Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-07-5-gagne-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Pion de la couleur du joueur. Nouvelle méthode. Grille ligne. Valeur de retour de ma méthode. Boucle principale. Couleur du joueur. Condition d'arrêt. Ligne lignedepart colonnedepart. Méthode estcegagne. Nombre de pions. Type booléen. Besoin de la grille. Pions rouges. Couleur de la case. Condition d'arrêt de la boucle.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en Java).)

vers la vidéo

EPFL

Puissance 4 : méthodes `estCeGagne` et `compte`

(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Dans la vidéo précédente, vous aviez vu comment écrire

notes

résumé

0m 1s

Retour sur la méthode `main`

```
int couleurJoueur = JAUNE;
do {
  demandeEtJoue(grille, couleurJoueur);

  affiche(grille);



  // on change la couleur pour la couleur de l'autre joueur:
  if (couleurJoueur == JAUNE) {
    couleurJoueur = ROUGE;
  } else {
    couleurJoueur = JAUNE;
  }
} while(    );
```

la boucle principale qui permet aux joueurs de jouer chacun leur tour, alors il manquait juste la condition d'arrêt,

notes

résumé

0m 6s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur
```

c'est-à-dire qu'il manquait ce qu'il faut écrire ici pour que la boucle s'arrête quand un des deux joueurs a gagné ou quand la grille est pleine. Alors pour l'instant, nous allons nous concentrer sur le cas où l'un des deux joueurs a gagné et pour cela je vais introduire une nouvelle méthode que je vais appeler estCeGagne et que je vais appeler ainsi : je vais mettre la valeur de retour de ma méthode estCeGagne dans une variable que je vais appeler gagne qui sera de type booléen. Ma méthode estCeGagne a besoin de la grille pour pouvoir fonctionner et de la couleur du joueur qui vient de jouer. Alors il faut que je déclare ma variable gagne, et il faut que je la déclare avant la boucle puisque je vais m'en servir dans la condition d'arrêt de la boucle. Elle est donc de type booléen et je peux maintenant m'en servir pour définir la condition d'arrêt, et on va répéter la boucle tant que le joueur n'a pas gagné. Il nous faut donc écrire cette méthode estCeGagne, ce qui est sans doute la partie la plus difficile de notre programme. Alors il y a plusieurs façons d'écrire cette méthode estCeGagne et nous vous proposons la stratégie suivante. Nous allons parcourir la grille ligne par ligne et colonne par colonne jusqu'à trouver un pion de la couleur du joueur qui vient de jouer, disons par exemple que c'est le joueur rouge qui vient de jouer, et que l'on trouve ce pion-ci. Nous allons partir de ce pion et parcourir la grille dans plusieurs directions, horizontalement, verticalement et en diagonale, pour compter combien il y a de pions à partir de ce pion-ci qui sont de la même couleur. Si par exemple, on considère cette direction-ci, on arrive à compter un, deux, trois, quatre pions rouges et donc on sait que le joueur

notes

résumé

0m 14s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur
```

a gagné. Si en revanche, on a parcouru toute la grille et qu'on a toujours compté moins que quatre pions, on sait que le joueur n'a pas gagné. On peut en fait remarquer qu'on est pas obligé de considérer toutes les directions et en fait nous allons nous contenter de compter les pions dans quatre directions seulement, et nous allons choisir par exemple, vers le haut et la droite, vers la droite, vers le bas et la droite et vers le bas. Alors pourquoi on peut se contenter de considérer ces quatre directions ? J'ai dit que nous allions parcourir la grille ligne par ligne et colonne par colonne, supposons encore une fois que c'est le joueur rouge qui vient de jouer, le premier pion que nous allons trouver est celui-ci. Nous allons compter tout d'abord les pions dans cette direction-ci, nous n'allons compter qu'un seul pion, celui-ci, même chose pour cette direction-ci et cette direction-ci, il n'y a qu'un seul pion dans ces trois directions. Dans cette direction-ci, il y a deux pions, ce n'est pas suffisant pour gagner, et nous allons continuer à parcourir la grille. Le prochain pion rouge que nous allons trouver est celui-ci. Dans cette direction-ci et cette direction-ci nous n'allons compter que deux pions, ce n'est pas suffisant pour gagner. Dans cette direction-ci et cette direction-ci, nous n'allons compter qu'un seul pion, ce n'est toujours pas suffisant pour gagner. Donc, nous allons continuer à parcourir la grille jusqu'à atteindre ce pion-ci, nous allons compter les pions dans cette direction-ci et nous allons compter cette fois un, deux, trois, quatre pions et on sait que le joueur rouge a gagné. Pour prouver que le joueur rouge vient d'aligner ces quatre pions, on peut donc soit partir de ce pion-ci, et aller dans cette direction-ci, soit partir de ce pion-là et aller dans cette direction-là. On n'est donc

