Puzzle Glisant

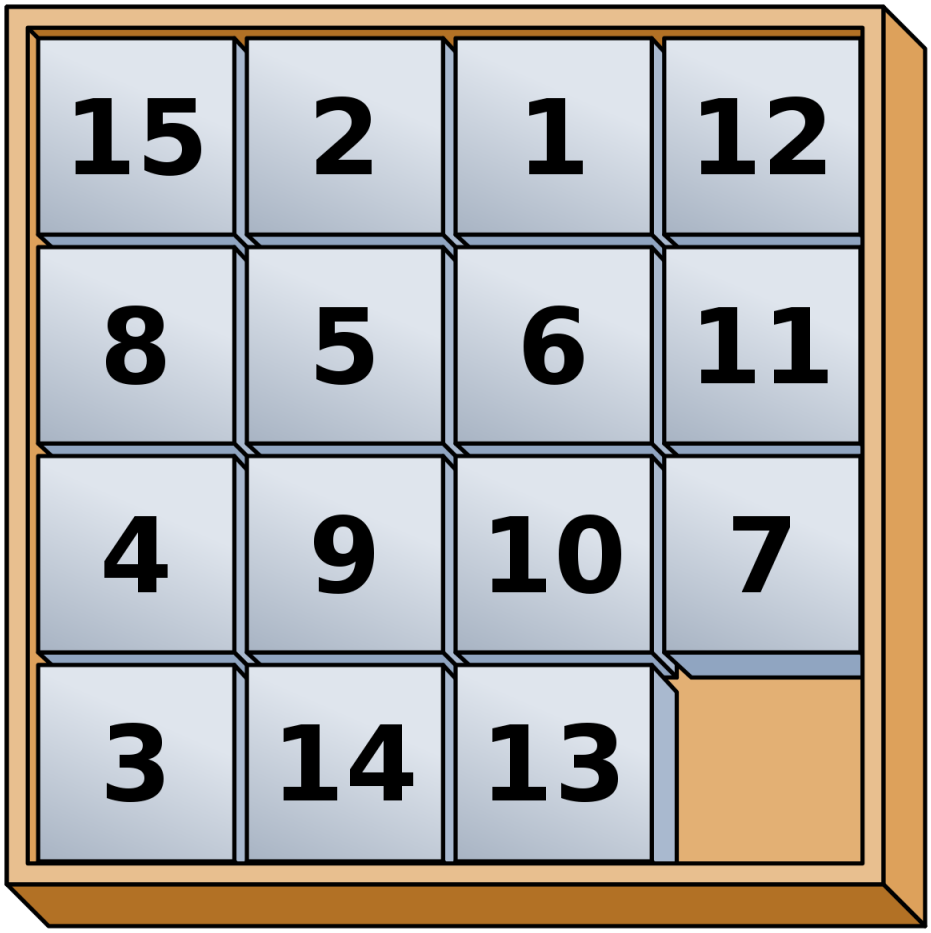
Proiect la Programare Orientata Obiect

An:2019-2020

Student: Chiras Valentin Fanica

Profesor laborator: Ana-Maria Nanes

Puzzle Glisant



Jocul de **puzzle** constă în reconstituirea unei imagini sau a unui obiect tridimensional având la numeroase piese (diferite ca formă și culoare) ce se pot intercala. Pentru cele mai multe persoane, această activitate este un hobby.

Istoria acestui tip de joc pornește de la o pictură sau un desen realizat pe o placă de lemn care apoi este decupată în mici forme ce pot fi mai apoi reasamblate. După aproximativ 300 de ani de la primele forme de joc, puzzle-ul este astăzi realizat din carton, întrucât acest material este mai convenabil. De asemenea, în locul picturii, intervine tiparul, astfel că jocurile pot avea un conținut mare de detalii precum și o libertate mai mare în alegerea imaginii ce urmează a fi realizată. După tipar, cartonul este decupat folosind un cuțit-matriță metalic după un contur complex.

Jocul poate avea de la 4-9 piese până la mii de piese, mărind astfel dificultatea dar și interesul pasionaților.

