Seminar 2

Object-Oriented Design, IV1350

Maher Jabbar, maherj@kth.se

2016-04-20

# Contents

# 1 Introduktion

uppgiften för detta labb är att designa program för medföljande program specifikation.
Vi är tilldelade för presentera vår lösning i Klassdiagram och kommunikations- eller sekvensdiagram som bygger på UML språk. Programmet måste vara utformad i enlighet med MVC Pattern. Designen bedöms på om design är fast med hög sammanhållning, låg koppling och god inkapsling.

# 2 Metod

Jag använade mig av programmet Asta för designa min Klassdiagram och SystemSequenseDiagram.
Jag har använt kommunikations istället för SystemSequenseDiagram diagram för detta labb för att designa system-verksamheten i programmet och En klassDiagram för att visa sambanden mellan olika skikt. System verksamheten är allt från SSD och de flesta klasserna är från domän model. Jag har försökt att hålla så låg koppling och hög sammanhållning som möjligt för klasserna, Attribut och metoder visas i klass diagrammet.

# 3 Resultat

Figure 3.1 diagram visar när en ny kund kommer. View informerar Controller att ny inspection kan startas och sen i sin tur informerar Garage. Garaget får då det nummer som ska presenteras utifrån kön och passerar det numret till Display och därefter öppnar GarageDoor.

