaller functions, hiding away the long lines of code in the methods rather than having a big block of code that is difficult to read. This applies to Abstraction as well.

Inheritance is letting one object acquire the properties and methods of another object. If you have many objects with similar properties, but different uses, you can create a *parent* class and give it all of the similar properties. You can then create a subclass, or *child* class that *extends*, or includes the properties of, the *parent* class. In the subclass, you can then put all of the things they do differently while still being able to access the methods for which they are similar.

Polymorphism is a method having the same name but a different implementation based on the class it is in, by either **method overloading** or **method overriding**.

Method overloading is having different numbers of arguments for a method. In the Parent class, you might see a function *example* that takes two arguments (a,b), but in the subclass you might have the same function with three arguments(a,b,c), and it performs a different task. This is an This is an example of Method Overloading.

Method overriding is using function or method with the same name in a subclass to *replace* the parent class's implementation. You might see *speak()* under an Animal parent class that returns "Animals make noise" , but under the Dog

subclass, the same *speak()* method now returns “The dog barks.” This is an example of Method Overriding.

What is the relationship between a Class and an Object?

A Class is a special function that is the blueprint, or template, for creating objects. A class will use the keyword *Class* and will always include the method *constructor().* Inside the constructor method, you can add properties that the created objects will have. An object is an item with properties. A Class is not an object, but an object is an instance of a class. You can also create classes manually without the use of a class.

Citations:

Parr, K. (2020, December 18). *The four pillars of object-oriented programming*. freeCodeCamp.org. Retrieved March 3, 2023, from https://www.freecodecamp.org/news/four-pillars-of-object-oriented-programming/

Object-oriented programming - learn web development: MDN. Learn web development | MDN. (n.d.). Retrieved March 3, 2023, from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_programming

Classes - javascript: MDN. JavaScript | MDN. (n.d.). Retrieved March 3, 2023, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

JavaScript classes. (n.d.). Retrieved March 3, 2023, from https://www.w3schools.com/js/js_classes.asp