Question 1

(a) interface A has no supertype
interface B has a supertype of A
interface C has a supertype of A
class D has supertypes of A, B, C
class E has supertypes of A, C
class F has supertypes of A, B, C, D
class G has supertypes of A, B

(b) Classes D, F, G

(c) No, because C is not a supertype of A, and it is not guaranteed that `a` is going to be of type C.

Question 3

Inheritance is the concept of creating derived classes from existing base classes to extend behavior and reuse code. Subtype polymorphism is the ability to substitute a subtype in for a supertype. They are implemented (usually) only in statically typed languages, as dynamically typed languages use duck typing instead. Dynamic dispatch is when the program determines at runtime the correct function to invoke.

Question 4

We cannot use subtype polymorphism in dynamically typed languages since types are bound to values, not variables. Thus, there is no way to know if a parameter of a function takes in a subtype, supertype, or an unrelated type. Thus, the notion of replacing a subtype for a supertype is undefined. We can use dynamic dispatch in dynamically typed languages, and it is used for duck typing. At runtime, the program consults (usually) the vtable pointing to the object, and whichever function the vtable points to is the one that gets invoked.

Question 5

These classes violate the Dependency Inversion Principle. Here, `ElectricVehicle` uses `SuperCharger`, so rather than directly using `SuperCharger`, we should define an interface `Charger` that `SuperCharger` implments. Then, have `ElectricVehicle` use `Charger`.

Question 6

Yes. This is because even dynamically-typed languages should exhibit structure when using the OOP paradigm. Thus, having a proper class hierarchy is essential for good OOP practice.