Labb 2

Objektorienterad Design, IV1350

Evan Saboo
saboo@kth.se
2015-06-03

Innehåll

1 Introduktion 3

2 Metod 4

3 Resultat 5

1 Introduktion

I den andra labben skulle man fortsätta designa klass och sekvens diagram för bilprovningsprogrammet från det första labbet. Detta skulle utföras genom att följas MVC mönster genom koppla klass och sekvens diagram tillsammans. Designen skulle också ha hög sammanhållning, låg koppling och bra inkapsling med ett väldesignad publikt gränssnitt.

2 Metod

I den andra labben var det viktigt att förstå hur MVC mönstret fungerade och vad vy, controller och modell hade för egenskaper. För att följa MVC mönstret behövdes 3 paket och 3 klasser vilket var **view**, **controller** och **model**. Jag började med att skapa alla nödvändiga klasser och tilldelade dem till sina rätta paket i klass diagrammet. Flera klasser skapades för att kunna ändra delar av programmet utan att påverka hela programmet, då klasserna gör sina egna uppgifter. Klasserna utför inte några extra uppgifter t.ex. klassen Receipt används bara för att skriva ut den slutförda inspektionsbetalningen, vilket leder till hög sammanhållning.

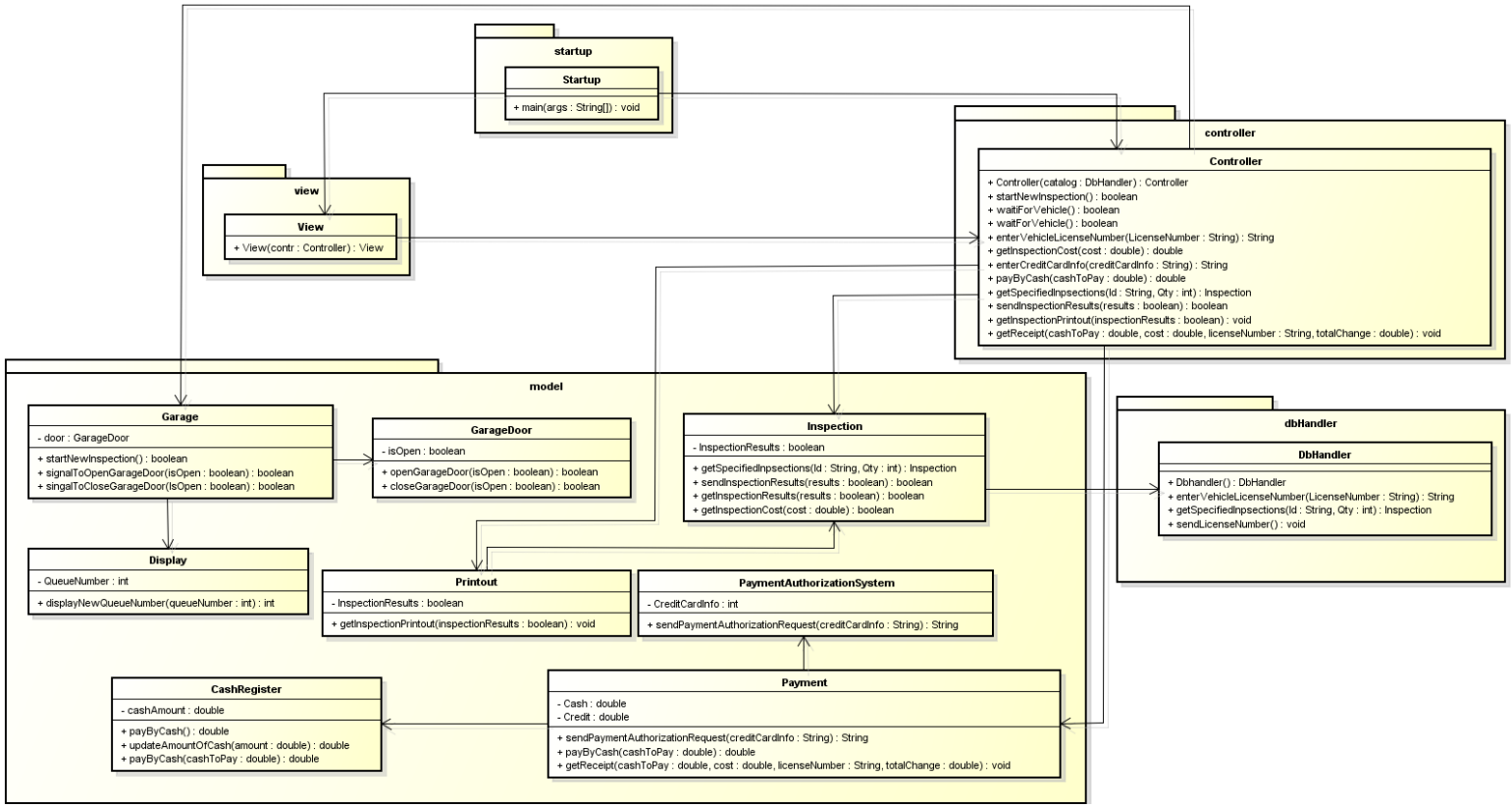
För att tydligare visa klassernas koppling drog jag pilar från klasser till andra klasser istället för från paket till annat paket, eftersom några metoder i ett lager inte hade någon koppling t.ex. från paketet controller till model.

