***PART I***

The first part of our assignment works perfectly as when we pass an array which contains both UAV and Airplane objects, the code filters (instanceof) and only compares the UAV objects. It prints out the cheapest and the most expensive UAV objects. In case only one UAV object is present in the array, the program informs the user that only one UAV object is present and prints its information. If no UAV objects are present, then the program displays the output accordingly.

***PART II***

The second part of our program does not work properly. The copying that takes place is improper. For example, a helicopter object with the following information:

This Helicopter is manufactured by null in 2000. It costs 1.0$ and has horsepower of 5HP. It has 4 cylinders and passanger capacity of 2 passengers.

is copied as:

This Airplane is manufactured by null. It costs 1.0$ and has horsepower of 5HP.

This happens because we were restricted to not find the exact type/class of the object. I only used instanceof to filter between airplane and UAV and use appropriate copy constructor. Since we only used the copy constructors of Airplane/UAV, the new object created will be an Airplane/UAV and when we pass objects of their child classes only the attributes of Airplane/UAV class get copied to the new object.

We expected polymorphism to perform its “magic” but this did not happen. Let’s revisit the 3 conditions for polymorphism to work. First condition was inheritance and we surely do have it. Another condition was confused casting and we did cast the objects of the array to Airplane/UAV before passing them as a parameter in the copy constructor. The last condition was overriding of methods, and this is where polymorphism breaks down. The copy constructors of each class have different names and were not overridden. This is the reason why polymorphism did not work, and we did not get an actual copy of the array.

This error can be avoided by using the clone() method which is inherited from the Object superclass. A simple example for implementation of clone() in, let’s say the Multirotor class, could be:

public Multirotor clone(){

return new Multirotor(this)  
}

Although the clone() methods may in fact use the copy constructors to perform the copying, this works because the method clone has the same name in all classes, and this fulfils our condition of overriding methods which was not fulfilled in our case. Hence, with all the three conditions satisfied, polymorphism will perform its “magic” but sadly, we were restricted to not use clone in our assignment.