**Proiect Java: Word Searcher**

Vîjaică Paul-Augustin

**Proiectul Java** constă în crearea unui joculeț numit Word Searcher.

Acest mini-joc conține o tabelă în care se află litere random, iar printer ele se află și cuvinte.

Jucătorul trebuie să găsească toate cuvintele din tabelă într-un anumit număr de încercări.

Dacă jucătorul nu reușește să găsească toate cuvintele și nu mai sunt încercări rămase, acesta a pierdut jocul.

Ideea mea a fost de a împărți acest joc în două părți, asttel:

O clasă se va ocupa de afișarea tabelei, iar alta va gestiona datele introduse de către jucător.

Pentru acest proiect am adoptat modelul de design MVC (Model-View-Controller).

Așadar, clasele mele **WordSearch** și **Input** vor avea fiecare câte un *Model*, un *View* respectiv un *Controller*.

Voi încerca să realizez un scurt rezumat al conținutului fiecărei clase.

**WordSearch**

***Model***

În această clasă citesc datele dintr-unul din cele patru fișiere (am vrut să pară că generarea tabelei este Random) în funcție de rezultatul împărțirii modulo 4 a unui număr generat random și literele citite le adaug într-o matrice de String de 10x10.

***View***

Aici lucrez pe Jframe. Am un Jpanel și, folosind GridLayout, adaug o matrice de JTextField (10x10). În acea matrice de JTextField voi insera matricea de String din *Model.*

***Controller***

Pur și simplu am legat *View*-ul și *Modelul*.

**Input**

***Model***

Selectez soluția corespunzătoare tabelei alese (în funcție de același n random de la *WordSearchModel*)*.* După ce a fost soluția aleasă (fișierul Solution.txt), conținutul său (cuvintele care trebuie găsite) va fi adăugat într-o listă ”words” (Ne va trebui mai târziu).

***View***

Partea vizuală a Input-ului. Aici am adăugat butonul ”Check” și în JTextField jucătorul va introduce cuvinte pentru a le găsi pe cele din tabelă. Am decis să folosesc două ferestre pentru acest joc pentru a-i fi mai ușor jucătorului să vadă cuvintele (am evitat adăugarea Input-ului și tabelei în aceeași fereastră, deoarece, astfel trebuia să micșorez dimensiunea tabelei de afișat).

Totodată, aici se vor afișa mesaje în funcție de ce a introdus jucătorul (Win, Game Over, cuvânt deja găsit etc.).

***Controller***

Aici este partea cea mai importantă: verificarea Input-ului. În această clasă am creat ActionListener-ul pentru butonul ”Check” și pentru JTextField (lumea de obicei tinde, pur și simplu să apese **ENTER**, în detrimentul apăsării unui buton). În acest ActionListener voi verifica dacă textul este format din caractere care nu reprezintă litere și voi afișa mesajul: ”Nu ați introdus nimic.” În caz contrar, voi transforma conținutul extras din Input, convertind toate literele în LowerCase (litere mici) și voi continua verificările. Dacă cuvântul se află în tabelă (adică, mai exact, există în lista ”words”), voi afișa mesajul ”Cuvânt găsit”, iar apoi voi elimina cuvântul din lista ”words”, dar nu înainte de a-l copia într-o altă listă, ”words2”, listă pe care o voi afișa în fereastră. Următorul pas al verificării este căutarea cuvântului introdus în Input în lista ”words2”. Dacă acesta există acolo, înseamnă că a fost deja găsit, și astfel voi afișa mesajul ”Cuvânt deja găsit”. Și nu în ultimul rând, dacă niciuna din condițiile de mai sus nu este îndeplinită, atunci numărul de încercări va scădea și un mesajul ”Cuvântul nu există în tabelă”.

În final, vom vedea dacă lista ”words” este goală (înseamnă că am găsit toate cuvintele), și astfel vom afișa ”Win”. În caz contrar, dacă numărul de încercări este mai mic sau egal cu 0, înseamnă că nu mai aveți posibilitatea de a introduce niciun cuvânt, și astfel se va afișa ”Game Over”.

NOTĂ: Numărul de încercări va scădea doar dacă a fost introdus un cuvânt care nu se află în tabelă.