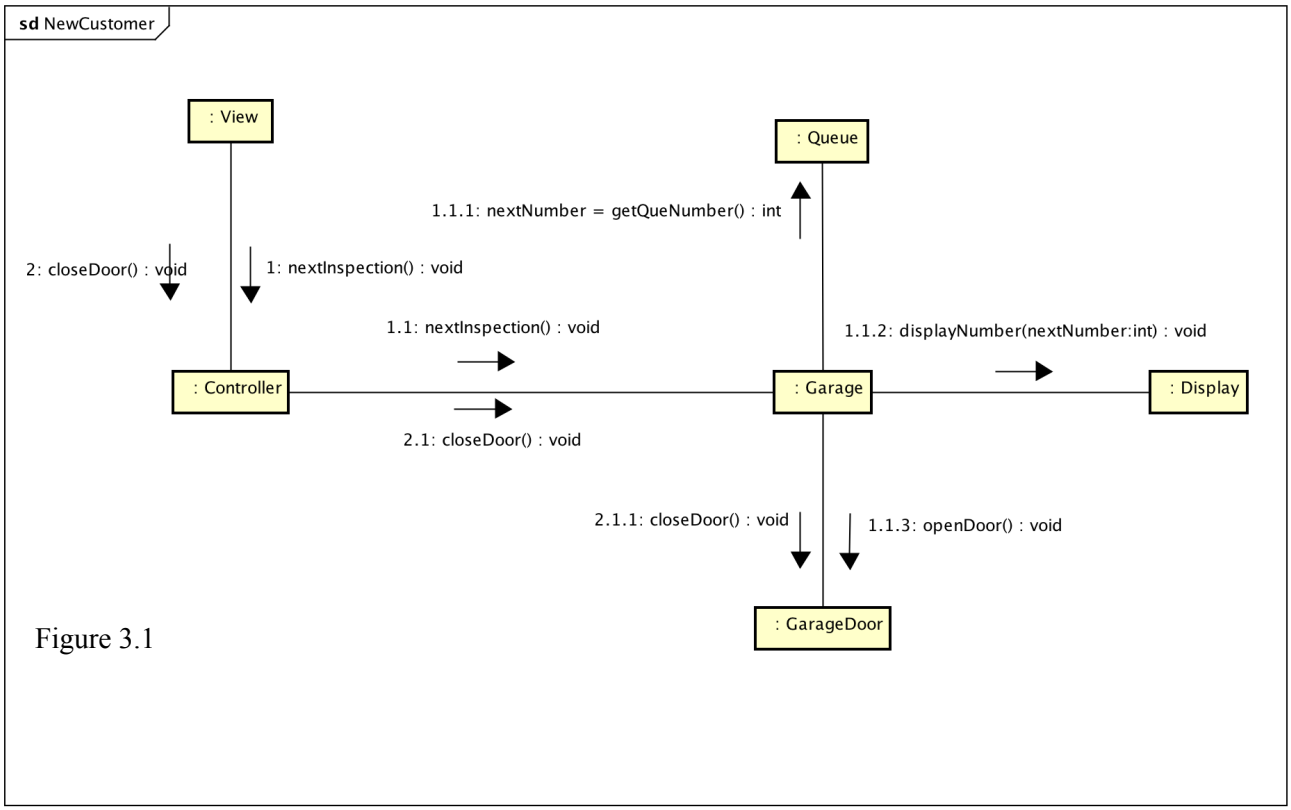
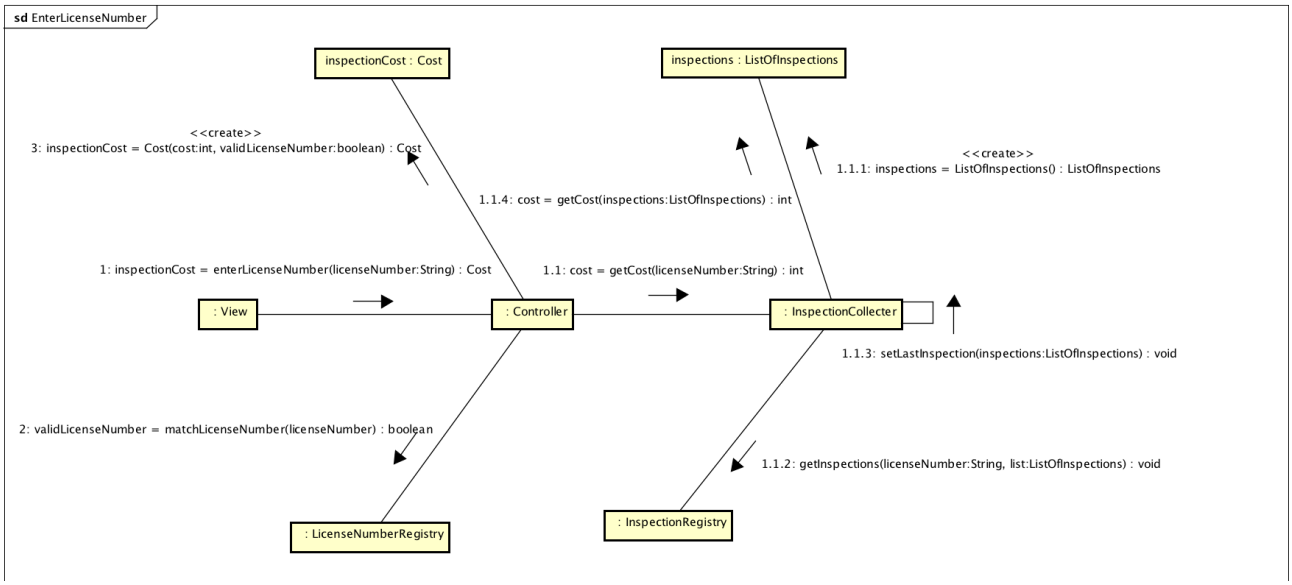



Figure 3.1

Figur 3.2 visar ett diagram när en inspektör skriver in licensnummer för ett fordon och får kostnaden för inspektionen. Den första sekvensen startar när View skriv in licensnumret och skickar den till Controll vilken i sin tur skickar den till InspectionCollecter. InspectionCollecter anropar sedan ListOfInspections och gör en lista över inspektioner. InspectionCollecter anropar sedan InspectionRegistry med listan fylld med inspektioner som krävs för fordonet med licensnumret anges. Metoden setLastInspection vilken i sin tur används sedan av InspectionCollecter är för inställning av boolesk variabel lastInspection att varje inspektion måste den senaste inspektionen i listan kommer vilken i sin tur krävas senare när inspektionerna utförs.




Diagrammet 3,3 visar hur auktorisationen för ett kreditkort fått och hur kvitto skrivs ut. Vid första vyn kallar kontroller för att få ett tillstånd med argumenten licenseNumber, inspectionCost och kreditkortsinformation. Kontrollen anropar sedan InspectionCollecter för att få en lista över inspektioner för kontroll som kommer att  användas för att skapa ett kvitto och skapar ett objekt Kreditkorts Vilket innehåller kreditkort information.

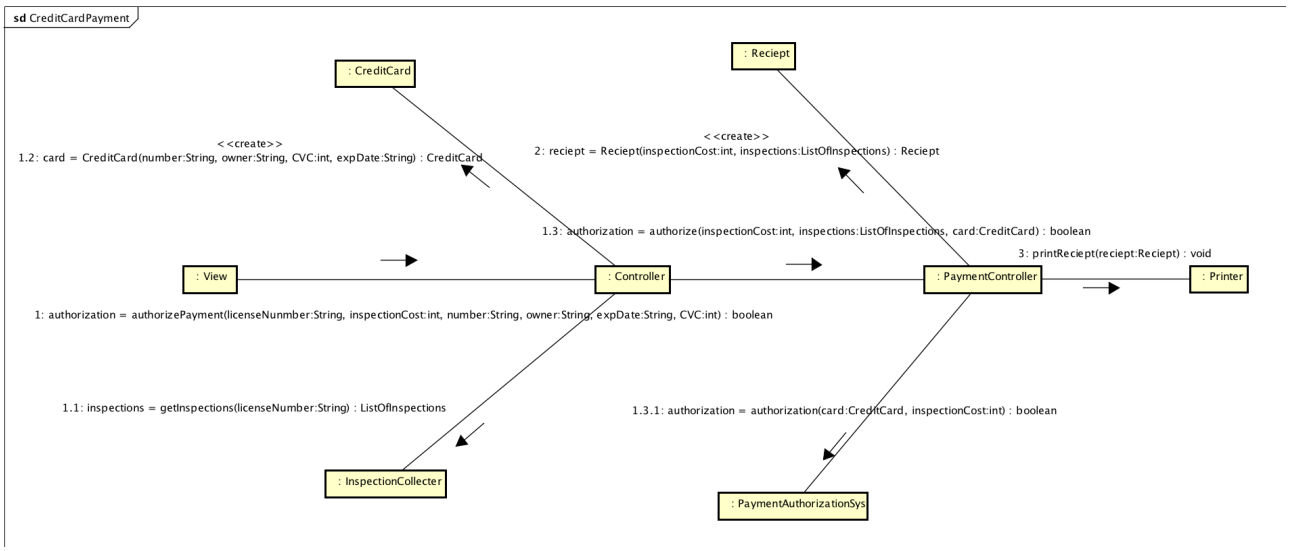
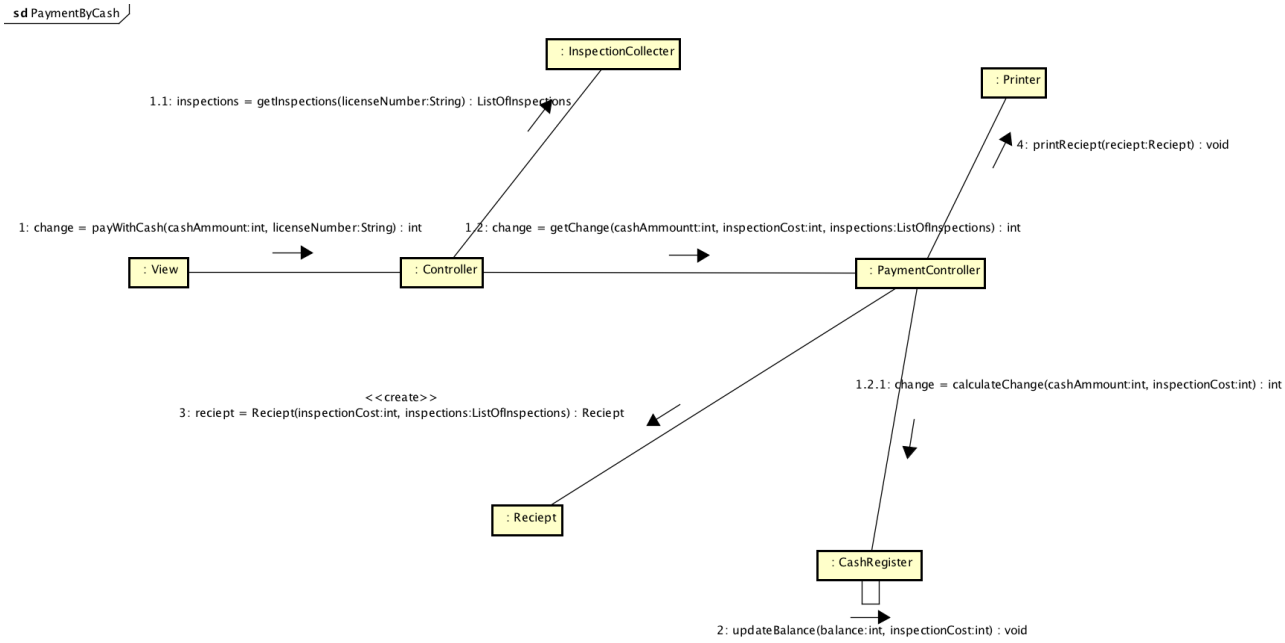
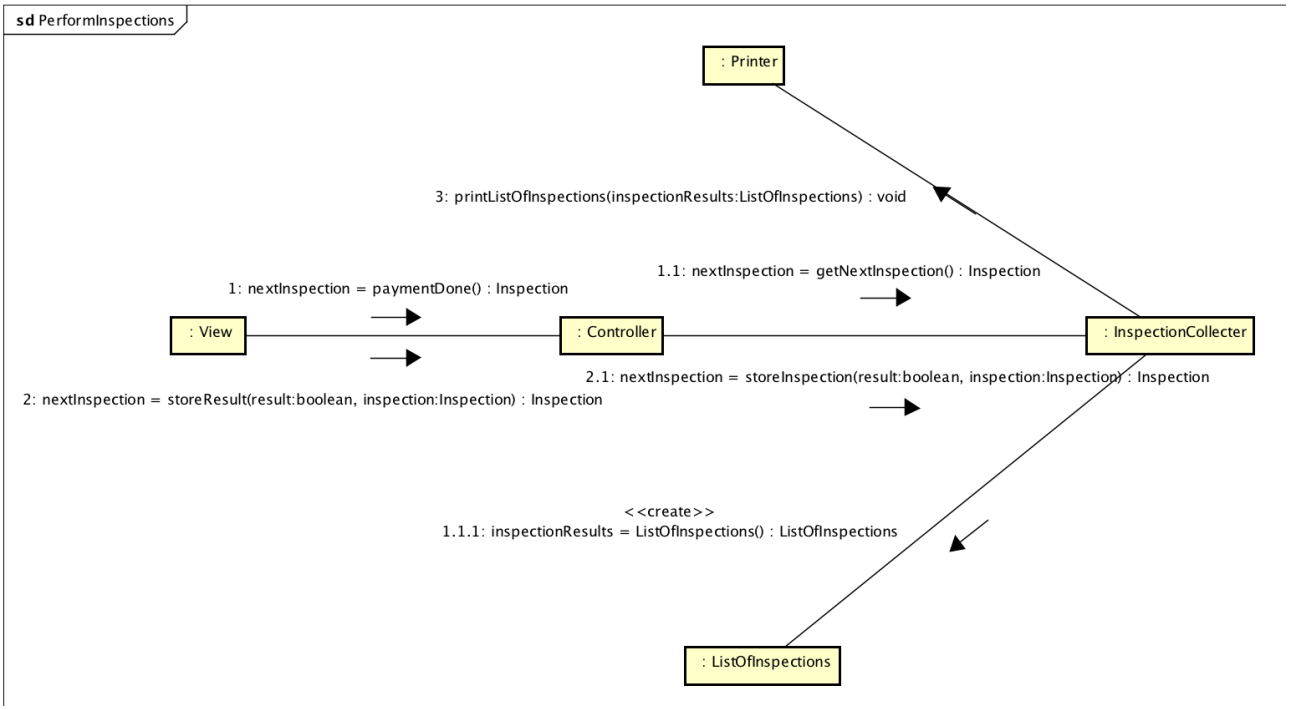


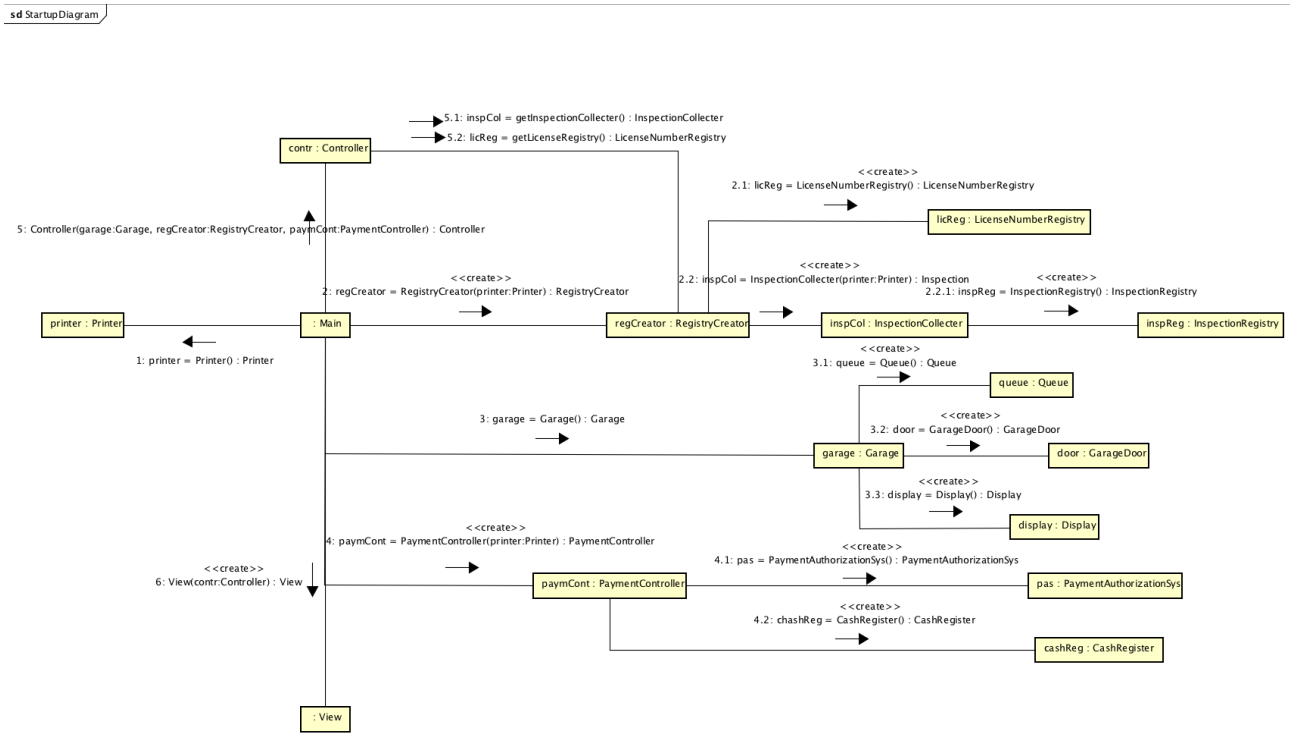
figure 3,4 visar hur betalning med kontanter sker och hur ett kvitto skrivs ut.



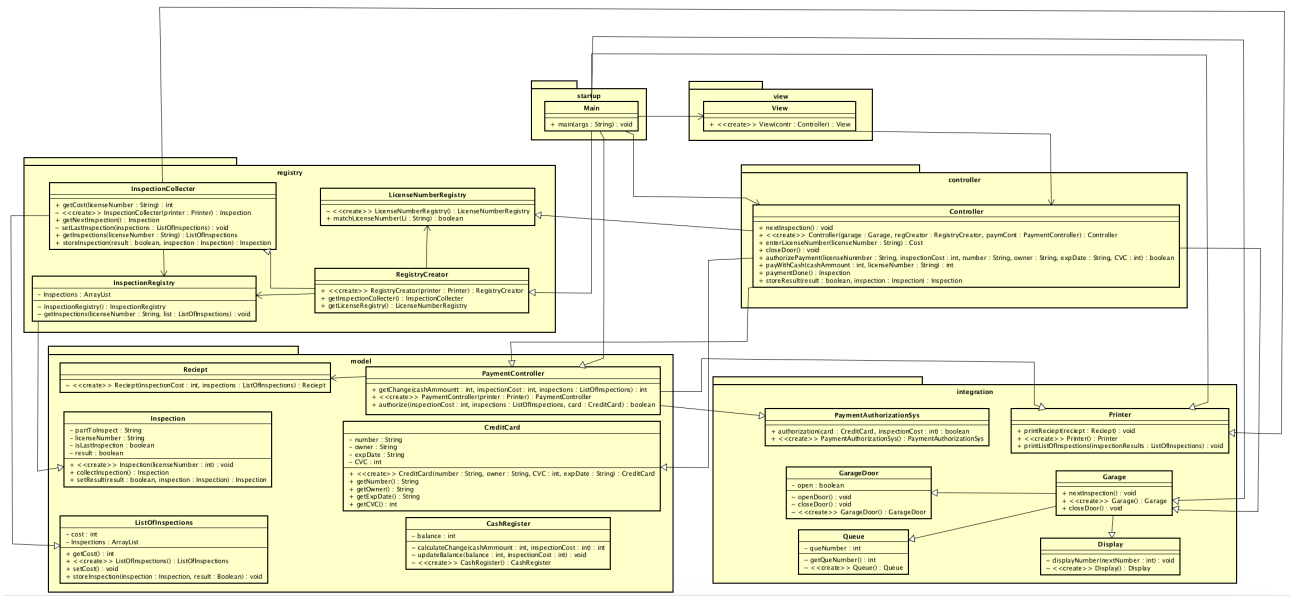

Figur 3.5 visar hur en inspektion görs. För att få den första inspektionen View anropar kontroller och meddelar att en betalning har gjorts, kontroller kallar sedan InspectionCollector att få en inspektion. InspectionCollector gör sedan en ListOfInspections som kommer att sändas för att lagra inspektionsresultaten . När en inspektion har gjort inspektion och resultatet skickas till InspectionCollector genom kontroller.



Figur 3.6 visar startup. Main börjar skapa skrivare. Main skapar sedan RegistryCreator och ger en referens till skrivaren. RegistryCreator sedan createsLicenseNumberRegistry och InspectionCollector och ger InspectionCollector en hänvisning till skrivaren. InspectionCollector skapar sedan InspectionRegistry. Nästa steg är att Main  skapa garage och sedan kö, Display och GarageDoor. Efter det Main skapar PaymentController och ger en referens till skrivaren.




Figur 3.7 visar alla lager klasser och referenser.



# 4 Discussion

Detta program skulle kunna utformas i ett oändligt antal olika sätt. Jag tror att det viktigaste för att kunna skapa en bra design är att följa den metod som grundligt och arbeta iterativa. Jag tycker att min kommunikation diagram beskriver alla nödvändiga händelser och är förståeligt. Klassdiagram är jag nöjd med, Jag tycker att man ska inte ha så många klasser i en package då blir det rörigt med många föreningar med olika skikten. Controller har referenser över hela diagrammet och bidrar till hög koppling. Klassen uppgift är att få programmet att gå och därför behöver skapa några klasser och distribuera referenser och alla händelser passerar genom kontroller på en gång så, Jag försökte alltid att ha god inkapsling, Display,Classes,Garage har alla deras metoder paket skyddade vilket bidrar till god inkapsling.