**Question 1: Inheritance vs. Composition**

Inheritance and composition are two techniques for building relationships between classes and objects in a script, and for avoiding repeated code. Where inheritance follows an *is a* relationship, composition follows a *has a* relationship.

For example, a bike *is a* mode of transportation. Each mode of transportation has several characteristics in common, such as maximum speed and the type of fuel consumed. But a specific mode of transportation, such as a bike, also has its own set of specific characteristics, such as how many gears it has and whether it has training wheels. Using inheritance, our bike class would inherit our modeOfTransportation class.

class modeOfTransportation {

constructor (maxspeed, fueltype) {

this.maxspeed = maxspeed

this.fueltype = fueltype

}

}

class Bike extends modeOfTransportation {

constructor (maxspeed, fuel, numgears) {

super(maxspeed)

super(fueltype)

this.numgears = numgears

}

}

A bike also *has a* set of specific parts; it has some wheels, a bike seat, a chain, and a set of handlebars. I can use these components to build a bike, and I can also use them to build other things. For example, I could use a bike seat to build a regular bike or a stationary bike. Note that the latter is not a form of transportation, which demonstrates another key difference between inheritance and composition: inheritance focuses on what the class *is*, and composition focuses instead on what the class *does*. For both a regular bike and for a stationary bike, the bike seat serves the same purpose of providing a place for the rider to sit.

Sources:

<https://betterprogramming.pub/inheritance-vs-composition-2fa0cdd2f939>

<https://www.infoworld.com/article/3409071/java-challenger-7-debugging-java-inheritance.html>

<https://www.w3schools.com/js/js_class_inheritance.asp>