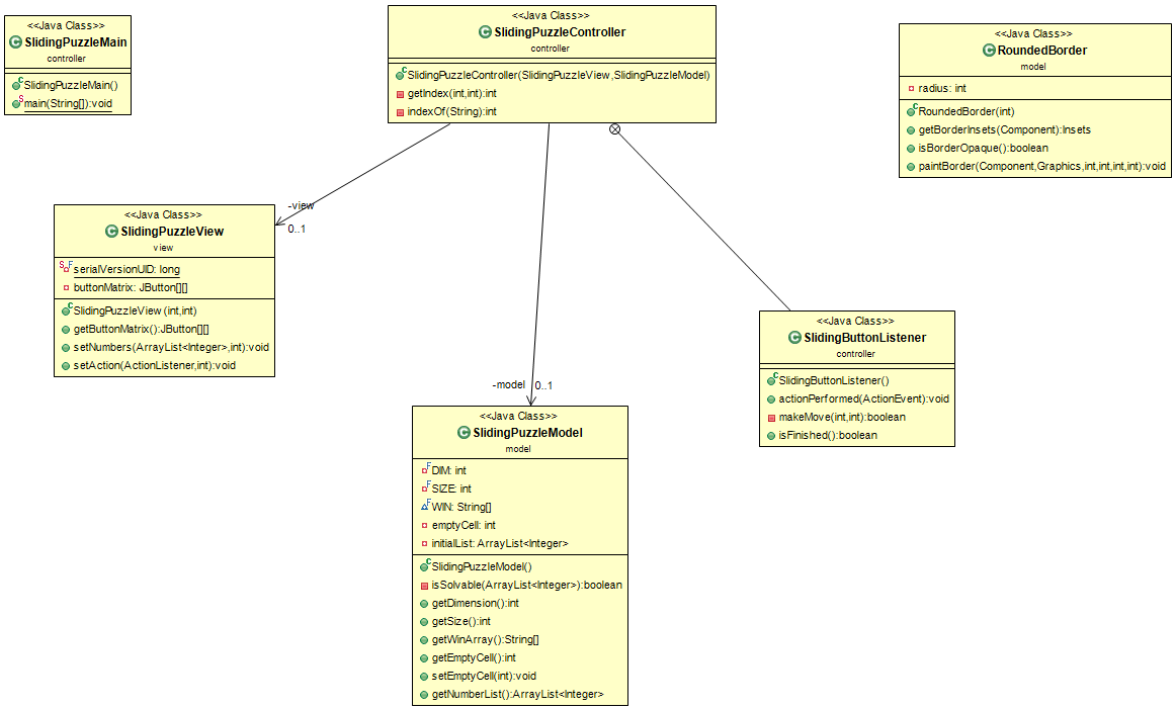
Modelul MVC

Model-ul se ocupa de comportarea si datele aplicatiei; raspunde la cereri despre starea sistemului, la cereri de schimbare de stare si notifica utilizatorul atunci cand aceste schimbari au avut loc pentru ca acesta sa poata reactiona.

View-ul transpune model-ul intr-o forma care permite o interactionare usoara, in mod tipic o interfata vizuala. Pot exista multiple view-uri pentru un singur model pentru scopuri diferite.

Controller-ul primeste input de la utilizator si initiaza un raspuns in urma cererilor catre obiectele model. Controller-ul este cel care controleaza celelalte doua clase de obiecte, view si model, instructandu-le sa execute operatii pe baza input-ului primit de la utilizator.

Diagrama UML



Clasele implementate

- **RoundedBorder**

Aceasta clasa face ca Butoanele panoului sa fiu rotunde ! Aceasta clasa implementeaza metode din clasa deja existenta in pachetul java.swing.border.Border.

- **SlidingPuzzleView**

Aceasta clasa creaza partea grafica a jocului. Se ia modelul abstract si se genereaza o matrice de butoane poitionata intr-un GridLayout. Tot aici se pun si numerele sub forma de text pe butoane de la “1” la “15” butonul “0” find cel care lipseste

- **SlidingPuzzleModel**

Acesta clasa creaza matricea de butoane pe care o amesteca astfel incat sa poate fi rezolvata.

