DIT954 - Laboration 2, del A

Uppgift 2:

Extends beroendet är nödvändigt för de objekt vi vill instansiera då extends indikerar att det är subklasser av den klassen de ärver. Detta är nödvändigt då dessa klasser ärver properties och metoder som är specifika för just den typen av objekt.

Det går att använda sig av composition istället för inheritance för exempelvis relationen mellan vehicle och car. Istället för att car ärver vehicles properties så kan man skapa ett vehicle objekt som en member i car klassen och delegera funktionaliteten där nödvändigt.

Klassen workshop går att generaliseras mer genom att den “extendar” vehicle istället för car. På detta vis så kan man skapa mer generella verkstäder som kan ta emot vilken typ av fordon som helst.

Beroendet mellan CarController, CarView och DrawPanel är nödvändiga för funktionaliteten av applikationen.
CarController måste ha kännedom av CarView och DrawPanel för att kunna uppdatera fordonens position och utföra repaint metoden för den nya positionen. På samma sätt så interagerar CarView med DrawPanel.

Implementationen kan dock ske på ett annorlunda sätt. Istället för att CarView skapar DrawPanel och kommunicerar det till CarController, så kan DrawPanel skapas separat och direkt skickas till CarView och CarController. CarController skulle kunna referera till både CarView och DrawPanel istället för bara att CarView är ansvarig för skapandet av DrawPanel.

Detta ökar i sin tur programmets modularitet och följer principerna för hög cohesion och låg coupling

Uppgift 3:

Abstrakta Klasser:

AbstractMovable - Har hand om position och rörelse.
Attributerna för klassen är hastighet, y-position, x-position och riktning. Implementerar movable interface som innehåller ett blueprint för tre metoder för rörelse och riktningsförändring.

Dekomposition på klassen skulle kunna göras i avseende till att separera positionering och rörelse. En AbstractPositionable klass skulle kunna implementeras för att bättre följa SRP.

En sådan klass skulle ha hand om all positionering och abstractmovable klassen skulle ha hand om rörelse.

Car - Samlar det som är unikt för ett bil-objekt.
Attribut för klassen är vikt.
Ärver Vehicle.
Liten subklass som i stort sett endast används för separera objekt och göra de mer läsbara. Detta genom subklassning.

HelperMethods - Innehåller eventuella hjälpmetoder som kan komma att behövas i flera klasser.
Innehåller för nuvarande endast en metod som används för mer än en klass.

TrailerTruck - Samlar det som är unikt för ett lastbils-objekt.
Ärver vehicle.
Några av de metoder som klassen implementerar är polymorfa metoder för hastighetsförändring och hastighet variabel.
Andra metoder som klassen implementerar är de som är unika för flak funktionen hos en lastbil. Detta inkluderar även metod som bestämmer om flaket får röra på sig baserat på om flakets ramp är uppe eller nere.

För att bättre följa SoC och SRP skulle en separat trailer klass eventuellt trailer interface som implementeras av de klasser som SKAPAR lastbils objekt där de kan overridas polymorfiskt för de unika egenskaper/funktionalitet som olika flak kan ha.

Vehicle - Innehåller allt som definierar ett fordon.
Attribut för klassen är: antal dörrar, kraft på motor, färg och modell.
Metoder klassen implementerar är getters och setters för attributen samt metoder för att starta/stoppa motorn, gasa/bromsa och öka/minska hastighet. Det finns även en abstrakt metod för hastighetsfaktor som räknas ut för individuella objekt baserat på dess unika egenskaper.

För att bättre följa SoC och SRP skulle klassen kunna delas upp ytterligare då inte alla fordon har en motor eller dörrar. Exempelvis en cykel eller skateboard osv. På detta vis skulle nya fordonstyper lättare kunna implementeras.

Klasser:

CarController - Ansvarar för att både kontrollera och uppdatera bilder som visas grafiskt.

Skulle kunna separeras till två klasser där en klass kontrollerar bilar och en uppdaterar bilden. Även inkonsekvent namngivning.

CarTransporter - Skapar ett biltransport objekt och samlar allt som är unikt för en biltransport.
Ärver TrailerTruck.

Attribut för klassen är max vikt, max antal lastade bilar, nuvarande vikt, last, min-avstånd för att lasta och lasta av bilar och en bool som håller koll på om rampen är uppe.
Metoder som implementeras är de som har hand om släpet: lasta bilar, höja/sänka släp samt det som bestämmer om flaket får röra på sig baserat på status av rampen.

I och med att denna klass ärver TrailerTruck där dekomposition skulle kunna användas för att följa SRP och SoC skulle även denna klass kunna vara mer i linje med SRP. En eventuell flak interface/klass (eller båda) där klassen innehåller specifik funktionalitet och interface ett kontrakt/blueprint för en trailer klass. Då skulle cartransporter använda sig av composition/delegation för att skapa trailer objektet inom cartransporter och där av separera trailer och truck därav göra det mer modulärt och mer i linje med SoC och SRP.

CarLoad skulle även kunna vara en egen klass som hanterar allt med lasten för car transporter.

CarView - Används för att bilda knappar till funktionalitet samt skapa spelfönster.

Kan förbättras genom variabelnamn, mer konsekvent metod för eventlistneners (även samla allt så det blir mer läsbart).

