Tower of Hanoi

Student:

Doană Vasile-Bogdan

Cuprins

[Introducere 3](#_Toc152431034)

[Cerinta problemei 4](#_Toc152431035)

[Solutia teoretica 5](#_Toc152431036)

[Sabloane folosite 6](#_Toc152431037)

[Indicele de stabilitate 11](#_Toc152431038)

[Plan de testare 12](#_Toc152431039)

[Bibliografie 14](#_Toc152431040)

# **Introducere**

Turnul din Hanoi este o problemă clasică în lumea programării. Configurarea problemei constă din trei tije/cuioare și n discuri.

Discurile pot fi mutate de la un cuier la altul. Cele n discuri sunt de dimensiuni diferite.

Inițial, toate discurile sunt stivuite pe primul turn. Discurile sunt stivuite în așa fel încât un disc să fie întotdeauna peste un disc mai mare decât el.

Tower of Hanoi setup


# **Cerinta problemei**

“ Mutați toate discurile stivuite pe primul turn în ultimul turn folosind un turn de ajutor în mijloc. La mutarea discurilor, trebuie respectate anumite reguli. Acestea sunt :

1. Numai un singur disc poate fi mutat.

2. Un disc mai mare nu poate fi plasat pe un disc mai mic. “

Deci trebuie să mutați toate discurile de la primul turn la ultimul. Puteți muta doar un disc la un moment dat și nu așezați niciodată un disc mai mic peste un disc mai mare.

Pentru a face acest lucru aveți un turn suplimentar, acesta fiind cunoscut sub numele de turn de ajutor/auxiliar.

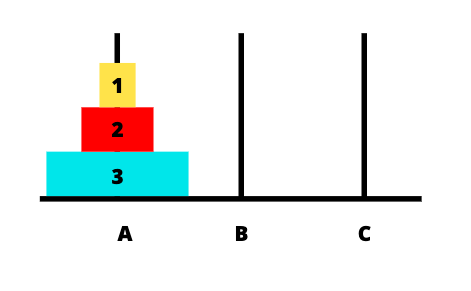
Deoarece puteți muta doar un disc la un moment dat, discul pe care îl mutați va trebui să fie în vârful turnului său.

# **Solutia teoretica**

Rezolvăm această întrebare folosind recursiunea simplă. Pentru a duce cele trei discuri la turnul final, trebuie să:

1. Luați discurile cu numărul 1 și 2 în turnul B.
2. Mutați discul numărul 3 în turnul C.
3. Luați discul numărul 1 și 2 de la B la C.

Desigur, nu poți face asta din cauza constrângerilor. Cu toate acestea, putem folosi acest lucru pentru a crea o funcție care o face recursiv.



# **Sabloane folosite**

În ingineria software, un model de proiectare software este o soluție generală, reutilizabilă la o problemă care apare frecvent într-un context dat în proiectarea software. Nu este un design finit care poate fi transformat direct în cod sursă sau mașină. Mai degrabă, este o descriere sau un șablon pentru cum se rezolvă o problemă, care poate fi folosită în multe situații diferite. Modelele de proiectare sunt cele mai bune practici formalizate pe care programatorul le poate folosi pentru a rezolva probleme comune atunci când proiectează o aplicație sau un sistem.

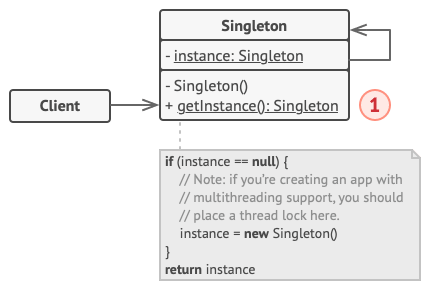
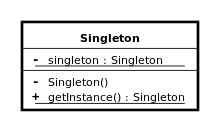
Modelele de proiectare orientate pe obiecte arată de obicei relații și interacțiuni între clase sau obiecte, fără a specifica clasele sau obiectele de aplicație finale care sunt implicate. Modelele care implică o stare mutabilă pot fi nepotrivite pentru limbaje de programare funcționale. Unele modele pot deveni inutile în limbaje care au suport încorporat pentru rezolvarea problemei pe care încearcă să o rezolve, iar modelele orientate pe obiecte nu sunt neapărat potrivite pentru limbajele neorientate pe obiecte.

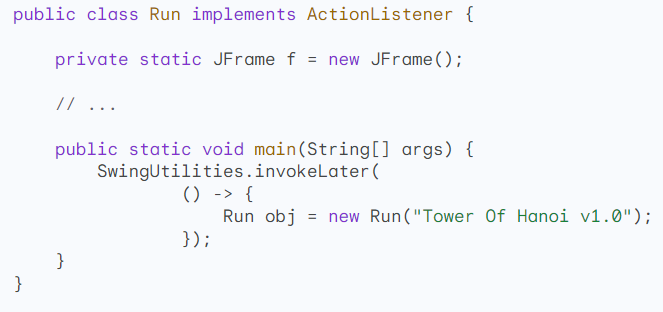
Modelele de proiectare pot fi privite ca o abordare structurată a programării computerelor intermediare între nivelurile unei paradigme de programare și un algoritm concret.

**Singleton**

În ingineria software, modelul singleton este un model de proiectare software care restricționează instanțiarea unei clase la o instanță singulară. Unul dintre binecunoscutele modele de proiectare „Gang of Four”, care descrie cum să rezolve probleme recurente în software-ul orientat pe obiecte, modelul este util atunci când este nevoie de un singur obiect pentru a coordona acțiunile dintr-un sistem.

Clasa Run utilizează modelul singleton pentru a se asigura că există o singură instanță a clasei Run. Acest lucru este important deoarece clasa Run gestionează starea jocului și vizualizarea.



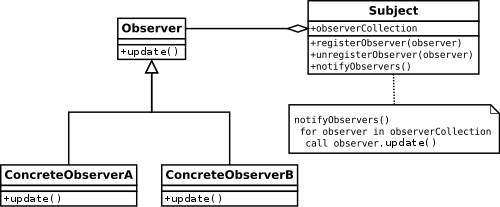
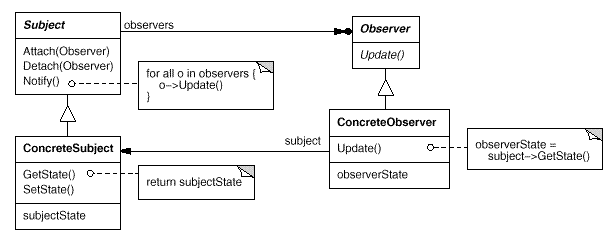


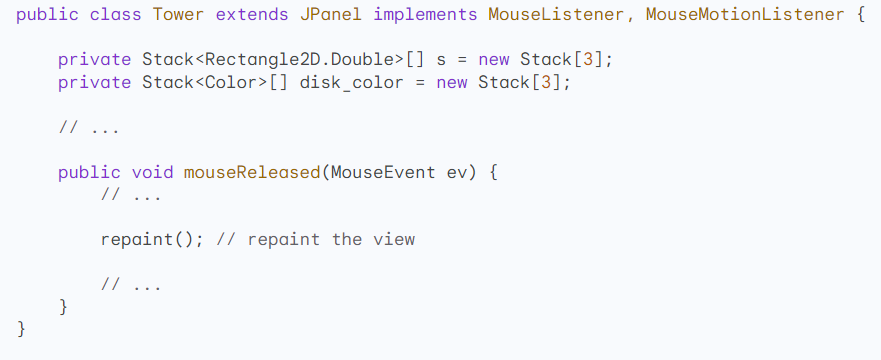
**Observer**

În proiectarea și inginerie software, modelul de observator este un model de proiectare software în care un obiect, numit subiect, menține o listă a dependenților săi, numiți observatori, și îi notifică automat despre orice schimbare de stare, de obicei apelând una dintre metodele lor.

Interfața ActionListener este folosită pentru a observa acțiunile evenimentelor din interfața grafică (cum ar fi acțiunile din meniu sau butoanele). Metoda actionPerformed este apelată atunci când un eveniment este activat (de exemplu, când se apasă un buton din meniu).

Clasa Tower folosește modelul de observator pentru a notifica vizualizarea când starea jocului se schimbă. Acest lucru asigură că vizualizarea este întotdeauna actualizată cu cea mai recentă stare a jocului.

