**4.**

- Genom att använda oss av MVC-design-pattern åstadkommer vi högre cohesion och lägre coupling samt följer Separation of Concern. Systemet blir alltså mer modulärt eftersom det inte förekommer starka beroenden mellan den grafiska modulen och "bil"-modulen. Att ändra den interna representationen för hur bilars position hanteras till en egen klass för detta gör att vi bättre följer Single Responsibility Principle och Separation of Concern.

- **Refaktoreringsplan**
    - Skapa ett interface refreshObserver
    - Skapa CarModel klass och dess implementationer
        - Lägg till variabler och metoder enligt UML design
    - Refaktorera CarController
        - Ändra berörda metoder efter implementation av CarModel enligt UML design.
    - Refaktorera CarView
        - Ändra berörda metoder efter implementation av CarModel och förändring i CarController enligt UML design.
        - Implementera interfacet refreshObserver och dess metoder
    - Refaktorera Drawpanel
        - Ändra berörda metoder efter förändringar i tidigare steg
    - Implementera DoublePoint i Vehicle och CarFerry
    - Skapa Application klass och implementera main-metod

- Man kan arbeta parallellt genom att varje utvecklare utför ett steg var i refaktoreringplanen eftersom att det handlar om olika klassfiler.