Refaktoriseringsplan

- Dela upp koden i olika package. Ex ett för fordon
- Från början höll vi koll på positionen på två olika ställen. Både i alla vehicles men också i drawpanel. Det är onödigt och den refaktoriseringen påbörjade vi redan innan vilket man ser i vårt första UML diagram. (Enligt single responsibility principle)
- Enligt principen composition over inheritance så vill vi även se till att inte använda arv i onödan för att minska complexiteten och öka flexibiliteten i koden.
- Drawpanel borde bara måla ut vad vår view innehåller men nu skapas det instanser av bilverkstad och nya points där. Det bryter mot SRP och bör fixas. Det är även där vi hämtar alla bilder, men eftersom bilden för en bil är något som rimligtvis bör tillhöra bilen själv så borde detta rimligtvis hanteras i de olika fordonsklasserna som Saab och Volvo. Enligt SOC är det rimligt då vi vill dela upp koden så att allt som har med en volvo att göra sköts på ett ställe, inte på flera olika ställen.
- Nu funkar det så att all input från användaren skickas till CarView innan det skickas vidare till CarController. Eftersom att det är carcontroller som ska ha hand om alla inputs och sen påverka viewn utifrån det så kan denna funktionaliteten rimligtvis flyttas dit också
- Vi tycker också att CarController fungerar lite konstigt nu. Det bygger till exempel upp hela modellen med bilar osv men med ett MVC mönster borde Modellen skötas i en egen modul. Vilket är rimligt enligt SRP. En modul borde sköta själva modellen, en modul tar emot user input som i sin tur uppdaterar modellen och skickar datan till våran view. Fixar hur bilden ska se ut och DrawPanel målar upp viewn för användaren.