notes

résumé

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur
```

pas obligé de considérer ces deux directions, on peut se contenter d'une seule, et c'est la même chose pour les six autres directions. On va regarder uniquement dans cette direction-ci et pas dans cette direction-ci. Dans cette direction-ci et pas celle-ci, dans cette direction-ci et pas celle-ci. Il ne nous reste donc que quatre directions à considérer. Il nous faut donc maintenant coder cette stratégie en écrivant la méthode estCeGagne, or je vous rappelle qu'on va appeler la méthode estCeGagne par exemple de la façon suivante. C'est-à-dire qu'on va mettre la valeur de retour de la méthode estCeGagne dans une variable de type booléen, et que la méthode va attendre une grille,

notes

résumé

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur);
static boolean estCeGagne(int[][] grille, int couleurJoueur)
{
```

et la couleur du joueur qui vient de jouer en paramètres. L'en-tête de la méthode sera donc tout simplement celui-ci.

notes

résumé

4m 49s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur);
static boolean estCeGagne(int[][] grille, int couleurJoueur)
{
  for(int ligne = 0; |
```

D'autres stratégies consistent à parcourir la grille,

notes

résumé

4m 59s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur);
static boolean estCeGagne(int[][] grille, int couleurJoueur)
{
  for(int ligne = 0; ligne < grille.length; ++ligne) {
    for(int colonne = 0; colonne < grille[ligne].length; ++colonne) {
      int couleurCase = grille[ligne][colonne];

      if (couleurCase == couleurJoueur) {
        if (
          // en diagonale, vers le haut et l
```

ce qu'on va faire à l'aide de deux boucles for et comme on a besoin des coordonnées du pion, on va utiliser deux boucles for classiques qui vont s'écrire de la façon suivante. Les variables ligne et colonne vont contenir les coordonnées du pion à partir duquel on va compter le nombre de pions alignés, et pour simplifier un peu l'écriture du code, je vais introduire une variable locale que je vais appeler couleurCase pour contenir la couleur de ce pion, qui est grille[ligne][colonne]. Si la couleur de la case est la même que celle du joueur qui vient de jouer, alors il faut que je compte les pions dans les quatre directions qui nous intéressent. Alors, comme c'est un peu compliqué de compter les pions,

notes

résumé

5m 3s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur);
static boolean estCeGagne(int[][] grille, int couleurJoueur)
{
  for(int ligne = 0; ligne < grille.length; ++ligne) {
    for(int colonne = 0; colonne < grille[ligne].length; ++colonne) {
      int couleurCase = grille[ligne][colonne];

      if (couleurCase == couleurJoueur) {
        if (
          // en diagonale, vers le haut et la droite:
          (compte(grille, ligne, colonne,
```

et bien, je vais introduire une méthode que je vais simplement appeler compte et qui va faire cette tâche. De quels arguments la méthode compte a-t-elle besoin ? Donc tout d'abord bien sûr, de la grille, des coordonnées du pion à partir desquelles on veut compter le nombre de pions alignés, et ces coordonnées sont contenues dans les variables ligne et colonne, et de la direction dans laquelle on veut compter le nombre de pions alignés. Alors comment peut-on définir cette direction ? Supposons par exemple que je veuille compter les pions

notes

résumé

5m 54s

Méthode estCeGagne

