Klassernas ansvar

Movable: Movable är ett interface som sätter upp ramverket för hur en bil rör sig.

CarModel: CarModel ansvarar för att implementera grunderna som krävs för en biltyp så som att gasa och bromsa samt det som defineras i movable.

Saab95: Saab är en subklass till CarModel som utökar dess funktioner som är specifika för saab.

Volvo: Motsvarande saab fast för modellen volvo.

Truck: Subklass till CarModels som definierar det attributer olika typer av lastbilar har. Utöver CarModels funktioner hanterar den även flak. (Kan delas upp så flak blir en egen klass med ett eget ansvar. Detta hade skapat bättre modularitet och referenser.)

Scania: Scania är en subbklass till Truck som ansvarar för de specifika krav en Scanialastbil har.

CarTransport: Motsvarande Scania fast för en biltransport, ex på nya ansvar är att kunna lasta på andra CarModels.

Garage: Garage är en klass som håller objekt av CarModels. Dess ansvar är att lasta in och ut ur garaget.

CarController: CarController ansvarar för logiken av bilarnas rörelse samt koppling mellan utförarkoden och resten av klientkoden, ex skapar panel och hanterar metoder för rörelse. (Kan delas upp mellan rörelse och logik. Detta hade skapat bättre modularitet och hade skapat svagare referenser mellan klasser.)

CarView: CarView ansvarar för att skapa panelen och dess knappar samt referenser mellan dem och olika metoder.

DrawPanel: Draw panel ansvarar för en klass med bildobjekt och metoder för dessa som samt att använda dessa för att visualisera bilarnas rörelse. (Kan delas upp för att separera skapandet av bildobjekten och visualiseringen. Även här bättre modularitet.)

UML förändring

Vi ändrade så att det inte fanns en mutual dependency mellan CarView och CarController. Vi vill inte ha för starka beroenden. Då blir det svårt att göra ändringar. Detta löses genom att skapa ett interface som kopplas till båda klasserna.

Vi ändrade image objects till en egen klass så att den går att använda på andra ställen samt att öka modulariteten.

Vi utförde dekomposition på truck och skapade klassen ramp för att även här öka modulariteten och förenkla referenserna.

Flyttade funktionerna för att testa bilarnas status till TimeListener. Detta på grund av modularitet.

Tog bort beroendet mellan CarControl och DrawModel.

Refaktoriseringsplan

1. Ta bort beroendet mellan CarController och CarView. Skapa ett interface som CarController implementerar och CarView refererar med metoderna för knapparnas actions.
2. Gör klassen ImageObjects till en egen klass så att den inte är nested med DrawModel. Efter att det är gjort kommer några referenser att behöva ändras.
3. Dekomposera klassen Truck till Truck och Ramp med relevanta metoder. Skapar en ny klass Ramp och kopplar Ramp till Truck med komposition.
4. Flytta metoderna touchesWall och touchesGarage till den nestade klassen TimeListener.
5. Skriva om referenserna till ImageObjects i touchesWall och touchesGarage för att uppnå LoD. Detta görs genom att skriva om getters i DrawModel.