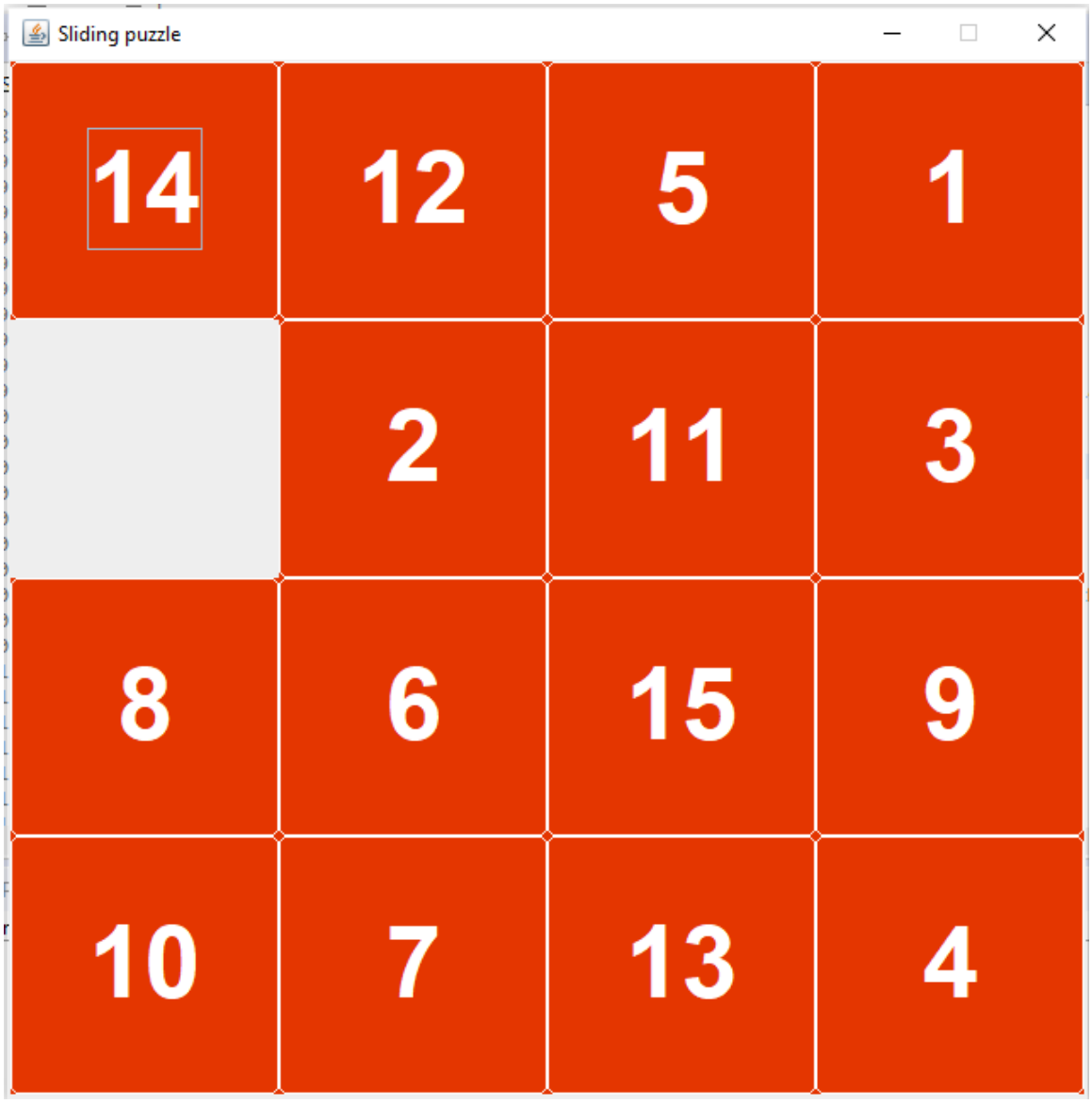
Metoda **isSolvable(ArrayList<Integer> numberList)** verifica daca matricea de butoane intiala poate fi rezolvata. Metodele **getDimension(), getSize(), getWinArray(), getEmptyCell(), getNumberList()** sunt gettere care returneaza numarul de butoane de pe un rand, numarul de butoane de pe panou, vectorul final la care trebuie ajuns pentru a castiga, celula/butonul “empty”, respectiv lista initiala de butoane.

