Refaktoriseringsplan & ansvarsområden

Car
-abstrakt klass som hanterar alla metoder och definitioner av allt som berör bilar. Alltså vilka egenskaper en bil har och metoder som får bilen att röra på sig. Syftet med klassen är att samla gemensamma metoder som olika typer av Cars kan ha.

Saab95
-subklass till Car, en typ av Car som har några overridade metoder med unik funktion (turbo).

Volvo
-subklass till Car, en typ av Car som har några overridade metoder med unik funktion (trimfaktor).

Truck
- Subklass till Car som ska fungera exakt som en Car men har ytterligare saker, tar hänsyn till ytterligare saker som plattformar. Gör allt som Car med en plattform

Scania
-subklass till truck, en typ av truck som har en plattform med medföljande vinkel.

Transporter
-subklass till truck, en typ av truck med en plattform som kan ta in andra bilar.

Plattform
-En klass som är tänkt att användas i Truck för att hantera metoder och logik relaterat till plattformar.

Workshop
- En klass som ska representera en workshop som kan ta in bilar allmänt eller en specifik typ av bil.

Movable
- interface som innehåller metoder för att en bil ska röra på sig.

CarController
- CarController är en klass som tar in instruktioner från en Carview och uppdaterar tillståndet och grafiken i simuleringen. Den gör dock flera grejer i detta fall (uppdaterar grafiken och tillståndet av simuleringen)

CarView
- CarView är en klass som initialiserar ett fönster och knappar och kommunicerar med en CarController alltså skickar instruktioner till den genom att kalla på metoderna som finns i CarController.

DrawPanel
- DrawPanel är en klass som representerar den animerade delen av simulationen, alltså den handlar om all grafik relaterad till planen som bilarna kör runt på.

Problem med Carcontroller (massa pilar)
-gör en massa saker samtidigt

- skapar och håller koll på alla bilar och workshops
- Innehåller metoder som bilarna använder för att röra på sig
- Hanterar hur grafiken ska ritas enligt drawpanel som finns i frame vilket är en carView och ber drawPanel att rita om grafiken
- Hanterar även hur workshops fungerar
- Skapar då ett beroende av drawPanel och carView
- Actionperformed metoden är överkomplicerad // mindre modulärt
- Cirkulärt beroende med carView // dåligt
- Den kör också simuleringen i själva klassen då den också hanterar simuleringslogiken. // bryter mot SRP

Vad kan vi göra?

- Vi vill inte att carController hanterar grafiken så vi kan flytta det som berör grafiken till drawPanel. // vi tillämpar single responsibility principle
- Hitta något sätta att skapa bilarna någon annanstans än carController (skapa en ny class kanske) // något med factories //factory designmönster och tillämpar SRP igen
- Tillämpa functional decomposition på actionperformed metoden //refaktorisering
- Vi kan köra simuleringen (main()) i en annan klass (application och main i detta fall) //SRP igen

DrawPanel

- Hårdkodat sätt att skilja vilken bild den ska ta //bryter mot OCP
- Läser in bilder från en filväg och ritar grafiken // bred ansvarsområde, dålig SRP
- Hårdkodat sätt att rita grafiken för workshops //bryter mot OCP (igen)

Platform

- Den har ett cirkulärt beroende med Truck eftersom den har en truck som komposition, Ta bort!

Vad kan vi göra?

- Skapa ett interface istället för att ta in en truck i platform. Där bilar med flak implementerar interfacet.

Truck

- Som tidigare nämnt, har ett cirkulärt beroende med plattform.
- Det enda som skiljer Truck och Car är att Truck tar truck tar hänsyn till en plattform i alla metoder som handlar om hastighet.

Vad kan vi göra?
. - Vi tar bort subklassen Truck och skapar ett interface som bilar som har en Platform ska implementera.
Refaktorisering

- Introducerat klasserna graphicsComponent, carFactory, Workshopfactory, Main, Application och graphicsFactory.
- Introducerat interfacerna hasPosition och HasRamp.
- Tagit bort truck och introducerat ett nytt interface hasRamp

graphicsComponent

- Syftet är att simplifiera ritandet av grafik i drawPanel.
- Gör så att drawPanel följer OCP bättre och att ansvarsområdet är mer precist (SRP). Alltså vi behöver inte lägga till nya filvägar när vi har nya bilar eller ändra på koden (där det använder instanceof t.ex)
- Vi kan använda denna graphicsComponent för att “länka” en bil eller en workshop till en bild som har en position, vilket drawPanel kan rita.

CarFactory

- Introducerar ett smidigare och lättare utökningsbart sätt att skapa bilar.

WorkshopFactory

- Introducerar ett smidigare och lättare utökningsbart sätt att skapa workshops.

GraphicsFactory

- Introducerar ett smidigare och lättare utökningsbart sätt att skapa graphicscomponents.

hasPosition

- Interface som graphicsComponent implementerar för att garantera ägaren av ett graphicsComponent har en position.

Main och Application

- Mains syfte är att hålla en application
- Applications syfte är att ta bort ansvar från carController så att simulationen inte körs helt i carController och att controllern inte behöver hantera någon logik av simulationen över huvud taget. Alltså vi flytter Timerlistener classen till application. Då kan vi länka CarView och carController till application istället för att ha ett cirkulärtberonede mellan carview och carController. Vi har tillämpat SRP

Ändringar med Truck och Platform

- Eftersom det var inte många skillnader mellan en Truck och en Car (Truck har en plattform) bestämmer vi att ta bort Truck och låta bilar som har en plattform implementera ett interface som har metoder som berör allt med plattformar. Det blir nog smidigare.

Ändringar med bilar med flak

- Lägg till en try catch exception som gör det mer användarvänligt när flaket orsakar en exception.