Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-07-4-main-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Méthode main. Boucle de jeu. Couleur du joueur. Numéro de colonne de la part. Couleur du premier joueur. Variable couleur du joueur. Joueur jaune. Vidéos précédentes. Petite méthode de test. Variable valide. Parties communes. Corps de la méthode main. Troisième étape de notre boucle. Message suivant. Boucle while.

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

vers la vidéo

EPFL

Puissance 4 : moteur de jeu
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

0m 0s

Continuons donc la programmation de notre fameux jeu puissance 4.

notes

résumé

0m 1s

Jouons !

```
public static void main(String[] args)
{
  int[][] grille = new i
}
```

A ce stade nous savons représenter le jeu, nous savons représenter les couleurs, nous savons initialiser le jeu, l'afficher, et puis nous avons vu comment faire le cœur du programme qui est donc cette méthode joue qui permet de recevoir un numéro de colonne de la part d'un joueur, vérifier si l'on peut effectivement jouer dans cette colonne, et si l'on peut jouer le coup. Mais à ce stade-là, nous ne savons toujours pas vraiment jouer au jeu. Donc essayons d'écrire une méthode main qui va effectivement faire que le jeu se déroule. Pour cela, ce qu'il va falloir faire c'est bien sûr demander un coup à un joueur, jouer ce coup, afficher le jeu, et puis alterner, changer la couleur du joueur. Bien sûr, tout ceci, nous voulons le répéter tour à tour tant que la partie n'est pas finie. Nous voici donc au clair sur ce que doit faire la méthode, sur le « quoi ». Comme toujours, quand on écrit une méthode, on doit d'abord se préoccuper de ce qu'elle doit faire, du « quoi ». Ceci étant clair, on peut maintenant commencer à écrire et on commence bien sûr par l'en-tête classique et puis le corps de la méthode main. Avant d'entrer dans cette boucle de jeu que l'on vient de décrire, il faut bien sûr déclarer un jeu. Donc ici c'est un tableau de deux entiers, qu'on appelle grille.

notes

résumé

0m 6s

Jouons !

```
public static void main(String[] args)
{
  int[][] grille = new int[6][7];

  initialise(grille);
  affiche(grille);

  int couleurJoueur = JAUNE;
  do {
    if (co
}
```

Et l'on sait qu'une grille de puissance 4 a six lignes et sept colonnes. Ensuite on va bien sûr l'initialiser. Donc on appelle notre méthode initialise. Et il est bon de l'afficher pour montrer qu'on démarre avec la grille vide. Voilà on va ensuite fixer la couleur du premier joueur. Donc au puissance 4, on commence avec un joueur jaune ici. Donc on initialise une variable couleur du joueur courant à jaune. Voilà, nous sommes prêts maintenant pour démarrer notre boucle de jeu dont on parlait tout à l'heure. La question qu'on se pose ici c'est : est-ce que c'est une boucle while ou une boucle do while ? Est-ce qu'on commence par tester quelque chose ? Ou est-ce qu'on commence par exécuter quelque chose, avant de faire le test et de boucler ? Ici, je ne vois pas très bien ce qu'on pourrait tester. On est dans la situation initiale, on contrôle tout. Donc on va commencer par faire notre séquence de jeu, puis ensuite on testera si on doit continuer, si le jeu se termine, et sinon, bien sûr, donc on bouclera. C'est donc une boucle do while. Et on commence donc par écrire do. Et donc ce que l'on veut faire en premier dans notre boucle c'est de demander un coup au joueur. Pour cela, on pourrait bien sûr en faire une fonction, mais ici je vais tout vous écrire dans le main et l'on reviendra sur cela un peu plus tard. Ceci dit, même si on n'en fait pas ici une fonction, on peut adopter la même méthodologie et décomposer donc en sous-tâches : pour demander un coup, qu'est-ce qu'il faut faire ? Il faut poser la question et récupérer la réponse. Donc on commence par poser la question.

notes

résumé

1m 26s

Jouons !

```
public static void main(String[] args)
{
  int[][] grille = new int[6][7];

  initialise(grille);
  affiche(grille);

  int couleurJoueur = JAUNE;
  do {
    if (couleurJoueur == JAUNE) {
      System.out.println("Joueur
}
```

