Uppgift 2: Model-View-Controller

Avvikelser från MVC-idealet

- I det ursprungliga användargränssnittet (del A) finns några avvikelser från MVC-idealet till exempel **carController** direkt hanterade inom **CarViewklassen**, vilket bryter mot principen om separation mellan **view** och **controller**.
- I **drawPanelklassen** finns direkt åtkomst till **carController**, vilket ökar kopplingen mellan view och controller so ska det inte vara.

Vilka av dessa brister åtgärdade ni med er nya design från del 2A?

- I 2A har vi skappat CarControllerklassen för att separera kontrolleransvaret från CarView, vilket följer MVC-idealet.
- Vi har också introdusera CarButtonListeners genom att skapa en egen klass för att hantera knapptryckningar separeras användargränssnittslogiken tydligare från både view och controller.
- Åtgärder som gas, broms, etc., hanteras nu av CarController, vilket skapar en bättre uppdelning av ansvar mellan view och controller vilket gör en bättre hantering av actions.

Kvarstående brister:

- Fordonsklasserna **Volvo240**, **Saab95**, **Scania**, etc. fungerar som modeller, men det finns ingen specifik modellklass. Införandet av en separat modellklass kan ytterligare förtydliga ansvar och underlätta hantering av tillståndsinformation.

Uppgift 3: Fler designmonster

Observer Pattern:
Vi har använd Observer Pattern i klassen **TimerListener** som observerar Timer events och uppdaterar cars i **CarController**.

Factory Method :
Vi har använd Factory Method i klassen **CarAssembler** och det är en bra sätt att montera olika typer av bilar.

State Pattern:
Vi har använd State Pattern i **Vehicle** med metoder ***startEngine***, ***stopEngine***, ***gas*** och ***broms***, för att ett objekt kan ändra sitt beteende när dess interna tillstånd ändras.
Composite Pattern:
Vi använde också Composite Pattern i **CarController** i ***gas***, ***brake***, ... metoder, där vi repeterar varje metod på samma sätt i hela lista.

Uppgift 5: Utöka användargränssnittet

Vi kan använda “Command Design Pattern” genom att:

- Att skapa Command Interface
- Att skapa AddCarCommand klass
- Att skapa RemoveCarCommand klass
- Uppdaterar till sist Car Button Listeners