Analysera era klasser med avseende pa Separation of Concern (SoC) och Single Responsibility Principle (SRP).

Vilka ansvarsomraden har era klasser?

Carcontroller - styra logik som typ gasa och liknande, koppla ihop DrawPanel med själva bilklasserna
CarView sköter alla knappar och liknande, vid tryckning på en knapp skickas detta till carcontroller
Drawpanel målar bara upp bilderna och ser till att bilderna rör på sig med moveit.
Resterande klasser för tidigare labb har funktionalitet som angår de olika delarna ex en för att röra på sig som alla har = vehicle, och vissa mer specifika grejer som Bed som representerar ett flak som kan lyftas upp och sänkas.

Vilka anledningar har de att forandras?

De vi kan ändra är framförallt
CarView kan vi dela upp då den nu både sköter knapparna och lägga till olika knappar. Detta hade kunnat delas upp så en del sköter knappar och en som bara lägger till knapparna. Enligt **single responsability principle**.
Vi kan också göra så att drawpanel får en relation direkt till carcontroller så vid initiering av bilar och liknande behöver man inte ha en point direkt i drawpanel, utan den kan använda punkterna som redan finns dvs x och y principen. Enlig **seperation of concerns** då drawpanel egentligen inte behöver göra något annat än att rendera punkterna.

Pa vilka klasser skulle ni behova tillampa dekomposition for att battre folja SoC och SRP?

Carview, Carcontroller, och Drawpanel är främst de som vi kan göra förändringar som nämnt ovan.

Refaktoriseringsplan

Planen bör bestå av ett sekvensfaktoriseringssteg som tar er fran det nuvarande programmet till ett som implementerar er nya design. Planen behöver inte vara enormt detaljerad.SoC Seperation of Concern, SRP Single Responsibility Principle.

1. Vi ska ha en class program som är som själva main funktionen, använder sig av en LogicPanel, där logicpanel styr all logik och kommunicerar med varje klass som har med renderandet osv att göra. Logicpanel ska ha instanser av de olika klasserna som krävs i sig.Som carbuttons, drawpanel etc.
2. Carbuttons ska implementeras så en egen klass håller koll på knapparna.(SoC)
3. Drawpanel ska sköta renderingen endast, och detta görs med att information skickas från LogicPanel klassen.(low coupling)
4. TimerListener skall vara en egen klass som hanterar “events” av användaren.(SRP)

Ja denna planen går att utföra parallellt då den handlar om att dela upp separata delar så jobba fler än en på detta borde vara möjligt. Exempelvis en jobbar på Vehicle och CarController och en annan jobbar på CarView