Please answer these questions:

1. You have a if..then..else statement in any language that returns a class instance from each result. In each Boolean statement there is a different series of code that creates and modifies the class. What would you do if you wanted to make this more extensible, i.e. you had to continually add more ‘if’ statements and the length and complexity was getting unmanageable. Is there a pattern or a way to change the code from a Boolean statement into something more encapsulated and compartmentalized?
2. What would be the preferred way in the example in the last question to implement your code?

.NET OOP Coding Test

1. Open Visual Studio and create a ASP.NET CORE API MVC project. Use version 6.0 or 5.0 CORE. Call the project OOPCoreApi.
2. In the project create a controller (name it HelloWorldController)
3. Create a folder called Service. This will house all interfaces and service classes.
4. Create an Interface called IHelloWorldService. Create a default implementation class with appropriate naming for the interface.
5. Add a folder called Extension, in this put an extension class named AddCompanyServices to house your service dependency injections calls. You should implement as a service the IHelloWorldService service from here.
6. Make the IHelloWorldService service be scoped as transient using appropriate dependency injection.
7. Add method to IHelloWorldService DoSomething with string as input and implement. This method should return a json class called SomethingModel. SomethingModel should have two properties Id (int) and Name(string)
8. Add method to IHelloWorldService DoSomethingElse with SomethingModel as input and implement. This method should return the previously created json class called SomethingModel.
9. Add two API methods to the HelloWorldController: DoSomething and DoSomethingElse
10. DoSomething is a GET that returns the matching service call from IHelloWorldService
11. DoSomethingElse is a POST that returns the matching service call from IHelloWorldService.
12. Make sure swagger UI is added to project so it can be run through swagger
13. For extra credit create a Test project to test your code