Pour cela, on va afficher un message au joueur et on va afficher un message différent si le joueur est rouge ou s'il est jaune puisqu'on va en profiter pour lui indiquer sa couleur. Donc si la couleur du joueur est jaune, alors on affiche le message suivant : Bon bien sûr, system point out point println

notes

résumé

2m 49s

Jouons !

```
public static void main(String[] args)
{
  int[][] grille = new int[6][7];

  initialise(grille);
  affiche(grille);

  int couleurJoueur = JAUNE;
  do {
    if (couleurJoueur == JAUNE) {
      System.out.println("Joueur X :  entrez un numéro de colonne");
    } else {
      System.out.println("Joueur O :  entrez un numéro de colonne");
    }

    int colonne = clavier.nextInt();
    // les indices des tableaux com
}
```

Ici, joueur et on affiche sa couleur donc ici on va mettre un X pour la couleur jaune, joueur X, et on lui demande donc d'entrer un coup. Sinon, on affiche cette fois-ci un message pour le joueur rouge On utilise ici la lettre O capitale pour afficher, comme pour les pions, la couleur rouge disons. Et on termine avec le même message. Ensuite on veut saisir la réponse. Donc pour cela, on introduit une nouvelle variable colonne qui sera la colonne dans laquelle le joueur veut jouer. Et on lit avec nextInt. Bien sûr, il ne faut pas oublier de déclarer de quoi pouvoir lire, donc ici le Scanner avec la déclaration usuelle. Et en suivant les conseils qu'on vous a donné dans des vidéos précédentes, on déclare cette variable hors du main ici pour pouvoir en profiter dans l'ensemble du programme. Voilà, décalons tout ceci un petit peu vers le haut pour continuer donc dans notre main et ne pas oublier que donc les indices qu'on utilise en java dans le tableau

notes

résumé

3m 7s

sont décalés de un par rapport aux indices que l'on utilise avec les humains. Les humains notent de un à six, alors que nous, nous allons commencer en zéro. Donc il faut décaler de un avec un « moins moins colonne ». Puis ensuite on va pouvoir effectivement jouer ce coup. Donc je vous rappelle que la méthode joue, que nous avions écrite dans la séquence vidéo précédente, renvoie un booléen qui indique vrai si le coup est valide et faux si le coup n'est pas valide. Donc on récupère ici cette valeur de retour dans un booléen, dans une variable valide que l'on a ici initialisé et déclaré avec donc l'appel à notre méthode joue qui prend donc la grille, le numéro de colonne que l'on vient de recevoir depuis le joueur et la couleur du joueur courant que l'on avait donc déclaré ici. Si le coup n'est pas valide, il faut quand même prévenir le joueur et donc si le coup n'est pas valide, on affiche un message indiquant que le coup n'est pas valide : donc println le message. Puis troisième étape de notre boucle : afficher le jeu. Et ensuite, quatrième étape de... on décale un peu tout ça pour qu'on y voit clair ...quatrième étape de notre boucle : alterner les couleurs, donc changer la couleur du joueur courant. Comme d'habitude, on met des commentaires pour être le plus clair possible. Donc comment changer la couleur : tout simplement.

notes

résumé

4m 13s

Jouons !

```
  do {
    if (couleurJoueur == JAUNE) {
      System.out.println("Joueur X :  entrez un numéro de colonne");
    } else {
      System.out.println("Joueur O :  entrez un numéro de colonne");
    }

    int colonne = clavier.nextInt();
    // les indices des tableaux commencent par 0 en Java:
    --colonne;

    boolean valide = joue(grille, colonne, couleurJoueur);
    if (!valide) {
      System.out.println(" > Ce coup n'est pas valide");
    }

    affiche(grille);
    // on change la couleur pour la couleur de l'autre joueur:
    if (couleurJou
}
```

