Facoltà di Ingegneria

Corso di laurea in Ingegneria Informatica

“Sudoku”

Professore del corso di POO:
Prof. Libero Nigro

Studente:
Carmelo Gugliotta

ANNO ACCADEMICO 2020-2021

Progetto: Sudoku

Il progetto consente di visualizzare le soluzioni possibili di un problema (gioco) di sudoku.

N.B: Si ricorda che nel gioco del Sudoku si dispone di una matrice 9x9 nelle cui celle vanno inseriti valori da 1 a 9 nel rispetto delle regole del gioco stesso.

La gerarchia di classi utilizzata per lo sviluppo del progetto è la seguente:

1. Sudoku (**Classe Concreta**)
2. SudokuGUI (**Classe Concreta, GUI**)

1. Sudoku (Classe Concreta)

La seguente classe estende la classe Backtracking<P, S>, ove P indica il punto e S la scelta.

All'interno della classe Sudoku, P è un array di due elementi che indicano la i-esima riga e j-esima colonna ove inserire il valore S che è un intero che appartiene al seguente intervallo (0,9].

Per raggiungere lo scopo proposto si ridefiniscono i metodi lasciati astratti in BackTracking<P, S> e si implementano alcuni metodi di supporto.

- Sudoku ()
  - Il costruttore si occupa di un istanziamento corretto dell'oggetto Sudoku, inizializzando una matrice 9x9 di soli zeri, una struttura dati che consentirà di allocare in memoria le possibili soluzioni, un contatore delle soluzioni trovate e un booleano che permetterà di stabilire se c'è stato un tentativo di superamento di capacità.
- Risolvi (String Espressione)
  - Riceve come parametro un oggetto stringa, che è l'espressione che deve essere valutata, e ritorna un intero che il valore dell'espressione. **Lancia IllegalArgumentException** se l'oggetto stringa corrisponde a un'espressione malformata (Il controllo avviene tramite la Regex).
    Il valore dell'espressione ritornato è calcolato tramite il metodo ValutaEspressione.

- imposta (**int** i, **int** j, **int** v)
  - Riceve come **parametri** tre interi, i e j indicano la posizione della cella ove deve essere inserito il valore v. Lancia **IllegalArgumentException ()** durante la sua esecuzione se è stato richiesto di inserire un valore in una posizione inesistente, o il valore non è un intero tra **[1,9]**. Lancia **IllegalStateException ()** se il valore non è assegnabile in quel punto.

- assegnabile (**int []** p, **Integer** s)
  - Riceve come parametri, un array **p** che contiene le coordinate del punto **[i, j]**,e **s** che corrisponde al valore da inserire in quel punto.
    Se in quel punto è stato già assegnato un valore restituisce **false**.
    Se è possibile assegnare il valore nella cella selezionata restituisce **true**.

La possibile assegnazione di un valore dipende strettamente dai vincoli che pone il gioco, implementati tramite i metodi **colonna (), riga (), SottoMatrice ()**.

- assegna(**int []** ps, **Integer** s)
  - Riceve come parametri un punto di scelta (l'array **ps**) e un valore intero s. Assegna il valore s alla cella le cui coordinate sono contenute in **ps**.
- deassegna (**int []** ps, **Integer** s)
  - Riceve come parametri un punto di scelta (l'array **ps**) e un valore intero s. Assegna il valore 0 alla cella le cui coordinate sono contenute in **ps**.
- ultimasoluzione(**int []** p)
  - Se il contatore N_Sol>=40, imposta Limite_Sup a true e restituisce true. In caso contrario restituisce false.
    Permette di allocare nella struttura dati (LinkedList) un numero finito di soluzioni.
- esisteSoluzione(**int []** p)
  - Restituisce un booleano che indica se è stata trovata una soluzione (Si->True; No->False).
    La sua funzionalità si basa sul controllare se esistono ulteriori puntiDiScelta sulla matrice di gioco corrente in cui deve essere inserito un valore tra **[1,9]**.
- scriviSoluzione (**int []** p)
  - Aumenta il contatore N_Sol, alloca nella struttura dati (LinkedList) la soluzione trovata e la stampa in output.
- PuntiDiScelta ()
  - Alloca all'interno di una struttura dati (List) le coordinate delle celle non bloccate, alle quali è possibile assegnare un valore, che viene infine restituita.
- scelte (**int []** p)
  - Riceve come parametro un punto di scelta (l'array **p**).
    Restituisce una Collection di Integer, che contiene tutti i valori assegnabili al punto di scelta passato come parametro.
- risolvi ()
  - Inizia il processo di risoluzione della matrice di gioco, richiamando il metodo tentativo (List<P> ps, P p).
- MatriceGiocoCurr ()
  - Restituisce una nuova matrice che è la copia (DEEP COPY) della matrice di gioco. (EVITA L'ALIASING).
- colonna (int j,int v)
  - Riceve come parametri due interi, j che corrisponde alla j-esima colonna ove effettuare i controlli per il valore intero v.
    Restituisce true se il valore v non è già presente lungo la j-esima colonna, false altrimenti.
- riga (int i, int v)
  - Riceve come parametri due interi, i corrisponde alla i-esima riga ove effettuare i controlli per il valore intero v.
    Restituisce true se il valore v non è già presente lungo la j-esima riga, false altrimenti.
- SottoMatrice (int r, int c, int v)

- Riceve come parametri tre interi, r e c indicano (riga-colonna) le coordinate della cella ove effettuare i controlli per il valore v.
  Restituisce true se il valore v non è già presente all'interno della sottomatrice, false altrimenti.
- getSoluzioni (int r, int c, int v)
  - Restituisce una LinkedList di matrici di interi, che è la copia della struttura dati (LinkedList) utilizzata per allocare le soluzioni. (Shallow-copy).
- TroppeSoluzioni()
  - Restituisce Limite_Sup, il booleano che indica se sono state prodotte più di 40 soluzioni. (true->Si, false->No)

2. SudokuGUI (Classe Concreta, GUI)

La classe concreta presenta un main, in grado di istanziare la Classe FinestraGUI e di visualizzare a schermo una vera e propria Graphical User Interface che tramite determinate interazioni permette di visualizzare le possibili soluzioni di una sessione di gioco.