DrawPanel - Används för att binda resurser (bilder) till objekt och även placera ut dem på skärmen.

För att förbättra koden bör variabelnamn skapas, generalisering av metoder för position samt bufferedimages och dela upp koden till mer läsbara snippets.

Saab95 - Skapar ett Saab95 objekt och samlar allt som är unikt för ett sådant.
Ärver Car.
Metoder som implementeras påverkar en overridad hastighetsfaktor som tar hänsyn till turbo.

Liten klass som endast används för dess unika hastighetsfaktor samt att skapa ett saab objekt.

Scania - Skapar ett scania objekt och samlar allt som är unikt för en scania-lastbil.
Ärver TrailerTruck.
Attribut för klassen är vinkel på rampen på släpet.

Även denna klass skulle kunna istället ha en separat trailer klass som antingen är blueprint eller använda composition för att initiera ett trailer objekt inom scania.

Volvo240 - Skapar ett Volvo240 objekt och samlar det som är unikt för en Volvo240 klass.
Ärver Car.
Attribut för klassen är en trimfactor.
Metod för klassen är en hastighetsfaktor som tar hänsyn till dess trim faktor.

Liten klass som endast används för dess unika hastighetsfaktor samt att skapa ett volvo objekt.

Workshop - Skapar ett workshop objekt och samlar funktionalitet för detta.
T även typen car.

Attribut är en deque med valfri typ av bilar. Antigen en generell biltyp eller specifika typer såsom volvo eller saab.

Innehåller metoder för att lagra bilar och hämta ut bilar.

Väl utformad klass som följer designprinciper bra samt är generell nog för enkel utökning.

Enum:
Direction - Innehåller endast enumerations av riktningar (skulle kunna implementeras lokalt i abstract movable) men ännu mer överskådligt på detta vis och gör det också möjligt att lätt lägga till riktningar.

Interface:

Movable - Blueprint för tre metoder som har hand om rörelse samt riktningsförändring.

Uppgift 4:

Abstractmovable - Dela upp till positionable och movable.

TrailerTruck - Gör separat trailer class som med hjälp av composition läggs in i trailertruck alternativt trailer.

Scania - Ändra så att funktionalitet funkar för nya trailer.

CarTransporter - Ändra så att funktionalitet funkar för nya trailer. Separera även carload till en separatklass även den kanske använda sig av compositon?

Vehicle - Dela till motorized vehicle och dela på funktionalitet för motorized och vanligt vehicle.

CarView - Kan förbättras genom variabelnamn, mer konsekvent metod för eventlistneners (även samla allt så det blir mer läsbart).

DrawPanel - För att förbättra koden bör variabelnamn skapas, generalisering av metoder för position samt bufferedimages och dela upp koden till mer läsbara snippets.

CarController - Skulle kunna separeras till två klasser där en klass kontrollerar bilar och en uppdaterar bilden. Även inkonsekvent namngivning.

Trailer klassen måste skapas först för att trailertruck, scania och cartransporter ska kunna refaktoriseras.

Resterande av alla klasser kan refaktoriseras parallellt.

1. Trailer -> TrailerTruck -> Scania -> CarTransporter

   Skapa ny trailer klass och flytta funktionalitet för trailer till den nya klassen. (interface)

   Använd composition för ett släp på TrailerTruck.

   Skapa sedan unika metoder för Scania respektive CarTransporter som uppfyller de krav som sattes. (Interface)

   Många variabelnamn kommer uppdateras.

2. CarLoad -> CarTransporter -> (WORKSHOP)

   Skapa en ny carLoad klass där funktionaliteten för carLoad flyttas.
   Composition för en carLoad i carTransporter. (interace)
   Skapa sedan de unika metoder som behövs.

3. AbstractPositionable -> AbstractMovable

   Skapa en ny AbstractPositionable klass och flytta funktionalitet samt attribut för att positionera ett objekt. AbstractMovable ärver sedan Positionable.

4. Vehicle -> MotorizedVehicles

   Skapa en MotorizedVehicle klass där funktionaliteten för motordrivna fordon sätts. Flytta de resterande funktionaliteterna och metoder som gäller för alla fordon i vehicle klassen. MotorizedVehicle ärver Vehicle klassen.

5. CarModel -> CarController

   Skapa ny CarModel klass och flytta funktionalitet för att uppdatera bilar.

   CarController uppdateras även från timelistener till scheduledexecutorservice som är ett mordernare och lättare alternativ. Bilarna initieras med hjälp av CarModel istället

för att skapa nya separata objekt.

6. VehicleFactory -> CarController
   Skapa en VehicleFactory klass med funktionalitet för att skapa de olika fordonsobjekten.

   CarController kommer att använda sig av dependancy för att få åtkomst till dessa objekt.

7. CarView -> CarController -> DrawPanel

   CarView behöver sammanställa samtliga knappar på ett bättre mer läsbart sätt samt alla listneners skrivas om till (e -> carC.method()) statements.

8. DrawPanel -> CarController

   Drawpanel kommer metoder för att binda resurser göras om till mer generella (abstraheras) sådana. Även nya variabelnamn kommer sättas.

9. Workshop -> Vehicle
   Workshop type parameter extends vehicle istället för car.

(fixa en car model typ klass)