- 1Ändrar så att vehicle representation inte är beroende av point och BufferedImage utan istället använder int för att ta bort onödiga beroenden.
    - ändra point till två variabler x och y för att representera x och y koordinater.
    - Ersätta buffered image med ett index av dess position i en arraylist
    - Kan göras parallellt med alla andra steg

- 2 Skapar en klass Main som:
    - skapar en Timer (tidigare använts i CarController),
    - skapar en CarController (tidigare använts i CarView),
    - skapar en DrawPanel
    - skapar en main metod (tidigare använts i CarController)
    - skapar en inre TimerListener klass (tidigare använts i CarController)
- Eliminerar beroende av CarController från CarView genom att ta in beteende i konstruktor istället för att specifiera beteende i CarView, single responsibility principle.

- Lägger till ett interface Controllable som har alla metoder för gas, brake, etc, vilket gör att CarView inte är beroende av CarController. CarView blir också mer flexibel och återanvändningsbar, eftersom man kan skapa implementationer av Controllable för till exempel en lastbil. Gör att vi följer dependency inversion principle bättre eftersom klasser är beroende av ett interface istället för en klass.

- CarView's konstruktor tar nu in en JPanel (DrawPanel från Main), vilket gör CarView oberoende av klassen DrawPanel. Gör att vi följer dependency inversion principle bättre eftersom klasser är beroende av ett interface istället för en klass.

- Följer till viss del Model View Controller

Vilka ansvarsomraden har era klasser?

- CarController - initialiserar bilar, hanterar bilars positioner, kollision, ritar ut bilar, main
- CarView - Specifierar grafiska gränssnittet, kopplar knappars beteende med CarController
- DrawPanel - Laddar in bilder, sparar en representation av bilar, ritar ut dem
- CarTransport och CarWorkshop - Hanterar sig själva samt bilar som är lastade