Academia de Studii Economice Bucuresti

Faculatea de Cibernetica, Statistica si Informatica Economica

Documentatie proiect

Piscuc Dinu

Seria B, Grupa 1054

1. Descrierea si detalierea pattern-urilor utilizate

In implementarea functionalitatii aplicatiei Sudoku am folosit design patterns, fiecare din ele servind unui scop precis.

Prin intermediul intefetei jucatorul alege dificultatea jocului (joc usor sau joc greu) iar programatic aceasta este implementata prin intermediul unui design pattern de tip strategie. Dupa definirea interfetei StrategieJoc, definim si cele doua trepte de dificultate ale jocului : JocGreu si JocUsor, fiecare din ele implementand interfata anterior definita - StrategieJoc.

Pentru modelarea eficienta a clasei TabelaSudoku am folosit doua design patternuri : composite si facade. Clasa face uz de modelul composite prin faptul ca contine ca atribute doua alte clase definite de programator : ButtonController, PatratTablaJ si StrategieJoc.

Tiparul facade presupune ca una din functiile / operatiile clasei ce-l implementeaza sa unifice mai multe operatii sub tutela unei singure functii/operatii. In clasa TabelaSudoku gasim functia genereaza\_joc() care apeleaza in corpul ei alte functii predefinite de programator : genereazaBlocIntreg(int, int), getVal(int, int), getStratSelectata() si setStrategie(StrategieJoc).

1. Definirea si detalierea metodelor testate prin Unit Testing

Constructor CelulaS

CelulaS.incearcaVal(int numver) - memoreaza intr-un set numarul incercat de jucator

CelulaS.aIncercat(int number) - returneaza valoare de adevar a expresiei : a incercat valoarea 'number'

CelulaS.reset() - reseteaza toate atributele clasei inclusiv hashset-ul

CelulaS.nrIncercari() - returneaza numarul de incercari ale jucatorului

CelulaS.clear() - reseteaza toate atributele clasei in afara de hashset

CelulaS.getVal() - returneaza valoarea inscrisa in celula

CelulaS.setVal() - seteaza valoarea

Constructor BlocS

BlocS.setVal(int val, int row, int column) - seteaza valoarea intr-o pozitie a matricei precizata

BlocS.getVal(int row, int column) - returneaza valoarea din matricea de CelulaS

BlocS.verificaNumar(int number, int row, int column) - returneaza boolean, adevarat daca valoarea number este solutie, fals altfel

BlocS.reset() - reseteaza intreaba matrice de CelulaS

BlocS.genereazaBlocIntreg(int a, int b) - genereaza o matrice de CeluleS cu lungimeBloc randuri si lungimeBloc coloane. Functia se apeleaza recursiv

BlocS.areContinut() - verifica daca toate celulele matricei sunt completate

BlocS.getPerimetru() - returneaza perimetrul blocului

Constructor JocGreu

Constructor JocUsor

StrategieJoc.genereazaJoc(List<Integer> positons) - salveaza in atributul joc matricea de celule din care au fost eliminate aleator valori inscrise

StrategieJoc.getJoc() - returneaza matricea joc

1. Definirea si descrierea Test Case-urilor

testAIncercat - verifica functionalitatea de baza a clasei CelulaS si anume functia a incercat

testIncearcaNumar - verifica functionalitatea functiei incearcaNumar

testReset - verifica functionalitatea functiei reset

testNumarIncercari - verifica functionalitatea functiei numar incercari

testClear - verifica functionalitatea functiei clear

testBlocSConstructor testeaza constructorul clase BlocS. In cazul in care constructorul primeste un parametru negativ se va arunca o exceptie cu mesaj

testValiditate - se citesc din fisier solutii valide ale jocului de sudoku si se verifica functionalitatea verifica functionalitatea functiilor getVal setVal si verificaNumar

testGenereaza - testam daca matricea generataya este solutie

testGreu - verific limita de timp de executie a functiei genereazaJoc din clasa JocGreu

testeazaStrategii - verific rezultatele diferitelor strategii

1. Definirea si descrierea Test Suite-ului

Suita AllTests cuprinde toate Case Test-urile prezentate anterior

1. Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate