Mittuniversitetet
MID SWEDEN UNIVERSITY

Lab 5 – **Kommando över "Towers of Hanoi"**

Designmönster med C++

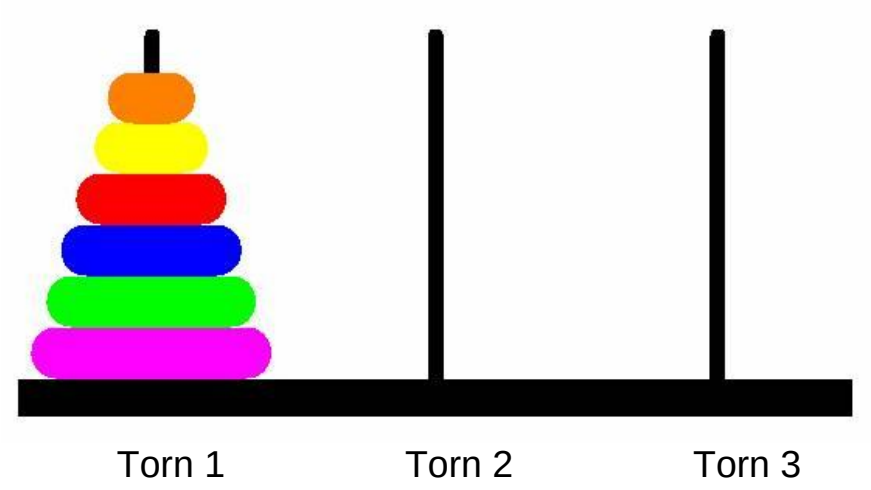
Syfte: Tillämpning av DP Command.

Lab 5 - Kommando över ”Towers of Hanoi”

The Towers of Hanoi är ett klassiskt problem (som också har en klassisk rekursiv lösning). Vi har ett antal skivor med olika diameter som kan flyttas mellan tre pinnar, de tre tornen, numrerade 1 till 3 från vänster till höger. Problemet är att flytta samtliga skivor från torn 1 till torn 2 med hjälp av torn 3.

Regler:

- Endast en skiva får flyttas åt gången från ett torn till ett annat.
- En större skiva får aldrig ligga ovanpå en mindre.
- I utgångsläget finns alla skivorna i torn 1 och i slutet ska alla finnas i torn 2.



Uppgift

Uppgiften i den här labben är att försöka lösa problemet genom att flytta skivorna ’manuellt’ och att använda DP Command för att implementera ’Undo/Redo’ och ’Replay’.
Till uppgiften finns ’spelmotorn’ i form av klassen HanoiEngine given. HanoiEngine lagrar aktuell status för spelet och har funktioner för att flytta en skiva i taget och för att visa en textbaserad bild av de tre tornen med sina skivor. Antalet skivor i spelet sätts i konstruktorn. Skivorna numreras 1 .. n där skiva 1 är den minsta och skiva nr n är den största.

```
class HanoiEngine {
public:
      HanoiEngine(int aLevels = 5); // aLevels is the nr of discs
      ~HanoiEngine() { }

      // Display the towers with the discs
      void show(ostream &aStrm = cout);

      // Move a disc from tower aFrom to tower aTo.
      // A successful move returns true
      bool move(int aFrom, int aTo);

      // Reset the game, start with aDiscs discs
      void reset(int aDiscs);

      // Is the game successfully finished?
      bool isDone();

private:
      enum { T1, T2, T3 };  // Index in iTowers
      int iDiscs;
      int iMoves;
      deque<int> iTowers[3];
};
```

Specifikationer

Programmet ska vara menystyrt med ett textbaserat gränssnitt. DP Command ska användas för att koppla isär gjorda menyval (command issuer) från HanoiEngine (command receiver) som utför kommandona. En mellanliggande CommandManager (command invoker) ska användas.
’Replay’ av det senaste spelet ska möjliggöras genom att kommandon loggas till fil så att hela förloppet senare kan återskapas genom återexekvering av kommandona. Loggfilens namn ska vara Hanoi.log.

Operationer:

- Replay - vid start av programmet ska möjlighet till replay av det senaste spelet ges (om en loggfil finns).
- Move - flytta en skiva från ett torn till ett annat
- ’Undo’ av gjorda flyttningar
- ’Redo’ av gjorda ’undo’
- Reset - återställning till startläge, antalet skivor ska anges.
- Show – visa upp aktuell ställning.

Vid start ska alltså användaren få chansen att se ett replay på senaste spelet om det är möjligt. För varje nytt spel ska användaren ange antalet skivor.

Inga publika metoder (utom möjligtvis konstruktorn) i HanoiEngine får anropas direkt från menykoden.

Replay

Informationen i log-filen ska vara sådan att hela sekvensen av gjorda flyttningar från senaste spelet kan återskapas genom att motsvarande kommando-objekt återskapas. Replay behöver bara kunna utföras direkt efter programstart. De kommandon som utförs i samband med detta ska inte loggas, Hanoi.log ska bara läsas.

För att bl.a. kunna lösa replay-delen på ett bra sätt är det lämpligt med följande Command-klass:

```
class Command {
public:
      virtual bool execute() = 0;  // Execute the command
      virtual bool unExecute() = 0; // Undo the command
      virtual bool isUndoable() = 0; // Undoable?
      virtual void readFromStream(ifstream&) = 0; //Read from stream
      virtual ~Command() { }
};
```

Varje konkret Command-klass har de datamedlemmar som är nödvändiga. Det kan t.ex. vara referenser till strömmar och till HanoiEngine-objektet, information om ett drag osv. Detta styr naturligtvis listan av argument till konstruktorn.

Inkapsling och principen att ett objekt ska ansvara för sig själv ska tillämpas. Det innebär bl.a. att ett Command-objekt

- själv skriver till log-filen i sin execute-funktion.
- själv läser in de data som behövs för att återskapa objektet, i funktionen readFromStream.

Implementera Replay-mekanismen i en egen klass (Replayer / ReExecutor).

Ett scenario i samband med replay kan vara:

- en läsfunktion läser första ordet från en rad i logfilen. Ordet innehåller klassnamnet för det Command-objekt som ska skapas.
- läsfunktion skapar ett sådant Command-objekt.
- läsfunktion anropar det nyskapade Command-objektets readFromStream-funktion med logfilens streamobjektet som argument. Command-objektet

vet vad som ska läsas in och läser resten av raden i filen för att få de data som det behöver.

Command-objektet är nu klart för vidare hantering/exekvering.

Krav på lösningen
Minst två typer av Command ska hanteras. Förutom det uppenbara MoveCommand ska också ResetCommand (som initierar ett nytt spel) och eventuellt ett ShowCommand (som visar upp aktuell ställning) användas.

Redovisning
Zippad fil innehållande samtliga källkodsfiler samt ett dokument som kort beskriver lösningen skickas in. Hela projektet om du använder Visual Studio.