**1-5 Assignment: UML Diagrams**

Emerald Tresch

Southern New Hampshire University

CS 230: Operating Platforms

Professor Emilia Butu

September 7, 2024

A diagram of a diagram

Description automatically generated

We can see two types of relationships, Association and Inheritance, within the Java program. The first is the Association relationship, which can be seen in the interaction of the Driver and Vehicle classes. The following relationship is represented through the Inheritance Object-Oriented Programming (OOP) principle, which is seen in the diagram, which shows an inheritance hierarchy where Bicycle inherits from TwoWheeled, which in turn inherits from Vehicle. This relationship allows Bicycle to use or override methods defined in TwoWheeled and Vehicle. This relationship is shown in the program Bicycle is extended from TwoWheeled, and TwoWheeled is extended from Vehicle.

While the Inheritance OOP Principle can be seen in the UML diagram through the open arrows, the program also uses Polymorphism, Encapsulation, and Abstraction. Starting with polymorphism, an overridden method or interface is implemented to illustrate polymorphism. It enables one interface to be used with several types. This can be seen when the Bicycle class uses the overriding method. The class contains two methods named "outputData," one method has no parameters, while the other takes a bikeText parameter. The Bicycle class offers flexibility in calling the process based on specific requirements by providing various versions of the same method.

The following OOP Principle is Encapsulation. Classes encapsulate data and behavior. Private attributes and getter/setter methods show encapsulation. The Bicycle class encapsulates its attributes by providing Accessors (getters) and Mutators (setters) methods for each attribute, which are used in the program to control access to and modify the attribute values. Encapsulation ensures data integrity and creates a straightforward interface for interacting with the Bicycle class.

The Java program is built around an abstract Vehicle class, a TwoWheeled interface, and a Bicycle class that extends Vehicle and implements TwoWheeled. The Driver class operates vehicles by calling their methods. The Main class demonstrates the operation of a bicycle by a driver, showcasing the object-oriented principles of inheritance, polymorphism, and interface implementation through a UML class diagram.