**Polymorphic Class Diagram**

Michael Maye

Colorado State University Global

CSC470: Software Engineering

Dr. Shaher Daoud

07/16/2023

**Polymorphic Class Diagram**

Object Oriented Programming is built around the four foundational principles Encapsulation, Data Abstraction, Polymorphism, and Inheritance. Throughout this analysis, we will focus on Polymorphism. Similar to Inheritance, Polymorphism allows objects of different classes to act as their superclass parents. While Inheritance allows child classes to take on methods of the parent, Polymorphism allows the system to treat unique subclasses differently when using those methods from the superclass, allowing for increased flexibility when implementing these classes and avoiding having to refactor classes when implementing them into different objects. The system implements dynamic and late binding to accomplish polymorphism, allowing runtime discovery of how the system interacts with these polymorphed objects and how their functions operate.

**Late and Dynamic Binding**

Late Binding is the process of pausing or deferring the binding of a method call to its implementation until the system's runtime, meaning that method calls do not resolve at compilation but rather during actual run time. This type of binding allows for a significantly more flexible system design by allowing new classes and methods to compile fully before deciding how the system implements methods.

Dynamic binding, also known as runtime binding, is one aspect of Late Binding that makes the runtime decision about which method to call based on the type of target object, which enables developers to create different implementations of a class, and with the use of override methods, define how that parent methods will behave on that object.

**Polymorphism Class Diagram Example**

In the example Polymorphism Class Diagram (Figure 1) and corresponding Pseudocode (Figure 2), the 'Garage' object can hold different vehicles, including 'Car', 'Motorcycle', and 'Bicycle'. The 'Vehicle' class is parent to each of these, and they all inherit its methods. The Pseudocode shows an example of adding the objects to the garage and then calling the move() method on each with a for-loop. Each vehicle has the move() method available as they are children of the 'Vehicle' class, but the 'Car' and 'Motorcycle' also contain variables that track their gas levels. In addition, the move() method is treated differently for each vehicle type. The 'Car' and 'Motorcycle' move() methods each return a new gas level to update that vehicle's gas after the movement. The 'Bicycle' class does not contain this return, as a bicycle does not use gas to move. When the system reaches the for-loop at runtime, it will simply call the vehicle.move() method for each object. With Polymorphism, and because of the Dynamic Binding, the system will know that it must return the new gas level when it calls this method on a 'Car' or 'Motorcycle'. When it calls it on the 'Bicycle' it is calling it on a vehicle that does not return a gas level, even though the move() method is explicitly being called from the 'Vehicle' class, not from each separate subclass.

**Figure 1**

Polymorphism Class Diagram

A screenshot of a computer

Description automatically generated

**Figure 2**

Polymorphism Class Pseudocode

Garage myGarage = new Garage();

myGarage.addVehicle(new Car());

myGarage.addVehicle(new Motorcycle());

myGarage.addVehicle(new Bicycle());

for (Vehicle vehicle : myGarage.getVehicles()) {

vehicle.move();

}

**Conclusion**

The Dynamic binding nature of polymorphism allows objects of different classes to be treated uniformly through a common interface while retaining their specific behaviors in method calls. As this binding moves decisions on method calls to runtime, Polymorphism is a runtime characteristic of programming.

**References**

12.3. polymorphism in depth. (n.d.). https://icarus.cs.weber.edu/~dab/cs1410/textbook/12.Polymorphism/poly\_and\_virt.html

Inheritance and Polymorphism. Inheritance and polymorphism. (n.d.). http://www.cs.utsa.edu/~cs3443/notes/chapter10/ch10.html

JavaTPoint. (n.d.). Dynamic polymorphism in Java. www.javatpoint.com. https://www.javatpoint.com/dynamic-polymorphism-in-java

Polymorphism. Polymorphism (The JavaTM Tutorials &gt; Learning the Java Language &gt; Interfaces and Inheritance). (n.d.-a). https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

Polymorphism. Polymorphism (The JavaTM Tutorials &gt; Learning the Java Language &gt; Interfaces and Inheritance). (n.d.-b). <https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html>

Unhelkar, B. (2020). 11. In *Software engineering with UML*. essay, CRC PRESS.