Man kunde välja mellan kommunikationsdiagram eller sekvensdiagram till sin design, jag valde sekvensdiagram eftersom vi hade jobbat med det i förra labbet och jag visste mer om hur man utförde diagrammet. Jag tilldelade alla operationer till flera sekvensdiagram t.ex. metoden openGarageDoor och closeGarageDoor visas upp i varsin sekvensdiagram för att ha så låg koppling som möjligt. För att undvika irrelevanta kopplingar i klassdiagrammet kopplades bara relevanta klasser med varandra, t.ex. klassen Payment kopplades med klassen CashRegister och PaymentAuthorizationSystem men dessa klasser hade ingen koppling mellan varandra, därför utfördes ingen koppling mellan dem.

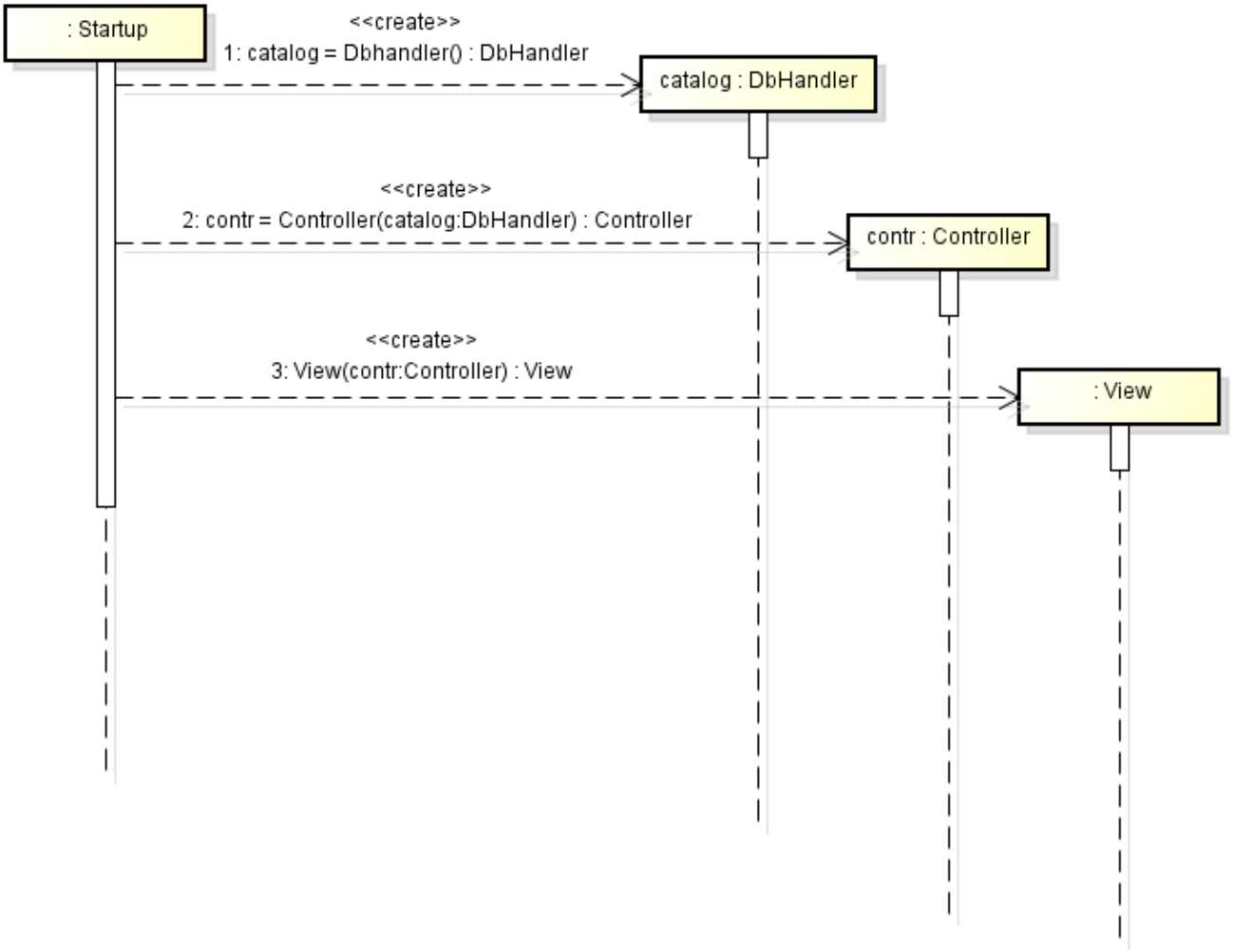
I designen skapades privata attribut vilket inte används i andra klasser än sin egen klass för designen ska ha bra inkapsling.