```
// gagne = estCeGagne(grille, couleurJoueur);
static boolean estCeGagne(int[][] grille, int couleurJoueur)
{
  for(int ligne = 0; ligne < grille.length; ++ligne) {
    for(int colonne = 0; colonne < grille[ligne].length; ++colonne) {
      int couleurCase = grille[ligne][colonne];

      if (couleurCase == couleurJoueur) {
        if (
          // en diagonale, vers le haut et la droite:
          (compte(grille, ligne, colonne, -1, +1) >= 4) ||
          // horizontalement, vers la droite:
          (compte(grille, ligne, colonne, 0, +1) >= 4) ||
          // en diagonale, vers le bas et la droite:
          (compte(grille, ligne, colonne, +1, +1) >= 4) ||
          // verticalement, vers le bas:
          (compte
```

en diagonale vers le haut et vers la droite, alors dans ce cas, pour passer d'une case à l'autre, il va falloir soustraire un à la ligne et ajouter un à la colonne, et donc, je vais utiliser ces deux valeurs-là, ce moins un et ce plus un, en argument de ma fonction compte pour dire que je veux aller vers le haut et vers la droite. Je vais faire envoyer à ma méthode compte le nombre de pions alignés et si ce nombre de pions est supérieur ou égal à quatre, alors je sais que le joueur a gagné. Alors pourquoi supérieur ou égal à quatre, et non pas simplement égal à quatre ? Et bien, c'est tout simplement qu'on peut gagner en alignant plus de quatre pions. Alors évidemment, il faut que je considère les trois autres directions et ça va s'écrire de la façon suivante. Pour compter les pions horizontalement, je ne vais pas changer la ligne, c'est pour ça que j'utilise zéro en argument ici, par contre la colonne va changer et je vais passer à la colonne suivante, c'est pour ça que j'utilise la valeur plus un en dernier argument. Ensuite, pour compter les pions en diagonale vers le bas et la droite, pour passer d'une case à l'autre, je vais ajouter un à la ligne, c'est pour ça que j'utilise la valeur plus un ici, et j'ajoute également un à la colonne, et c'est pour ça que j'utilise la valeur plus un ici. Enfin il nous reste une dernière direction.

notes

résumé

6m 25s

Méthode compte

```
//
```

Pour compter les pions verticalement vers le bas, je vais passer d'une case à l'autre, en ajoutant un à la ligne, c'est pour ça que j'ai la valeur plus un ici, par contre la colonne ne va pas changer et j'utilise la valeur zéro ici. Si l'une de ces quatre conditions est vraie, alors je sais que le joueur a gagné. Alors il suffit bien qu'une des quatre conditions soient vraies, c'est pour ça que j'ai utilisé un or ici et dans ce cas-là, je vais faire renvoyer tout à ma méthode, je vais fermer les accolades que j'avais ouvertes précédemment, et si j'arrive à ce stade de ma méthode, c'est que je ne suis jamais passé par ici, c'est-à-dire que je n'ai jamais trouvé au moins quatre pions d'alignés, et je sais que dans ce cas-là, le joueur n'a pas encore gagné. Donc je vais faire renvoyer false à ma méthode, et en bon programmeur, je vais ajouter des commentaires pour expliquer les points un peu difficiles du corps de ma méthode.

notes

résumé

8m 1s

Méthode compte

```
// if (compte(grille, ligne, colonne, -1,
```

Évidemment, il faut encore écrire la méthode compte. Alors je vous rappelle qu'on va appeler la méthode compte par exemple de cette façon-ci,

notes

résumé

9m 12s

Méthode compte

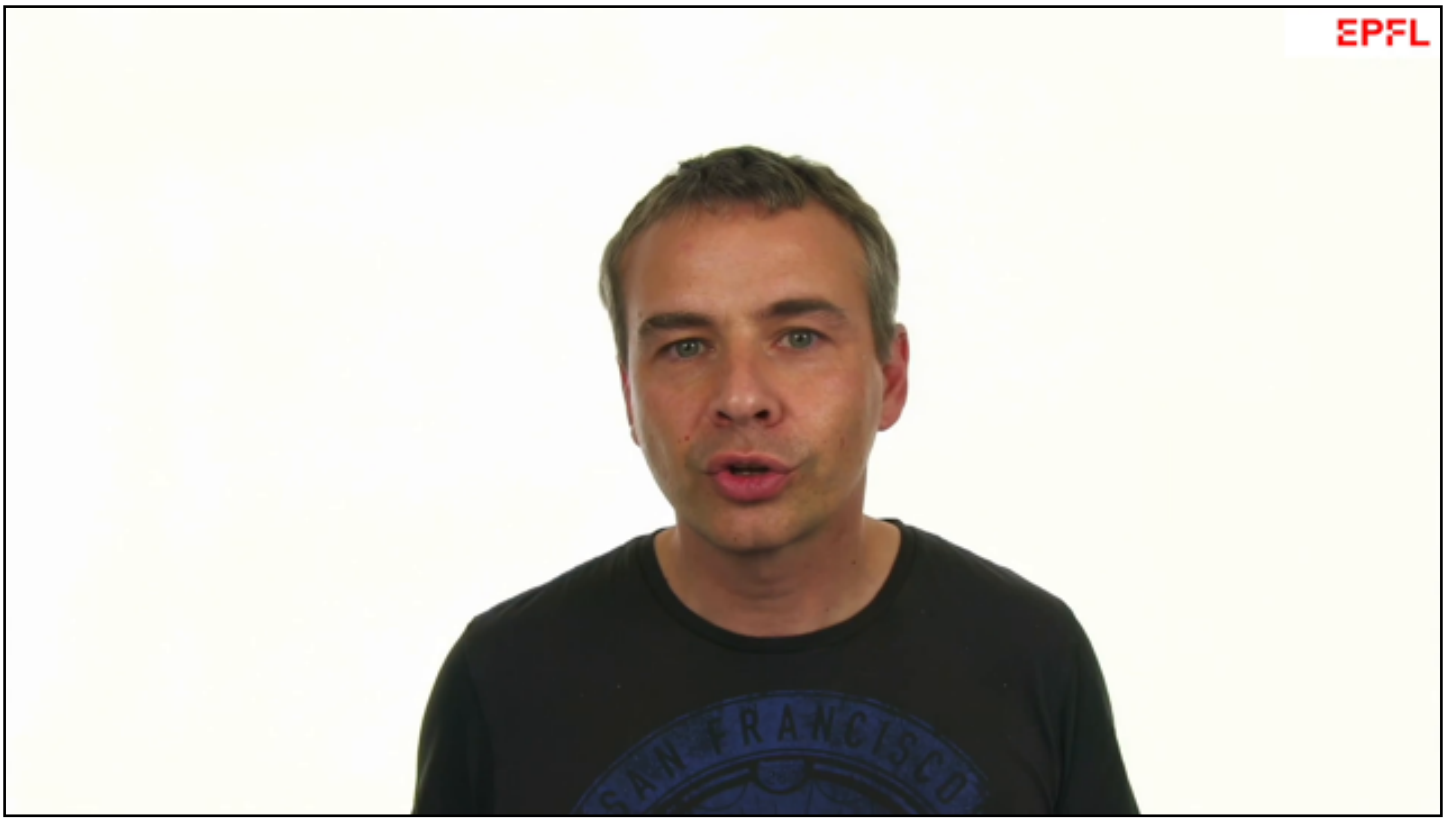
```
// if (compte(grille, ligne, colonne, -1, +1) >= 4 ...
static int compte(int[][] grille,
                  int ligneDepart, int colonneDepart,
                  int dirLigne, int dirColonne)
{
  int compteur = 0;

  int ligne = ligne
```

c'est-à-dire que la méthode va renvoyer un nombre de pions, c'est pour ça que je vais utiliser le type int pour définir le type de la valeur de résultat de la méthode. Le premier paramètre est donc la grille, les deux paramètres suivants sont la ligne et la colonne du pion à partir duquel on va compter, ils sont tous les deux de type int, et les deux derniers paramètres sont les directions pour la ligne et la colonne que je vais appeler dirLigne et dirColonne puisqu'ils sont de même nature, et que l'un fait référence à la ligne et l'autre à la colonne. Je vais déclarer une variable compteur initialisée à zéro pour compter le nombre de pions alignés. Que doit faire la méthode compte exactement ? Et bien, elle doit partir de la ligne ligneDepart colonneDepart et elle doit parcourir la grille dans la direction donnée en partant de ligneDepart colonneDepart tant qu'on trouve des pions qui sont de la même couleur que le pion qui est en ligneDepart colonneDepart. Alors comment on va faire ça ? Et bien, on va utiliser deux variables que je vais appeler ligne et colonne, qu'on va initialiser respectivement à ligneDepart et colonneDepart, et pour passer d'une case à l'autre, à chaque tour de boucle, on va ajouter à ligne la valeur de dirLigne et à colonne la valeur de dirColonne. Comme on va parcourir la grille tant qu'on trouve des pions de la même couleur que le pion qui est en ligneDepart colonneDepart, et bien on va utiliser une boucle while, et à chaque tour de la boucle while, on aura trouvé un nouveau pion de la même couleur et on va incrémenter la variable compteur. Comment tout cela va-t-il s'écrire ?

notes

résumé

9m 19s



Et bien, on va commencer par déclarer les variables ligne et colonne, initialisées à ligneDepart et colonneDepart, on va donc utiliser une boucle while dont la condition d'arrêt dépend de la couleur du pion en ligne colonne, et on va continuer tant que la couleur de ce pion est la même que la couleur du pion qui est en ligneDepart colonneDepart. À chaque tour de boucle, on va incrémenter la variable compteur, puisqu'on a trouvé un nouveau pion de la même couleur que le premier, et puis on va ajouter dirLigne et dirColonne à ligne et colonne. Notez au passage que j'ai ajouté des espaces ici et ici pour aligner les signes égal et plus sur ces deux lignes, ça rend le code plus clair, ça met en évidence que les variables ligne et colonne sont de même nature.

notes

résumé

11m 13s

Méthode compte

```
// if (compte(grille, ligne, colonne, -1, +1) >= 4 ...
static int compte(int[][] grille,
                  int ligneDepart, int colonneDepart,
                  int dirLigne, int dirColonne)
{
  int compteur = 0;

  int ligne = ligneDepart;
  int colonne = colonneDepart;

  while (grille[ligne][colonne] == grille[ligneDepart][colonneDepart]) {

    ++compteur;
    ligne   = ligne   + dirLigne;
    colonne = colonne + dirColonne;

  }

  retu
```

Quand on sort de la boucle, c'est qu'on a trouvé un pion qui n'était pas de la même couleur, et on peut renvoyer la valeur contenue dans la variable compteur,

notes

résumé

12m 1s

sauf que notre méthode n'est pas tout à fait correcte, et peut-être que vous voyez pourquoi.

notes

résumé

12m 8s

Méthode compte

```
// if (compte(grille, ligne, colonne, -1, +1) >= 4 ...
static int compte(int[][] grille,
                  int ligneDepart, int colonneDepart,
                  int dirLigne, int dirColonne)
{
  int compteur = 0;

  int ligne = ligneDepart;
  int colonne = colonneDepart;

  while (grille[ligne][colonne] == grille[ligneDepart][colonneDepart]) {

    ++compteur;
    ligne   = ligne   + dirLigne;
    colonne = colonne + dirColonne;

  }

  return compteur;
}
```

Comme pour la méthode joue, il se peut très bien qu'on sorte de la grille, c'est-à-dire que les variables ligne ou colonne peuvent prendre des valeurs qui sont interdites. Alors, pour corriger ça, je vais ajouter plusieurs tests

notes

résumé

12m 14s

à la condition de la boucle while, je vais vérifier que la ligne est bien supérieure ou égale à zéro, et qu'elle est inférieure au nombre de lignes, et je vais ajouter les mêmes tests pour la colonne. Et finalement, je vais ajouter un commentaire pour décrire ce que fait la boucle. pour décrire ce que fait la boucle.

notes

résumé

12m 27s