Problem innan:

Controller ska vara tunnare, ska inte ha main metoden eller modellen i sig. Uppdateringsdelen ska heller inte ligga i controller.

Finns ingen modell, så ska teknisk sätt vara smartare skapa en modell

View var dum nog, fast man skulle kunna flytta ut knapparna till en controller

Ingen Applikation, applikationen kan ta hand om fönstret och bakgrunden

Problem och fixar nu:

Det finns en modell nu, så modellen är smartare. Applikationen har nu uppdateringsmetoden och main metoden (uppdateringsmetoden bör vara i modellen).
Knapparna är fortfarande i view också.
Applikationen har fortfarande ingenting med bakgrund och rutan att göra

3.

Hur bör MVC:n vara
Lägga till en lista med panels (eller views) samt typ en lista för allt data som kan uppdateras.
View har en modell är då behövligt, (samt lägga till en observer pattern så att man kan uppdatera view).

4.

Factory method:

- Vi använder en factory method p.g.a att vi tänkte i framtiden kanske man vill ändra views, samt som att det skapar mindre coupling.
- Vi använder redan factory method pattern på de lämpliga ställen, så nej, det behövs inget extra

Observer pattern:

- Vi använder för tillfället en observer för att view inte skulle vara direkt kopplad till controllern. Vi använder observern för att berätta för controllern att en knapp har blivit tryckt på.
- I framtiden så bör modellen ha all uppdateringslogik, detta innebär för att view ska vara uppdaterad, och modellen inte ska ha en view, så måste man ha en observer mellan dessa som uppdaterar view.

State pattern:

- Vi använder ej en state pattern.
- Vi använder inga states på det sättet som en state pattern skulle vara lämpligt

Composite pattern:

- Vi använder ej composite pattern
- Man skulle kunna ha alla bilar i en composite pattern istället för det vi har nu (vilket är en carworld som gör ungefär samma sak) ifall man istället vill ha en ”world” där fler objekt än bara bilar kan finnas. Är ej nödvändig då vi bara gör globala kommandon