Jag utförde designen själv men jag jobbade också med Emil Nordin för att utföra designen på ett korrekt sätt.

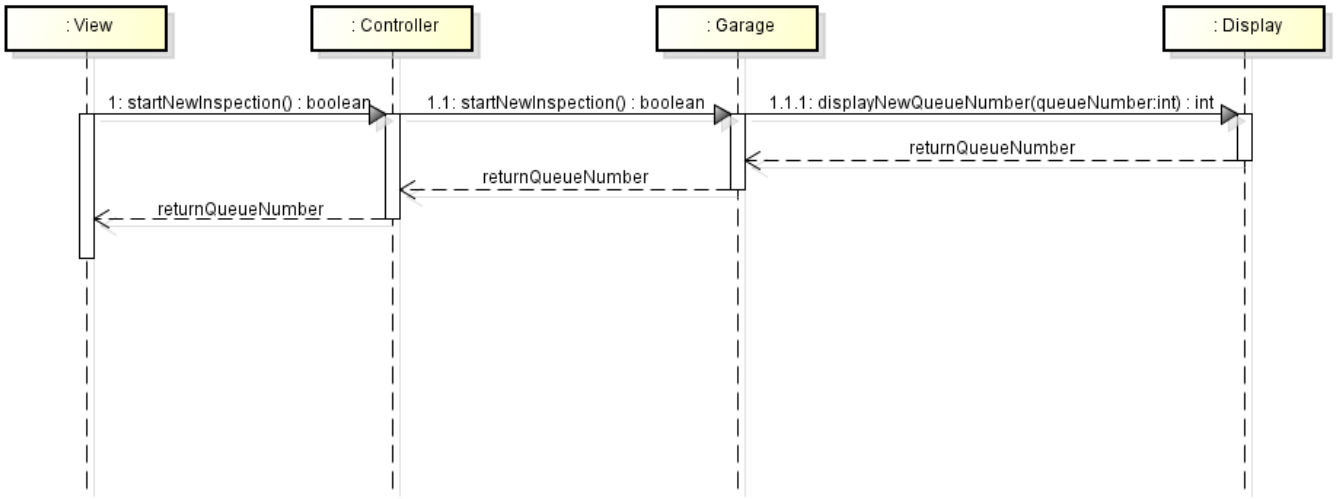
3 Resultat



