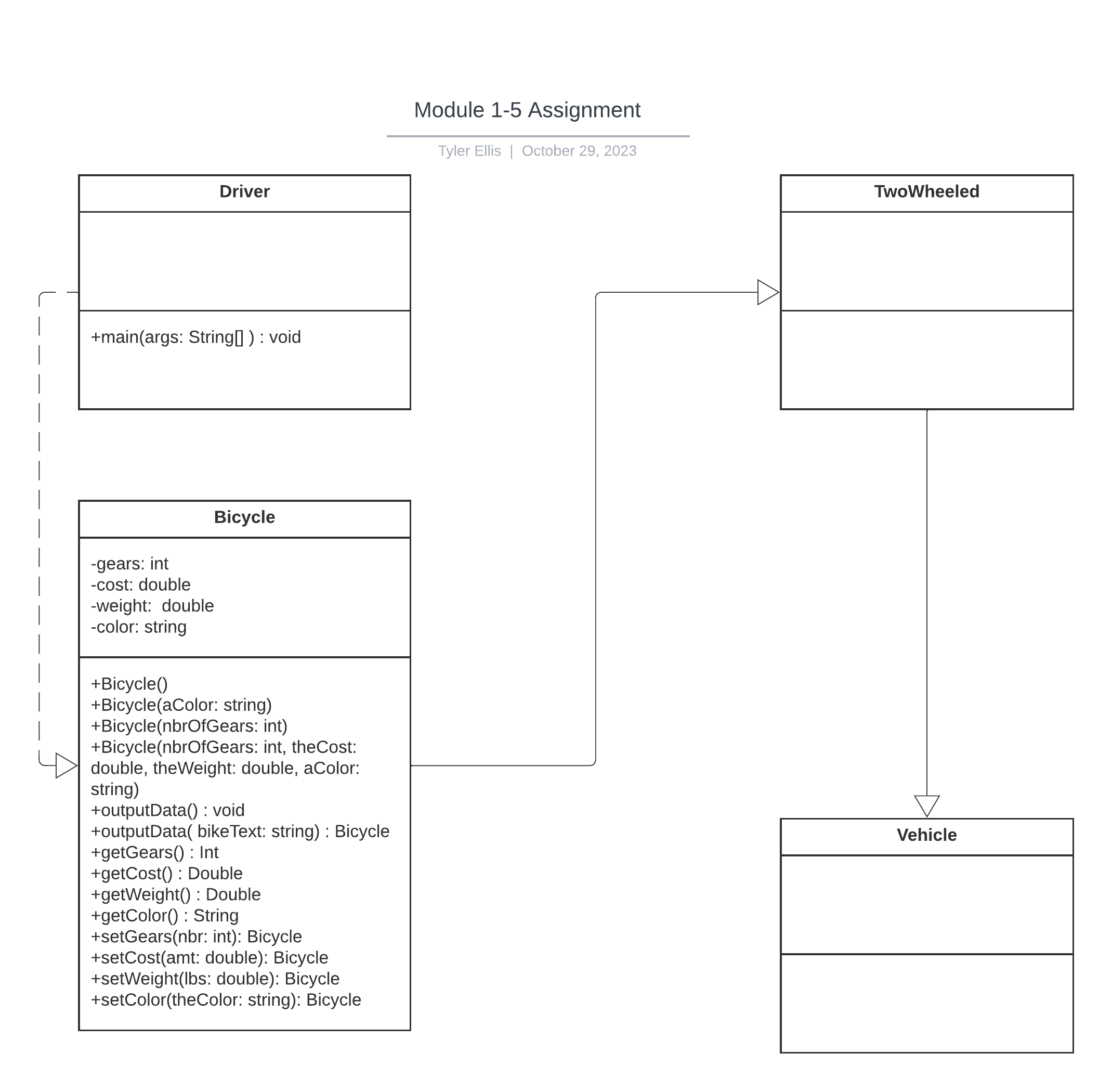
CS-230 Operating Platforms

Tyler Ellis

[Tyler.Ellis1@snhu.edu](mailto:Tyler.Ellis1@snhu.edu)

Southern New Hampshire University



Several primary concepts of Object-Oriented Programming can be seen in the UML diagram. The inheritance notion is clear given that the Bicycle class is displayed as an instance of TwoWheeled, which stems from the Vehicle class. This structure illustrates the typical arrangement among objects, enabling code replication and the creation of a class hierarchy. The architecture of the Bicycle class displays encapsulation, which is another aspect of OOP. Private variables in the class, like gears, cost, weight, and color, are protected, while public methods provide controlled access. This helps in protecting internal data and preserving data security.

The inheritance architecture emphasizes polymorphism, which is not openly apparent in the program. Polymorphism would be more recognizable if the Bicycle class overwrote the superclass methods found in either the Vehicle class or TwoWheeled class. However, the Bicycle class's overloaded method outputData() offers a weak indication of it. This is due to the fact there are two outputData() methods. One that takes a string argument and one that takes no arguments. Depending on how the method is called, this would be a form of compile-time polymorphism. The UML diagram also introduces abstraction, an important OOP principle that promotes the concealment of specified attributes, showing only what is necessary. The UML diagram does this by establishing classes and their hierarchies. The design gives a basis for attainable wide abstractions in future iterations by establishing core classes like Vehicle and TwoWheeled, even if they are currently empty.