Chris Williford

Southern New Hampshire University

CS230: Operating Platforms

Tracey Lanham

March 06, 2025

**UML Class Diagram**

A diagram of a vehicle

AI-generated content may be incorrect.

The diagram effectively represents inheritance, encapsulation, and abstraction, which are central to object-oriented programming.

**Inheritance:**

The ‘**Bicycle**’ class inherits from the **‘TwoWheeled’** class, which in turn is likely part of a broader class hierarchy involving the **‘vehicle’** class. This means the **‘Bicycle’** class inherits the properties and methods of the **‘TwoWheeled’** class, allowing for code reuse and maintaining a hierarchical relationship. The diagram shows a triangle with a solid line, indicating inheritance.

**Encapsulation:**

The classes are encapsulating their properties and methods. For example, the **‘Bicycle’** class has private properties such as **‘gears**, **‘cost’**, **‘weight’**,and **‘color’**, and public getter and setter methods for accessing and modifying these properties. This is a standard OOP practice that keeps the internal state of an object hidden and only accessible via well-defined methods. The **‘outputData()’** method is an example of an encapsulated method to display the properties of a **‘Bicycle’** object.

**Abstraction:**

Abstraction is seen in the way that classes like **‘Vehicle’**, **‘TwoWheeled’**, and **‘Bicycle’** define the general structure of the objects but may not specify every detail (such as implementation of specific methods in the abstract or parent classes). This allows users to focus on high-level operations, hiding the complexity of individual implementations.