PS: Nu am reușit să colorez în tabelă cuvintele găsite.

**MVC**

Aceasta este clasa ”Main” pentru acest proiect. Aici am creat un controller de *WordSearch* și unul de *Input*.

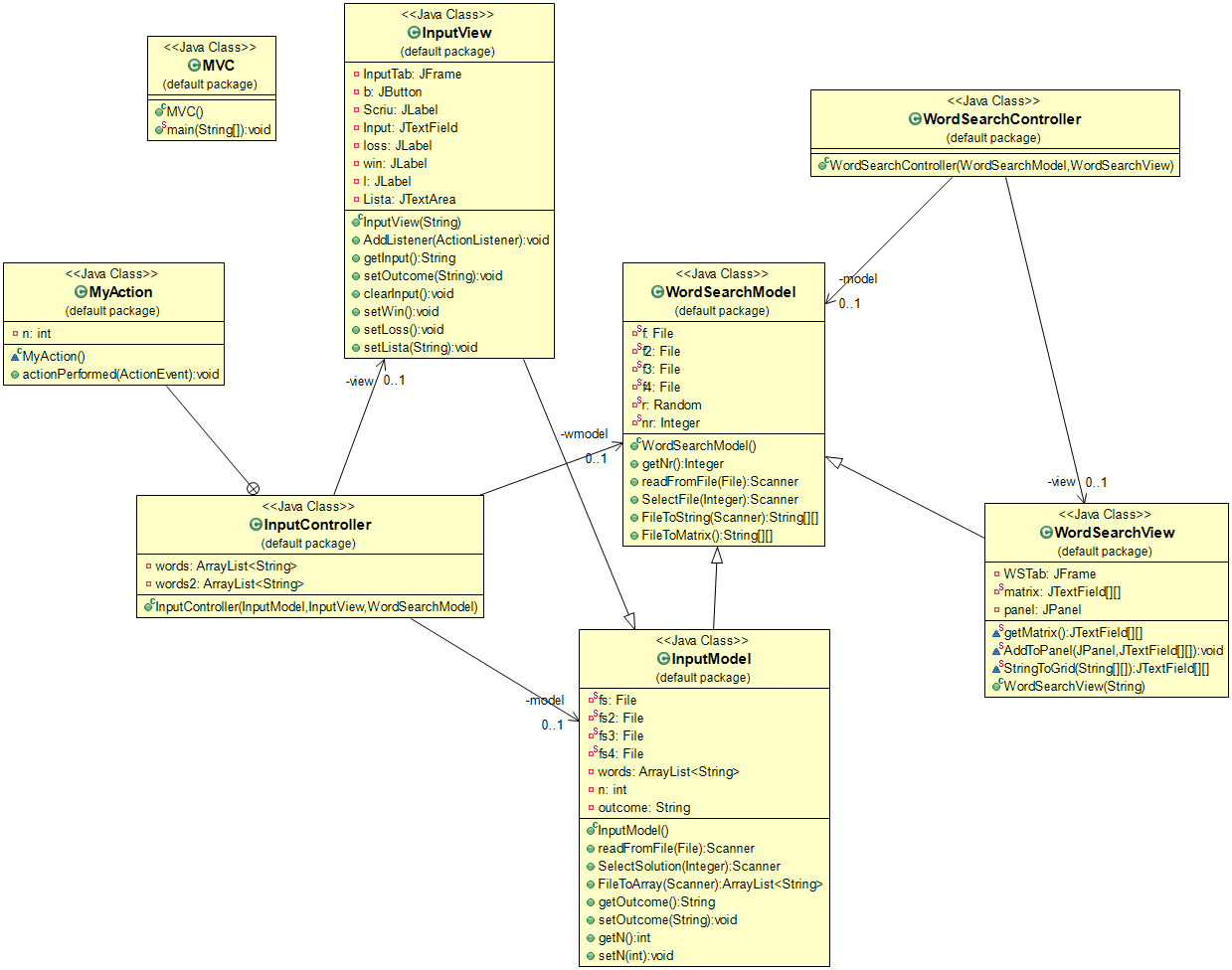


Diagrama UML a proiectului