Sofia Valiante (40191897)
Assignment 2
March 4th, 2022

Part 1, Question 5: Investigate the output of your program! You will need to submit a separate document (pdf, MS-Word, or text) along with your assignment indicating whether or not the shown display from the findLeastAndMostExpensiveUAV() is correct. In either case, you need to explain the reason for the correctness or failure of your program's behavior/output.

I believe that the program's behavior is correct. The static method considers two for loops, the outer for loop is meant to represent the index of the current UAV, and the inner for loop next UAV object. In the outer loop, the current object is checked whether it is a UAV or child object of the UAV class, like MAV and AgriculturalDrone, if so then the current object is compared to the next object that is also a UAV or child of a UAV, also checked with another conditional if statement. Once both objects are confirmed to be of UAV or a child of UAV, their class is then determined, and they are then downcasted to their respective class so that the method .getPrice() is able to actually access the attributes of the object. This is done because the flying objects are actually passed by into the method as being of the superclass Object, so the flying objects must therefore be downcasted to access their information. After each respective UAV object is downcasted, their prices are compared using a conditional if statement that swaps the locations of the UAV's if the next UAV's price is higher than the current UAV's price. After the swapping is complete, they should be ordered in the original passed array of type Object[] in decreasing order. Thus, in order to print out the information, the first instance of a UAV in the passed array will be the UAV with the highest price and the last will be the cheapest UAV. Thus, a total count is made of all UAV objects in the passed array done with a for loop. Then, another for-loop passes through the same passed array again but employs a conditional statement that prints out the information of the UAV using a toString() when a second counter (uavNb) is incremented, where the uavNb = 1 is the first UAV appearance in the array and when uavNb = counter, it means that the uavNb counter is at the last appearance of a UAV in the passed array. Thus, the information is printed out. If there are no UAV's in the array however, then uavNb will never be incremented and will therefore equal zero, so another conditional statement is used to print out that there are no UAV's in the array.

Part 2, Question 3: Does my program work correctly, or does it misbehave! WHY? Display the contents of both arrays, then submit a separate document (pdf, MS-Word, or text) along with your assignment indicating whether or not the copying is correct. In either case, you need to explain the reason for the correctness or failure of your program's behavior/output.

The copying is correct. The printed initialized array using the toString() method has the same information as the information printed from the array containing all the copies. The method has the initialized array as its argument, thus the array of multiple flying objects from differing classes is passed to the method, however the array is of the superclass type Object so that the

array can hold all the different types of objects. Then, inside the method, a new array of type Object is created to be the size of the passed array. A for loop is then used to traverse both arrays with a variable ‘i’ declared so that the index at ‘i’ refers to both arrays, so that the object at array ‘i’ of the passed array can be copied to the new copy array. In order to do so however, the correct constructor must be used for each type of the object within the array at the current index. Thus, a String is used to hold the class name of the current flying object from using java library getClass().getSimpleName() methods. A switch is then used to identify which class the current object is, where the current object is at the index ‘i’ of the for loop, and then at each instance of the switch statement, the appropriate copy constructor is called. Note, in this case, the object in the array at the current index ‘i’ actually of type Object, even though they have all of the attributes of a flying object from one of the classes Airplane, Helicopter, Quadcopter, Multirotor, UAV, MAV and Agricultural Drone. Thus, by calling the copy constructor and passing it the argument of the object at the current index, you are actually passing an object of the class Object to the copy constructor instead of the specific class you wish the flying object to be. However, this is allowed by my copy constructors because each copy constructor takes an Object as an argument, and downcasts the argument to be an instance of the respective class, thus the copy constructor already assumes that the object as the argument already has all the attributes of the class, this is why this downcasting is allowed by java. Then, the downcasted object's information is copied to a new object that will be outputted by the constructor. Thus, the actual copying of all the attributes of the instance of the class is allowed as well. Given that you’re now assigning an object of a subclass to the array of the type of the superclass is allowed and will therefore be outputted correctly as an array of type Object with all the copied flying objects.