La classe FinestraGUI estende e presenta i seguenti metodi:

- FinestraGUI ()
  - Costruttore che si occupare del corretto istanziamento della Finestra, con cui l'utente dovrà successivamente interagire.
    Si noti che durante l'istanziamento della Finestra viene istanziato anche un oggetto AscoltatoreEventiAzione (la classe implementa ActionListener) che consente la gestione delle seguenti componenti:
    - **JMenuItem**: nuova,apri,salva,salvaConNome, esci, risolvi, about;
    - **JButton**: Next,Previous;

    Si noti anche che la finestra istanziata non sarà ridimensionabile a piacere tramite il mouse dall'utente, poiché per evitare di cicalare sul GridDiGioco[][] ,che contiene al suo interno tutti i JTextField (celle) ,ogni volta per determinare quale delle 81 celle fosse stata modificata, si è preferito attuare delle operazioni sulle posizioni rispetto al pannello dei componenti JTextField. Il problema è che se la size della finestra viene modificata continuamente dall'utente, tramite Componente.getLocation() c'è la possibilità che venga restituita una posizione passata e non corrente.

    https://docs.oracle.com/javase/7/docs/api/java/awt/Component.html#getLocation() : "Due to the asynchronous nature of native event handling, this method can return outdated value".

- PreMenu()
    - Si occupa di disabilitare alcune componenti se l'utente non ha ancora inizializzato una sessione di gioco.
- Start()
    - Si occupa di abilitare alcune componenti se l'utente inizializza una sessione di gioco
- NewGame()
    - Si occupa di inizializzare in modo corretto una nuova sessione di gioco, svuotando i JTextField presenti nel GridDiGioco e istanziando un nuovo oggetto della classe Sudoku.
- consensoUscita()
    - Si occupa di gestire il caso in cui l'utente volesse uscire dal programma.
      N.B: nel costruttore abbiamo ridefinito l'operazione di chiusura aggiungendo un WindowListener con parametro un WindowAdapter() (Per ridefinire solo i metodi interessati).
- InizializzaCelle()
    - Si occupa dell'inizializzazione delle 81 celle di gioco.
      In particolare, per ogni cella corrisponderà un JtextField, il quale riferimento sarà presente all'interno del GridDiGioco e che sarà anche aggiunto al pannello SudokuPanel per permettere la sua visualizzazione a schermo.
      Importante è il listener ,che serve a gestire il caso in cui l'utente interagisse con la cella.
      Il listener viene istanziato tramite una lambda expression.
      Nel caso in cui l'utente interagisse con uno dei JTextField, le coordinate della cella corrispondente vengono calcolate nel seguente modo:

      N.B: la larghezza (x) di una cella è possibile calcolarla dividendo la larghezza del Sudoku_Panel per 9. La lunghezza (y) è possibile calcolarla dividendo la lunghezza del Sudoku_Panel per 9.
      Per determinare la colonna:
        - Viene presa la coordinata X del JtextField e tramite il risultato della seguente operazione
          "N_Box.getLocation().getX()/(Sudoku_Panel.getSize().getWidth()/9"
          Si calcola quante volte la larghezza di una cella sta nella posizione X del componente.
          Si è calcolata quindi la colonna corrispondente.

      Per determinare la riga:
        - Viene presa la coordinata y del JtextField e tramite il risultato della seguente operazione
          "N_Box.getLocation().getY()/(Sudoku_Panel.getSize().getHeight()/9)"
          Si calcola quante volte la lunghezza di una cella sta nella posizione Y del componente.

      N.B: N_Box indica (anche nel codice) la JTextField con cui l'utente ha cercato di interagire.

      Se l'utente inserisce un valore non valido per la posizione scelta, un JOptionPane si preoccuperà di avvisarlo.

Se l'utente inserisce un valore valido per la posizione scelta, la cella cambierà di colore (verde) e conserverà al suo interno il valore inserito.

- DisabilitaCelle()
  - Si occupa di interrompere la possibilità di interagire con le 81 celle del GridDiGioco, nel caso si volessero visualizzare le soluzioni del gioco.
- VisualizzaMatriceSoluzione(int[][] Matrice)
  - Riceve una Matrice, e visualizza a schermo il corrispondente contenuto, inserendo i valori all'interno dei JTextField di cui è composto il GridDiGioco. Utile per visualizzare le N-Soluzioni presenti all'interno della LinkedList.
- refresh()
  - Si occupa della visualizzazione corretta dei componenti di cui è composta la finestra.
    Ogni qualvolta si inseriscano dei componenti alla finestra è bene rivalidare il Frame e attuare un repaint dello stesso.
- ripristina(String nomeFile)
  Riceve come parametro un oggetto stringa il cui contenuto è caratterizzato dal nome di un file. Durante la sua esecuzione può lanciare IOException.
  Si occupa di ripristinare un gioco precedente tramite l'utilizzo di un BufferedReader.
- salva(String nomeFile)
  Riceve come parametro un oggetto stringa il cui contenuto è caratterizzato dal nome di un file. Durante la sua esecuzione può lanciare IOException.
  Si occupa di salvare la sessione di gioco visualizzata a schermo.