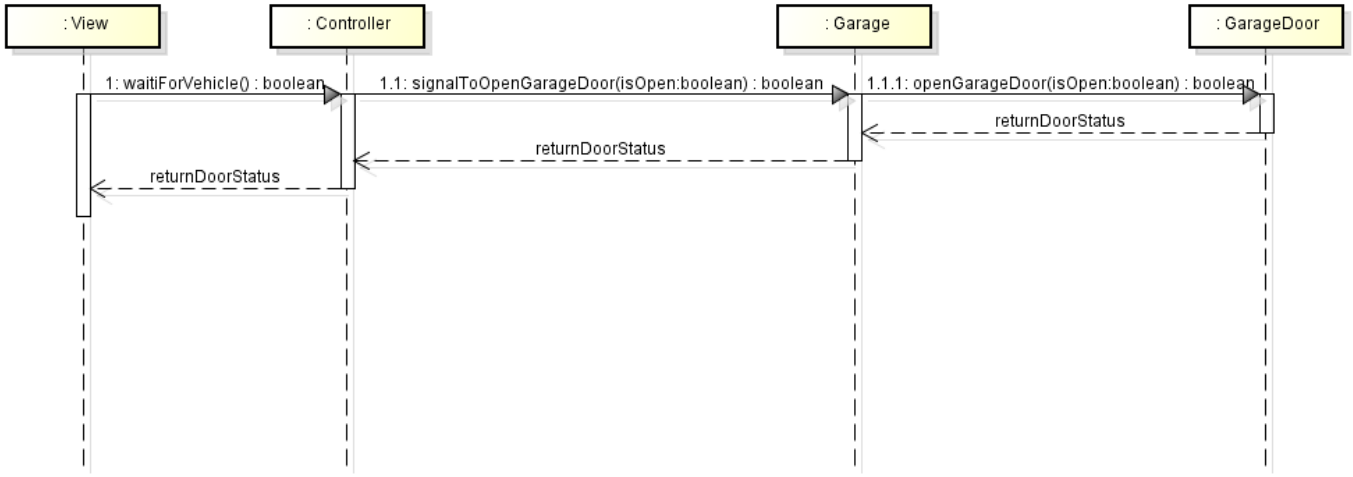
Figur 1: Ett klass diagram som följer MVC mönster.



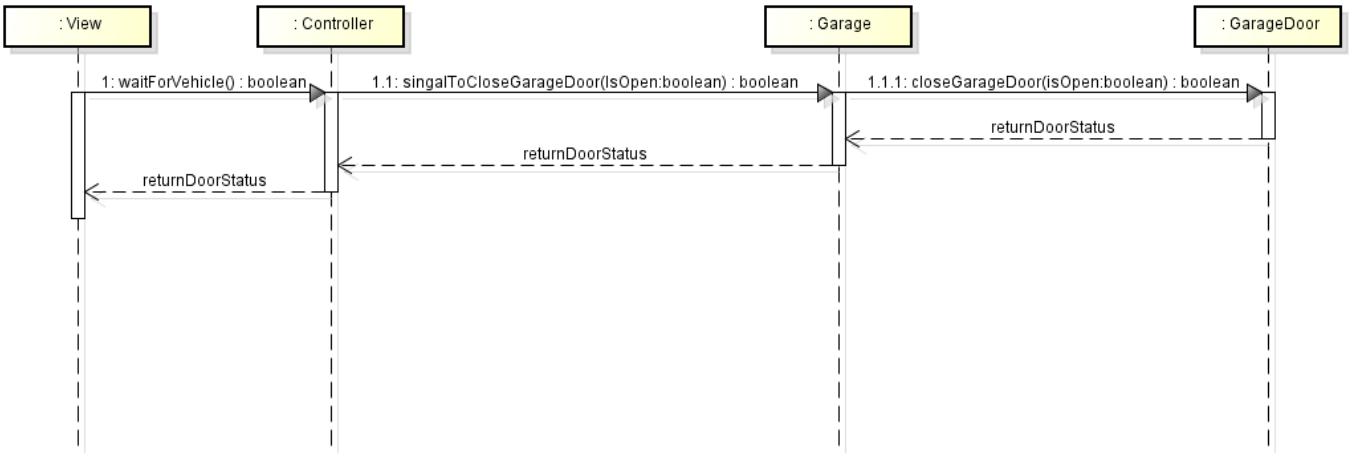
Figur 2: Et sekvensdiagram på Startup.