- **SlidingPuzzleController**

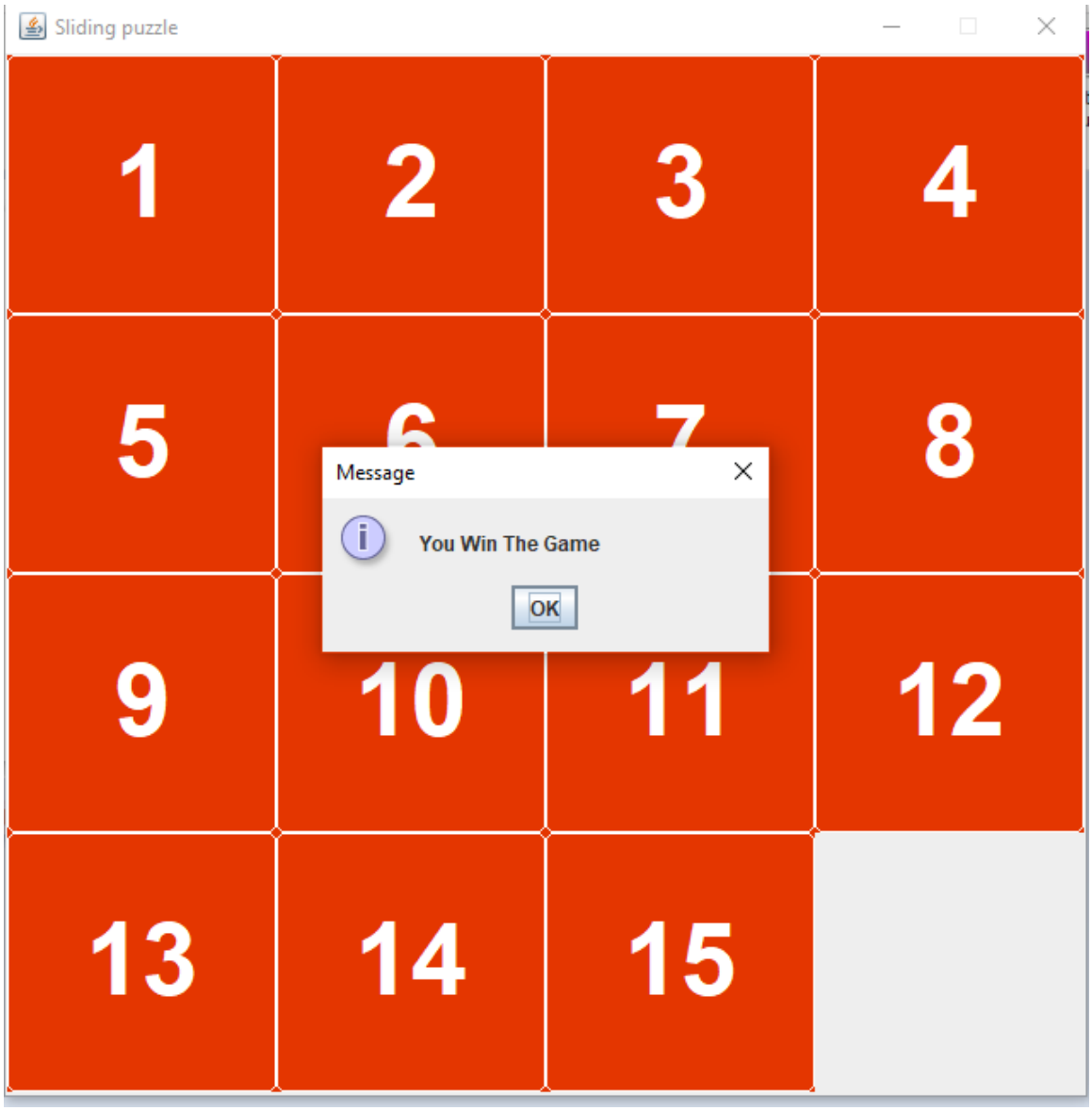
Aceasta clasa logica pentru evenimentele care au loc la fiecare click pe GUI. Aici la fiecare click metoda MakeMove verifica daca se poate executa miscarea butonului in zona libera. La fiecare eveniment, metoda **isFinished()** verifica daca fiecare buton se afla la locul sau. Daca da se va afisa o bara de dialog care te felicita apoi alta care te pune sa alegi daca joci din nou sau iesi.

