Mini-projekt 5

Stefan Larsson Alexander Eriksson

28 februari 2016

Innehåll

1 Inledning och syfte

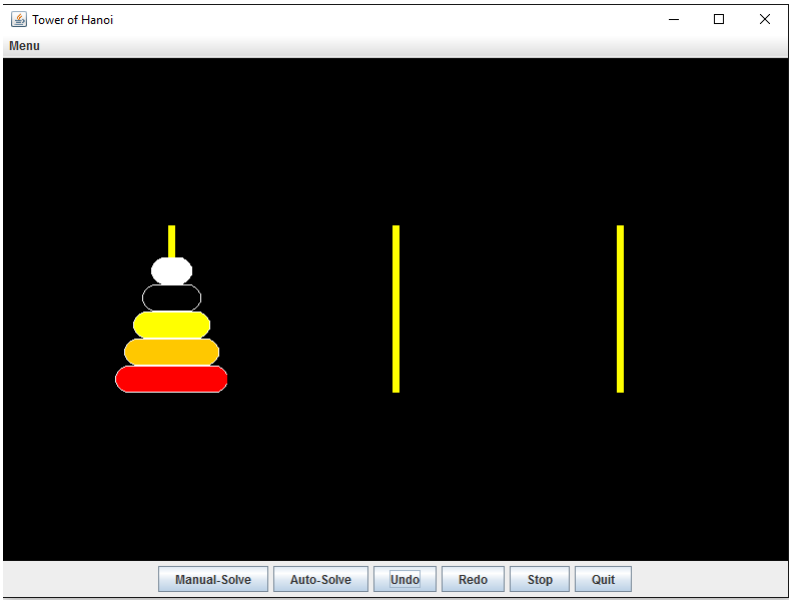
Detta mini-projekt går ut på att bygga om en existerande applikation till en mer interaktiv version. Applikationen, Tower of Hanoi, löser endast spelet med hjälp av en algoritm. Tanken är att man ska kunna flytta brickorna själv och lösa spelet. Syftet med uppgiften är att använda sig av designmönster som `Command`, `Mediator` och `Observer` för att lösa detta problem.

2 Metod

I uppgiften ska `Eclipse` användas och den existerande koden som har tilldelats projektgruppen skall utökas för manuellt användande av applikationen. Den kod som presenterats ska kollas igenom och undersökas för att se vilka åtgärder som behöver vidtas för att sedan applicera tidigare nämnda designmönster.

3 Resultat

I applikationen så har det lags till ytterligare en separat panel för manuellt användande där det skapas tre egna pinnar för att kunna ha två olika spelplaner. Man kan till exempel välja att köra den automatiska lösningen och sedan gå tillbaka till den manuella körningen och fortsätta. Det finns möjlighet att ångra drag och att göra om ett ångrat drag. Detta fungerar med `Command` genom att spara ett drag som ett `MoveCommand`. I ett `MoveCommand` sparas förflyttningen och förflyttningen tillbaka. Dessa `Command`s sparas med hjälp av en `CommandManager` som hanterar `undo()` och `redo()`-funktionerna.

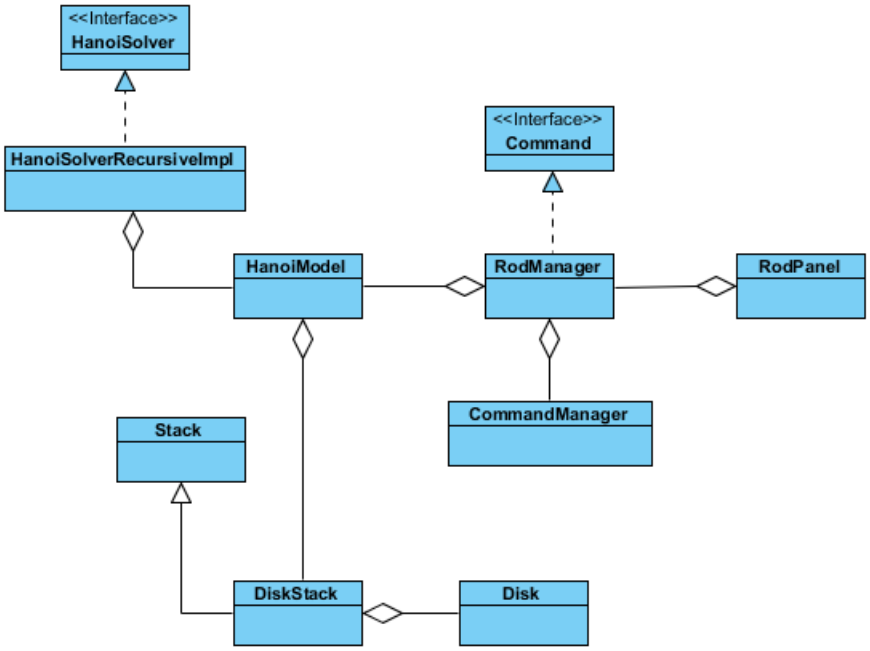


Figur 1: Bild på applikationen.

Ovan i Figur 1, så kan man se knapparna som lagts till för den nya funktionaliteten - Manual-Solve, Undo och Redo.

`Mediator`-mönstret har använts och kallas för `RodManager` i detta projekt. Den sköter kommunikationen mellan modellen och det grafiska gränssnittet. `RodPanel` observerar i sin tur `DiskStack` och blir då notifierad när en förändring sker i modellen.

I figur 2 nedan så kan man se ett UML-diagram över strukturen i systemet.



Figur 2: Klassdigram över systemet.

4 Diskussion

Det finns lite buggar i funktionaliteten i spelet som inte riktigt hunnits med att åtgärdats. Den stora buggen är att man kan lägga en större bricka på en mindre bricka vilket bryter mot i princip den viktigaste spelregeln i själva spelet. I koden som vi fick fanns det ingen hantering av detta då endast en algoritm kördes utifrån hur brickorna låg. Detta hade varit orelevant för uppgiftens syfte att lösa, då fokus låg på designmönster. Det hade dock varit en rolig utmaning men tiden räckte inte till.

En annan bugg finns som triggas när man klickar på en tom pinne och försöker förflytta till en annan. Där kraschar programmet för att den försöker hämta en färg från en disk som inte finns. Detta problem ligger i tidigare kod som vi inte kände oss manade till att lösa.

5 Slutsats

Det var lärorikt för oss att sätta oss in i existerande kod och implementera nya funktioner. Det är ofta en svår uppgift men en nyttig erfarenhet. Det var roligt att applicera designmönstren på en modell som redan byggts. Tack vare att vi använde oss av designmönstren så blev uppgiften ganska hanterbar och rakt på sak.