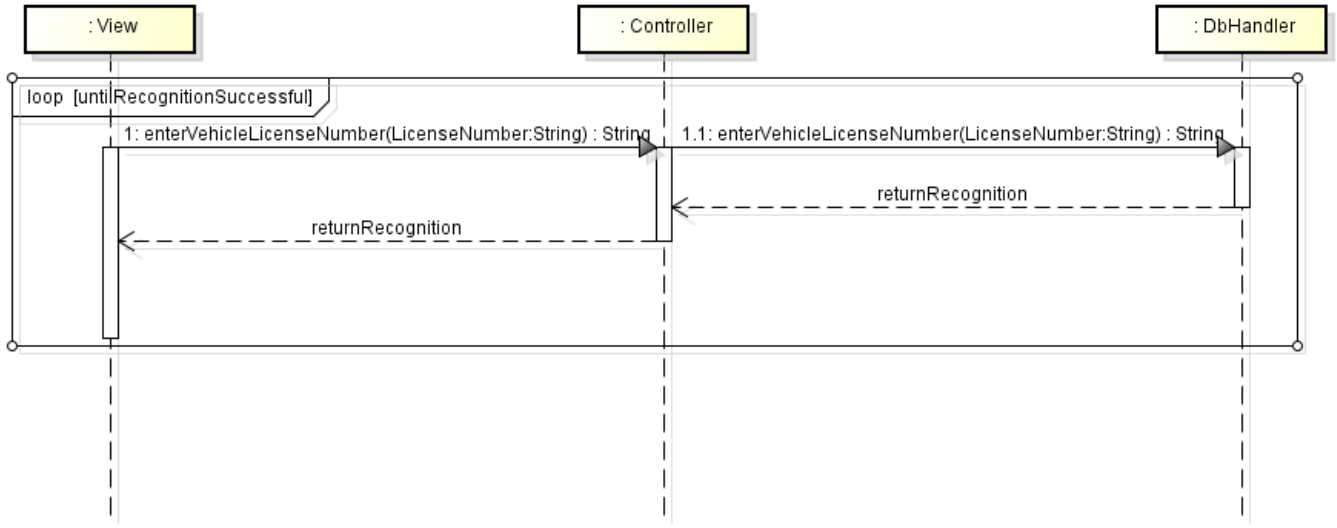
Figur 3: Ett sekvensdiagram för att starta inspektion och visa nytt könummer.



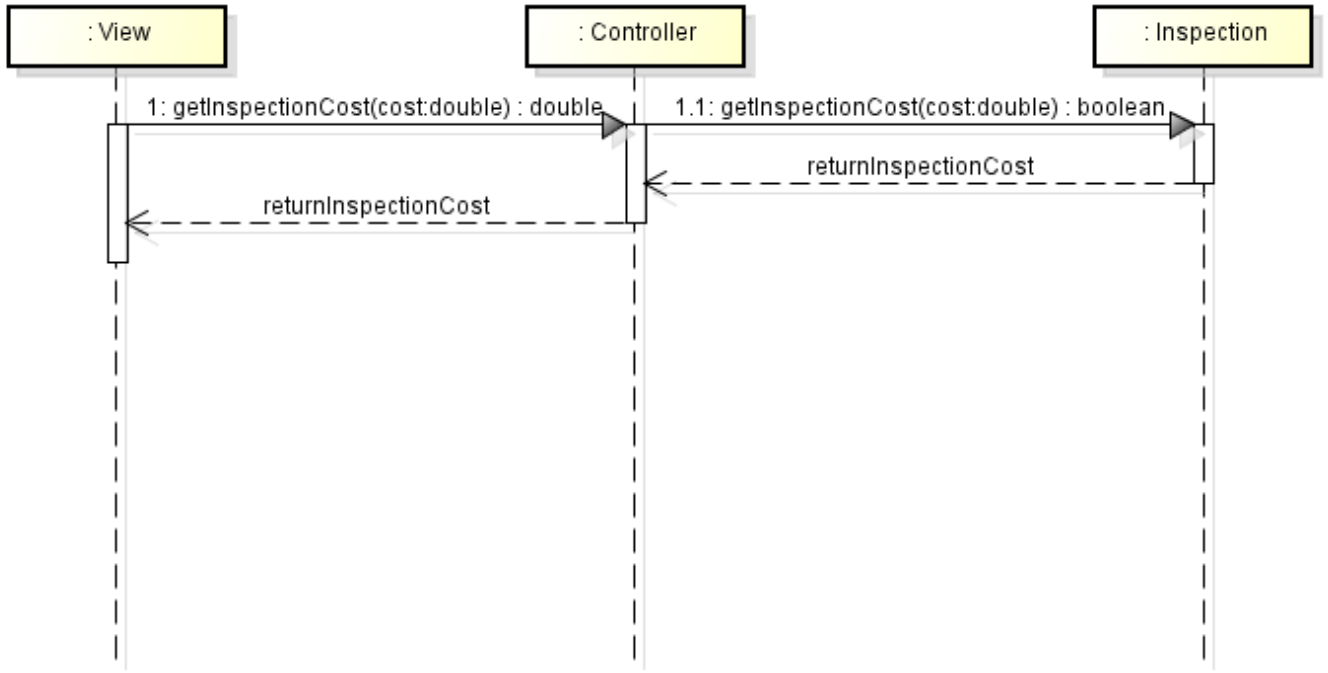
Figur 4: Ett sekvensdiagram för att öppna garageporten.