Implementare ulterioara si dezvoltarea

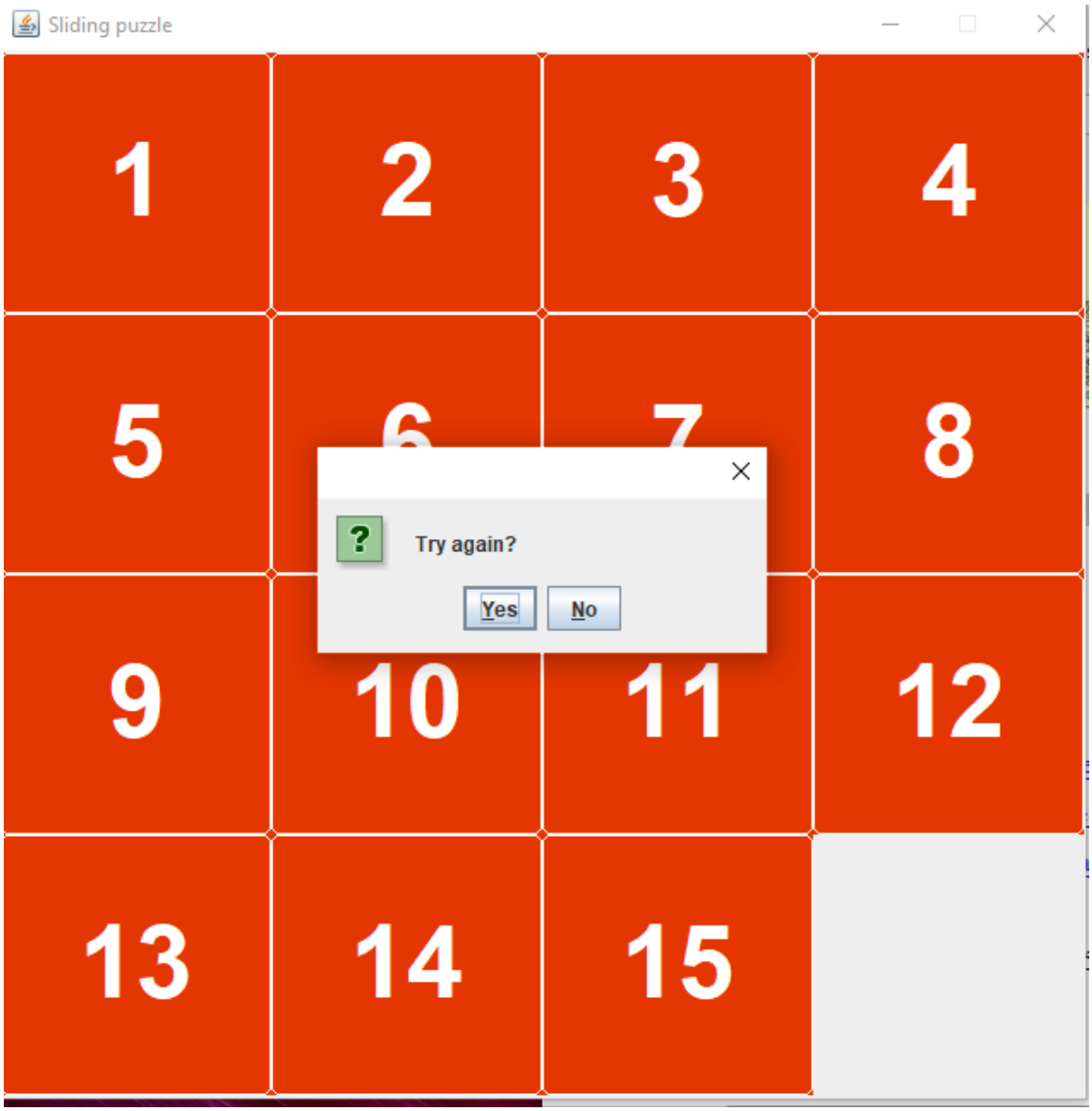
- Adaugarea unui timer
- Un meniu in care sa legi intre 1-15 sau 1-8 puzzle
- Adaugarea unui buton de reset care sa reseteze ordinea butoanelor cand doreste jucatorul


Sliding puzzle
14
12
5
1
2
11
3
8
6
15
9
10
7
13
4

Win the game!



Try again?



Bibliografie

- https://ro.wikipedia.org/wiki/Puzzle

- http://stst.elia.pub.ro/news/IS/IS_Teme1101/mvc.pdf
    - https://www.google.com/search?q=1-15+puzzle&source=lnms&tbm=isch&sa=X&ved=2ahUKEwie5u3BmvTmAhUDrosKHfa4CIYQ_AUoAXoECAsQAw&biw=1366&bih=655#imgrc=N1IApGs0xqebZM: