**1. Diagram of the inheritance relationships between the classes:**

I have used hierarchical inheritance which has only one super class (**Car**) and many sub class (**CarFactory and model**).

**Car**

**Model**

**CarFactory**

**2. The explanation of why should I use method overriding in the subclasses:**

Method overriding is mainly used in a situation when we want to create a more specialized version of the general method inherited from a superclass within the subclass. It helps writing general code based on parent class or interfaces as object resolution happens at runtime and it also provides multiple implementations of the same method and can invoke parent class overridden method using super keyword. In method overriding the name and parameter can be same, no errors will occur here. It allows run-time polymorphism which in turns helps in write robust code. Method overriding is increasing the reusability the code and it also helps in progressive requesting where we can move from general to explicit. That’s why I should use method overriding in the subclasses.

**3. I choose an abstract class,**

An abstract class is a special kind of class which can’t be instantiated. We can only instantiate the subclasses of the abstract class if they are not abstract. I am choosing an abstract class because I can group several related classed together as siblings. Grouping classes together is important in keeping a program organized and understandable. It is a template for future specific classes. The main reason why I am choosing is if there is some code that has to be shared between classes of the same type. Usually, I can use an interface for this, but sometimes the functionality of such classes may overlap and I will end up with code duplication. Basically, an abstract class should be used for an “is-a” relationship. When someone wants to restrict all the subclasses must have certain functionality then abstract class with implementation is recommended.

**Brief documentation of every class in my program:**

1. **Main class:** The main class is for the users, what they really want to do, what kind of cars to build, or if they want to build a car, then there are various cars model and they have chosen one of them. And if they do not want to create a car, then they will be able to exit the program by typing no.
2. **Car class:** The car class is an abstract class, in this class, there is an abstract method and non-abstract method. In the non-abstract method, there is some basic structure of cars.
3. **CarFactory class:** This is the most valuable class where cars are created. It takes input from many classes and creates a car.
4. **ChoiceNo class:** If a buyer doesn’t want to create a car he can see how many cars company have created.
5. **Toyota class:** In this class, a buyer can input their choice able color, engine type and is self-driving or not if he wants to buy Toyota brand car.
6. **Volkswagen class:** In this class, a buyer can input their choice able color, engine type and is self-driving or not if he wants to buy a Volkswagen brand car.
7. **Mitsubishi class:** In this class, a buyer can input their choice able color, engine type and is self-driving or not if he wants to buy Mitsubishi brand car.
8. **Model class:** After creating a car company will give the buyer a model number of his car and the programmer is doing a program for this work in this class.
9. **CreatorInfo class:** This is the information class about the creator of the car, who actually create the car.