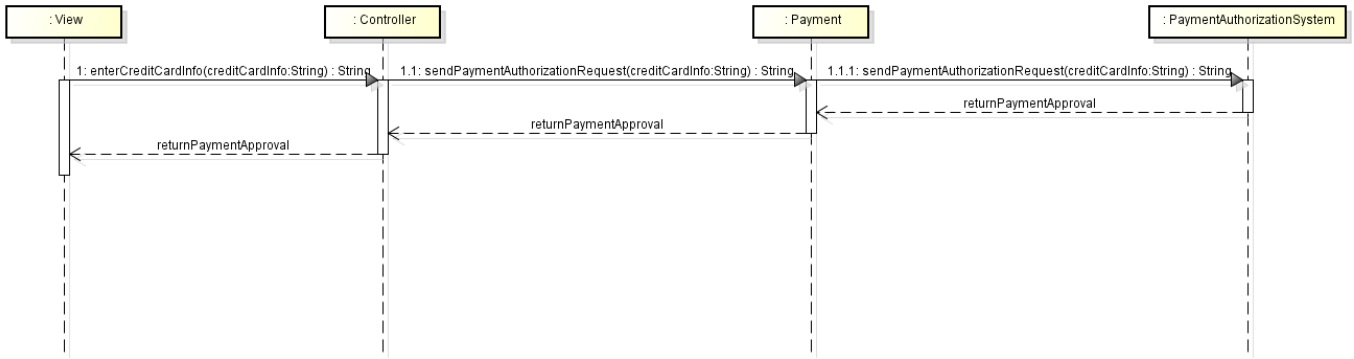
Figur 5: Ett sekvensdiagram för att stänga garageporten.



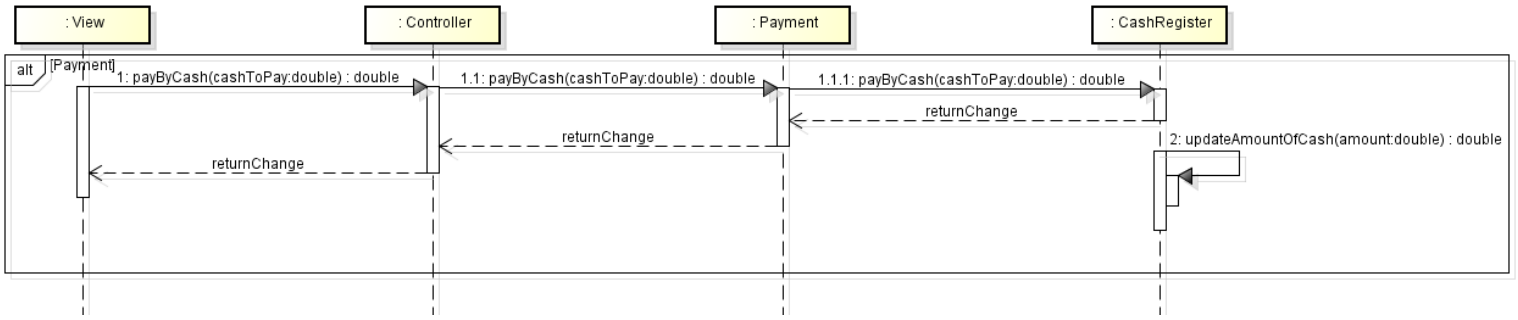
Figur 6: Ett sekvensdiagram för validering av licensnummer.



