**1-5 Assignment: UML Diagrams**

Cristiano Miranda

Southern New Hampshire University

CS-230: Operating Platforms

Professor Sarkar

7 September 2025

**1-5 Assignment: UML Diagrams**

            The object-oriented programming paradigm emphasizes the creation of code by separating all aspects of a project into distinct objects, each holding the methods and attributes associated with that portion of the project. In doing this, objects can be reutilized in different parts of the code without requiring rewriting or recreating the same object in multiple places. Additionally, this helps to better organize all code by grouping associated methods and attributes, which has the added benefit of making it easier to find a desired method or attribute, as all one would have to do is search through the object associated with the method or attribute they wish to find, as opposed to looking through the entire code base. The OOP paradigm achieves this functionality through its emphasis on core principles, including portability, inheritance, encapsulation, and polymorphism. The UML class diagram presented below demonstrates all the principles using its inherited classes, overloaded constructors, and various other attributes and methods housed within the classes. To start, potability is present via the use of the classes to construct these objects as this allows them to be initialized multiple times and at any place in the code without requiring the programmer to rewrite the same code multiple times, which keeps the code dry and helps makes the code easier to maintain as any changes to the object needs to be performed in only one place as opposed to multiple points in the code. Additionally, portability is presented via the fact that the various classescan be copied and reused in other programs without compromising their functionality. For inheritance, this is presented in the class model via the *Bicycle* class inheriting attributes and methods from the *TwoWheeled* class and the *TwoWheeled* class inheriting attributes and methods from the *Vehicle* class. As opposed to rewriting an attribute or method multiple times in various classes, inheriting said attributes or methods from a parent class allows multiple child classes to utilize the attributes or methods without requiring them to be written in their class. In doing this, rather than having to write the same code multiple times for any child class, the parent class can house said attributes or methods and pass them down to its child classes. Also, a child class inheriting attributes or methods from its parent classes presents the principle of portability since it allows multiple classes the potential to utilize the same parent class to avoid the need to rewrite the same attributes or methods multiple times in different classes. In terms of encapsulation, this is evident in the *Bicycle* class, which houses all the information and logic associated with a bicycle object. Rather than having the associated code spread throughout the program, making it harder to find, everything is housed in a singular place, in this case, in the *Bicycle* class and its parent class, which also allows for control over which attributes and methods are accessible outside of the class and which are not. Finally, the principle of polymorphism is presented via the methods of the *Bicycle* class, and specifically the Bicycle constructor method and the outputData methods. The Bicycle constructor is declared four times, and the outputData method is declared twice; however, each declaration of each method utilizes different parameters. Thus, when a call is made on the object to the specified function, and depending on the parameters passed in, its associated method would trigger, which makes it easier to reuse the same method name and purpose but with different data types to avoid the need to utilize differently named methods for specific data types. Through this, code becomes more straightforward as the same method name can be used even with different data types to reduce the overall complexity of the code. Overall, the UML class diagram is a good example of all the OOP paradigm principles as it highlights portability using a class that can be reused in multiple places in a project or even exported to another project, inheritance via the child class *Bicycle* inheriting any attributes or methods within its parent class *TwoWheeled*, encapsulation via all attributes and methods relating to an object being housed within a singular class, and finally polymorphism, via the similarly named methods but with each having different parameters to handle different data types.