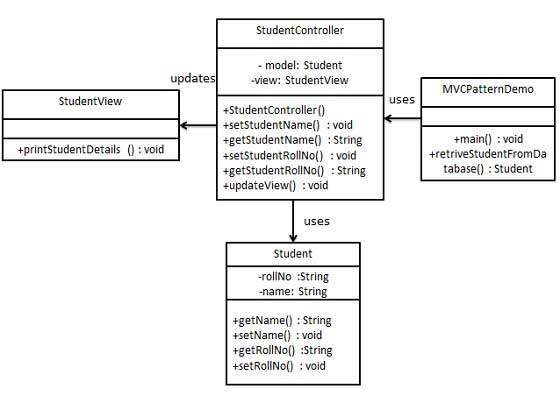


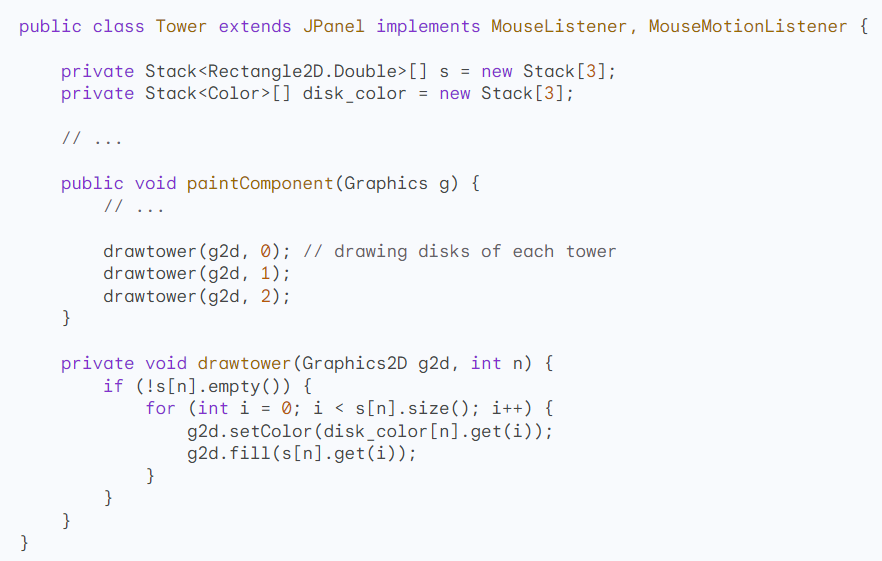


**Model-view-controller**

Model–view–controller (MVC) este un model de proiectare software utilizat în mod obișnuit pentru dezvoltarea interfețelor utilizator care împarte logica programului aferentă în trei elemente interconectate. Aceste elemente sunt reprezentările interne ale informațiilor (Modelul), interfața (Vizualizarea) care prezintă informații și le acceptă de la utilizator și software-ul Controller care leagă cele două.

Codul este scris folosind modelul Model-View-Controller (MVC). Modelul este reprezentat de clasa Tower, care conține datele despre starea jocului. Vederea este reprezentată de clasa Tower, care pictează și starea jocului pe ecran. Controlerul este reprezentat de clasa Run, care se ocupă de intrarea utilizatorului și actualizează modelul și vizualizarea în consecință.

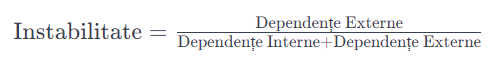






# **Indicele de stabilitate**

Calculul indicelui de stabilitate pentru fiecare pachet din codul dat necesită identificarea dependențelor dintre pachetele respective. Indicele de stabilitate poate fi calculat folosind formula:



Pentru a calcula indicele de stabilitate a proiectului, avem nevoie să calculăm valorile cuplajului aferent (CA) și cuplajului eferent (CE).

**Cuplajul aferent (CA)** este numărul de clase din afara pachetului care depind de clasele din pachet. În cazul proiectului Tower of Hanoi, există o singură clasă din afara pachetului care depinde de clasele din pachet, și anume clasa Run. Această clasă depinde de clasa Tower pentru a obține datele despre jocul Tower of Hanoi.

**CA = 1**

**Cuplajul eferent (CE)** este numărul de clase din interiorul pachetului care depind de clasele din afara pachetului. În cazul proiectului Tower of Hanoi, nu există nicio clasă din interiorul pachetului care să depindă de clasele din afara pachetului.

**CE = 0**

Prin urmare, indicele de instabilitate este:

**I = CE / (CA + CE) = 0 / (1 + 0) = 0**

Având în vedere că indicele de instabilitate este egal cu 0, proiectul este total stabil.

# **Plan de testare**

Scop:

* Asigurarea funcționării corecte a jocului Tower of Hanoi și a interfeței cu utilizatorul.

Obiective:

1. Verificarea inițializării jocului la deschiderea aplicației.
2. Asigurarea funcționării butoanelor din meniu (**New Game**, **Best Time**, **Exit**, **How to Play**).
3. Testarea corectitudinii mișcărilor discelor pe turnuri.
4. Verificarea condiției de câștig și afișarea mesajului corespunzător.
5. Testarea afișării timpului și a listei cu cele mai bune timpuri.

Cazuri de Testare:

1. **Inițializare joc (New Game):**
   * **Acțiune:** Se inițializează un joc nou cu numărul specificat de discuri.
   * **Justificare:** Verifică dacă jocul se inițializează corect cu numărul corect de discuri.
2. **Interacțiune butoane din meniu:**
   * **Acțiune:** Se apasă pe fiecare buton din meniu (New Game, Best Time, Exit, How to Play).
   * **Justificare:** Verifică funcționarea corectă a butoanelor și meniului.
3. **Mișcare disc:**
   * **Acțiune:** Se trage și se plasează un disc pe un alt turn.
   * **Justificare:** Testează corectitudinea mișcării și interacțiunea cu discul.
4. **Verificare câștig:**
   * **Acțiune:** Se mută toate discurile pe ultimul turn pentru a câștiga jocul.
   * **Justificare:** Verifică dacă condiția de câștig este detectată corect.
5. **Afișare timp și cele mai bune timpuri:**
   * **Acțiune:** Se termină jocul și se verifică afișarea timpului și a celor mai bune timpuri.
   * **Justificare:** Verifică corectitudinea afișării timpului scurs și a celor mai bune timpuri

# **Bibliografie**

<https://mathworld.wolfram.com/TowerofHanoi.html>

<https://www.geeksforgeeks.org/java-program-for-tower-of-hanoi/>

<https://www.digitalocean.com/community/tutorials/tower-of-hanoi#implementing-the-solution-to-tower-of-hanoi-in-java>

<https://en.wikipedia.org/wiki/Tower_of_Hanoi>