Algorithme du Jeu Takuzu

KHOULOUD BEN SALEM
&
ABIR BEL HAJ YOUSSEF

31/07/2018

La partie console

Elle contient :

La classe Main qui contient la méthode principale main, la classe Takuzu qui comporte toutes les méthodes nécessaires pour manipuler un objet Takuzu, une interface Serializable et des classes Exceptions.

La classe Main :

Définition main :

taille : entier
takuzu=nile : Takuzu
date_debut, date_fin : Date
reponse : caractère

Ecrire (« donner la taille de la grille »), Lire (taille)

si (est_vide(le fichier takuzu.serial)= faux) alors Ecrire(« Voulez-vous continuer la partie enregistrée ? O/N »), Lire(reponse)
avec O :oui ; N :non

si reponse = 'O' alors charger la partie enregistrée et
date_debut=charger_temps(taille)

finSi

si reponse='N' alors générer une nouvelle partie

finSi

=> dans les deux cas, on initialise l'objet takuzu et date_debut

Tant que (takuzu.estValide () = faux ET existe_moins_un () = vrai)

Faire

Ecrire(« Voulez-vous continuer ou enregistrer cette partie ? C/E »)

avec C :continuer ; E :enregistrer

Lire(rep)

i , j , val : entier ; i :abscisse, j : ordonnée, val : 0 ou 1

Si (rep = 'C') alors lire (i,j,val)

Si (i>taille OU j>taille OU i<0 OU j<0 OU grille[i][j] !=-1 OU (val !=0 && val != 1)) alors

jeter une exception IndiceException

sinon

grille[i][j]=val

afficher la grille

finSi

Sinon si (rep='E') alors quitter la boucle Tant que

finSi

Fin Tant que

Si (rep='C' ET takuzu.estValide()=vrai) alors

Date_fin=Date()

Ecrire(« Félicitations ! Vous avez terminé en » + calcul_durée(date_debut,date_fin))

takuzu.enreg_meilleurs_temps(taille)

Ecrire(« Meilleur Score » + min_temps(taille))

Sinon si (rep='C' ET takuzu.estValide() = faux) alors

Date_fin=Date()

Ecrire(« Temps : » + calcul_durée(date_debut,date_fin))

Ecrire (« Dommage ! Grille invalide »)

Sinon si (rep='E ') alors

Date_fin=Date()

calcul_temps(date_debut,date_fin)

enreg_temps(taille)

enregistrer(takuzu)

finSi

Si (reponse='O' ET takuzu.estValide() = vrai) alors

Suppr_fichier(taille)

finSi

fin main

La classe Takuzu :

Elle implémente l'interface Serializable.

taille : attribut privé de type entier

temps : attribut privé de type entier

grille : matrice d'entiers : attribut privé

nbre_meilleur_score_4, nbre_meilleur_score_6, nbre_meilleur_score_8 : attributs privés de type entier

Définition constructeur par défaut :

```
Ecrire (« Donner la taille de grille »)
Lire(taille)
Tant que (taille !=4 ET taille !=6 ET taille !=8)
    Ecrire (« Donner la taille de grille »)
    Lire(taille)
```

```
 Fin Tant que
grille = new int [taille][taille]
temps = 0
```

fin constructeur

Def generate () : génére une grille aléatoirement et elle retourne un objet Takuzu

```
Random rand = new Random()
nb = 0 : entier : nombre de cases à remplir au maximum
l, c, val :entier ; l :abscisse, c :ordonnée, val :valeur
 initialiser la grille par -1 :
pour i de 1 à taille
       pour j de 1 à taille
              grille[i][j] = -1
       finpour
finpour
Tant que (nb < taille)
Faire
   répéter
              l = indice aléatoire < taille
              c = indice aléatoire < taille
              val = valeur aléatoire soit 0 soit 1
        Si(controle()=vrai ET pas_plus_deux_chiff_ident()=vrai)
        Alors
                     grille [l][c] = val
        finSi

   jusqu'à (pas_plus_deux_chiff_ident()=faux ET
controle()=faux)

        nb=nb+1
 FinTantque

retourner Takuzu
```

Fin generate

pas_plus_deux_chiff_ident() : vérifie s'il n'existe pas plus de deux chiffres identiques consécutifs.

Contrôle () : contrôle la génération aléatoire de la grille

enreg_temps(taille) : une méthode qui permet d'enregistrer le temps de la partie enregistrée dans un fichier txt en ouvrant un flux d'écriture.

charger_temps(taille) : une méthode qui permet de charger le temps de la partie chargée en lisant le fichier associé.

Définition enreg_meilleurs_temps(taille) : permet d'enregistrer les dix meilleurs scores

```
max = max_temps(taille) : entier
```

```
        fileWriter =un flux d'ecriture de type Writer
     selon (taille) :
       cas 4 : fileWriter = new FileWriter("les_dix_meilleurs4x4.txt")
       cas 6 : fileWriter = new FileWriter("les_dix_meilleurs6x6.txt")
       cas 8 : fileWriter = new FileWriter("les_dix_meilleurs8x8.txt")
    finSelon
                    si (nbre_meilleur_score < 10) alors
                      Ecrire dans fileWriter (temps)
                    finSi
                    si (nbre_meilleur_score = 10 ET temps < max) alors
                          supprimer(max,taille)
                          nbre_meilleur_score = 10
                    finSi
       fermer fileWriter
      delete_file(taille)
fin enreg_meilleurs_temps
```

max_temps(taille) : une méthode permet de retourner le score maximum

Supprimer(max,taille) : une méthode qui permet de supprimer le score maximum afin d'ajouter un autre plus petit.

Delete_file(taille) : une méthode permet de vider le fichier "les_dix.txt" et le remplacer par "les_dix_mailleurs.txt"

Définition calcul_duree(date_debut,date_fin) : calcule la difference entre les deux dates pour retourner la durée du jeu :

```
        h :nbre d'heures, min :nbre de minutes, sec :nbre de secondes
        res = chaine de caractères :résultat à retourner

        h = heures(date_fin)-heures(date_debut) // nbre des heures
        min = minutes(date_fin)-minutes(date_debut) // nbre des minutes
        sec = secondes(date_fin)-secondes(date_debut) // nbre des secondes
             si (h < 0) alors
                    h = 0
                    min = 60 - min
             finSi
             si (sec < 0) alors
                    min = min - 1
                    sec = sec + 60
             finSi
             si (min < 0) alors
                    min = min + 60
                    h = h - 1
             finSi
             temps = temps + h * 3600 + min * 60 + sec
             h = temps / 3600
             min = (temps % 3600) / 60
             sec = (temps % 3600) % 60
             res = h + " h " + min + " min " + sec + " s"
             si (h = 0) alors
                    res = min + " min " + sec + " s"
             finSi
             si (min == 0 ET h == 0) alors
                    res = sec + " s"
```

```
            finSi
            retourner res

Fin calcul_duree


Definition estValide() :
   si ( pas_plus_deux_chiff_ident() ET lignes_differents() ET
colonnes_differents() ET Compare() ET verif_valeur() )
   alors
            retourner vrai
   sinon
            retourner faux
Fin estValide
```

verif_valeur() : cette méthode parcourt la matrice et tant qu'il existe une valeur différent de 0 et de 1, on retourne faux sinon on retourne vrai.

```
Definition lignes_differents() :
            res = faux  // resultat à retourner
            c = 0 : entier // nbre de colonnes
            i,j,k : entier
            pour (i de 1 à taille)
            faire
                  tab1 = grille[i] // la i-éme ligne
                  pour (k de (i + 1) à taille)
                   Faire
                        tab2 = grille[k] // la k-éme ligne
                        pour (j de 1 à taille)
                        Faire
                              si (tab1[j] = tab2[j]) alors
                                    c++
                              finSi
                        FinPour
                        si (c = taille) //c=taille càd la i-eme et le k-
eme ligne sont égaux
                        alors
                              retourner faux
                        finSi
                        c = 0
                  finPour
            finPour
            retourner res
Fin lignes_differents



Definition colonnes_differents() :
            res = vrai :resultat à retourner
            c = 0 :entier //nbre de lignes
            i,j,k :entier
            pour (j de 1 à taille )
            Faire
                  pour (k de (j+1) à taille)
                   Faire
                        pour ( i de 1 taille)
                        Faire
```

```
                                    si (grille[i][j]=grille[i][k]) alors
                                          c++
                                    finSi
                              finPour
                              si (c = taille) alors
                                    retourner faux
                              finSi
                              c = 0
                        finPour
                  finPour

                  retourner res
Fin colonnes_differents



Définition Compare() :
                  i :entier
                  pour ( i de 1 à taille) // faux sinon
                  faire
                        si (Non(Compare_Ligne(grille, i))) alors
                              retourner faux
                        finSi
                        si (Non(Compare_Colonne(grille, i)))  alors
                              retourner faux
                        finSi
                  finPour
                  retourner vrai
fin Compare
```

Compare_Ligne(grille,i) :une méthode qui retourne vrai si le nombre des 0 est égal au celui des 1 sur la ième ligne , faux sinon

Compare_Colonne(grille,i) : une méthode qui retourne vrai si le nombre des 0 est égal au celui des 1 sur la ième colonne , faux sinon

Fin de la classe Takuzu

La partie interface graphique

Elle contient :

Une classe principale MainFrame, une classe pour chaque interface Takuzu (selon la taille de la grille), une classe MyDefaultTableModel qui rendre les valeurs remplies préalablement non éditables et une classe XTableRenderer qui permet de centrer les valeurs dans la grille.

La classe MainFrame :

Elle contient un menu nommé Aide et un sous menu nommé Règles du jeu qui expliquent le jeu en citant les règles du jeu Takuzu.

Elle contient aussi un message « Donner la taille de la grille », un champ pour saisir la taille et un bouton « commencer » qui, selon la taille, nous envoie à une autre page parmi les suivants :

La classe Interface4x4 && La classe Interface6x6 && La classe Interface8x8 :

Elles possèdent les mêmes composants graphiques :

Une barre de menu, une matrice, un chronomètre, un bouton « Recommencer » et un bouton « Verifier ».