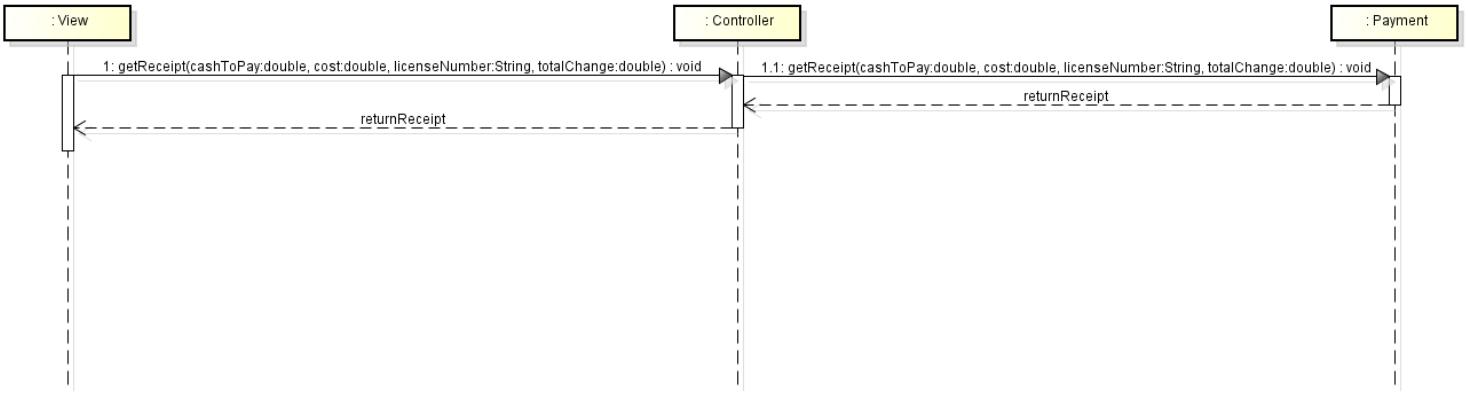
Figur 7: Ett sekvensdiagram för att få kostnaden av inspektionen.



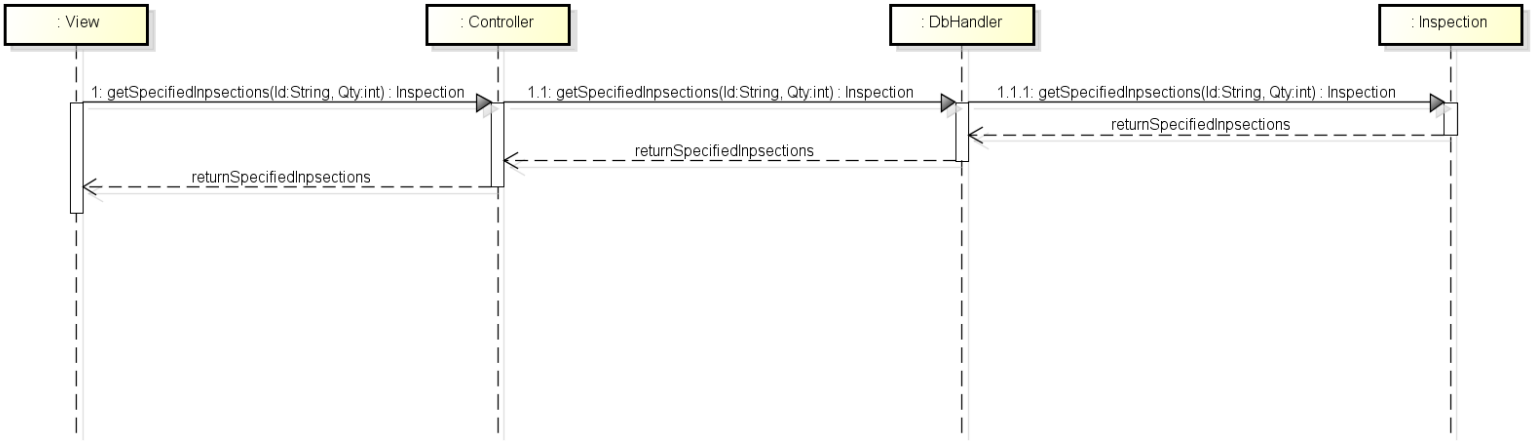
Figur 8: Ett sekvensdiagram för betalning med kreditkort.



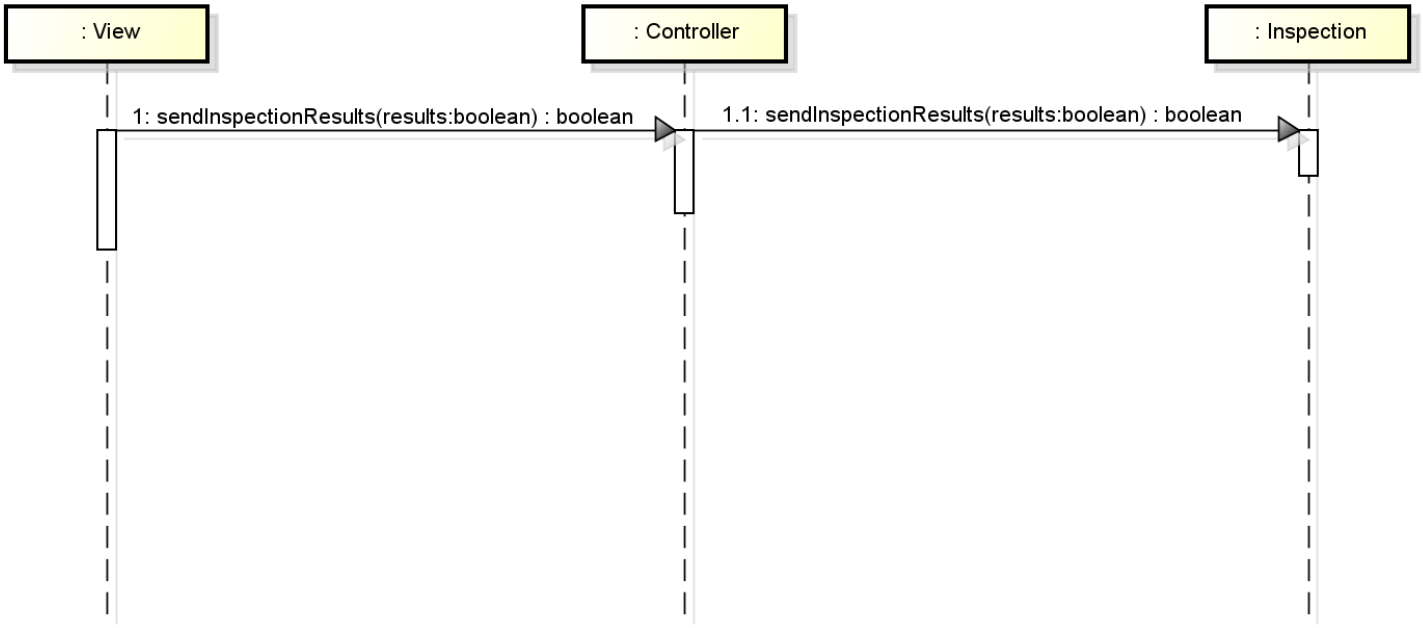
Figur 9: Ett sekvensdiagram för alternativ betalning med kontanter.



Figur 10: Sekvensdiagram för mottagning av kvitto.



Figur 11: Sekvensdiagram för att få specificerade inspektioner.



Figur 12: Ett sekvensdiagram för att skicka tillbaka inspektionsresultaten.

Figur 13: Ett sekvensdiagram för att få utskriften av inspektionsresultaten.