On pourrait aussi écrire une fonction, bien sûr ce serait une bonne occasion, mais la méthode c'est simplement : si on est couleur jaune, alors on change et on met couleur rouge. Et sinon, on met couleur jaune. Donc if la couleur du joueur courant est jaune,

notes

résumé

5m 33s

on la met en rouge. Donc si la couleur du joueur est jaune, je la mets en rouge. Et sinon, je la mets en jaune. couleur_joueur, je vous rappelle, est la variable qui représente la couleur du joueur courant. Voilà, on a donc terminé notre boucle et il n'y a plus qu'à ici mettre la condition. La condition de quoi ? La condition de continuation du jeu. Donc on continue à jouer tant que... Et bien, je laisse cela pour la séquence vidéo suivante et laisse ici un blanc.

notes

résumé

5m 49s

Jouons !

```
  do {
    if (couleurJoueur == JAUNE) {
      System.out.println("Joueur X :  entrez un numéro de colonne");
    } else {
      System.out.println("Joueur O :  entrez un numéro de colonne");
    }

    int colonne = clavier.nextInt();
    // les indices des tableaux commencent par 0 en Java:
    --colonne;

    boolean valide = joue(grille, colonne, couleurJoueur);
    if (!valide) {
      System.out.println(" > Ce coup n'est pas valide");
    }

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

Voilà, ceci termine notre méthode main qui nous permet de jouer. Je vous encourage à compiler et à tester. Bien sûr, il faut mettre quelque chose ici dans le blanc. Dans un premier temps, vous pouvez mettre while true qui va faire une boucle infinie, puis vous arrêtez le programme avec l'environnement. Mais je vous encourage donc systématiquement, quand on est à une étape, bien sûr de tester comme ceci. Pour ma part, je préférerais maintenant reprendre ce code avec vous afin de mieux l'organiser, le modulariser,

notes

résumé

6m 22s

c'est-à-dire faire des méthodes des parties les plus importantes. On a déjà dit qu'on pourrait faire une méthode ici d'affichage, on va revenir sur ce point tout à l'heure. On pourrait ici faire une méthode qui alterne les joueurs. Et puis, de façon plus globale, tout ceci on pourrait, en fait, en faire une méthode, puisque c'est un peu le cœur du jeu. Le main est beaucoup trop gros ici, vous voyez d'ailleurs qu'il ne tient pas sur un transparent,

notes

résumé

6m 51s

Jouons !

```
  do {
    if (couleurJoueur == JAUNE) {
      System.out.println("Joueur X :  entrez un numéro de colonne");
    } else {
      System.out.println("Joueur O :  entrez un numéro de colonne");
    }

    int colonne = clavier.nextInt();
    // les indices des tableaux commencent par 0 en Java:
    --colonne;

    boolean valide = joue(grille, colonne, couleurJoueur);
    if (!valide) {
      System.out.println(" > Ce coup n'est pas valide");
    }

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

c'est un signe qu'il n'est pas bien écrit. Un main doit être très court, très clair. Et donc on va le modulariser, c'est-à-dire le découper en méthodes.

notes

résumé

7m 13s

Méthode demandeEtJoue

Donc le plus simple c'est de couper directement la partie qu'on veut mettre en méthode, comme on vous l'avait expliqué dans la vidéo sur la méthodologie pour développer des fonctions et des méthodes. Et puis commençons par écrire l'appel qu'on imagine, donc on va par exemple dire que cette partie correspond à demander et jouer, donc on va l'appeler demandeEtJoue, ce sera le nom de notre méthode, demandeEtJoue, et qu'est-ce que doit recevoir cette méthode ? Elle doit recevoir, bien sûr, le jeu donc la grille et puis la couleur du joueur courant. Voilà les deux arguments qu'on lui passe. Et nous pouvons donc maintenant passer à l'écriture de cette méthode.

notes

résumé

7m 22s

Méthode demandeEtJoue

```
// demandeEtJoue(grille, couleurJoueur);
void demandeEtJoue(i
```

Donc pour cela je vous rappelle ici en commentaires, l'appel. Donc demandeEtJoue, on passe la grille et on passe la couleur du joueur. Ce qui va donc donner l'en-tête suivant :

notes

résumé

8m 0s

Méthode demandeEtJoue

```
void demandeEtJoue(int[][] grille, int couleurJoueur)
{
  System.out.print("Joueur ");
  if (couleurJoueur == JAUNE) {
    System.out.print("X");
  } else {
    System.out.print("O");
  }
  System.out.println(" :  entrez un numéro de colonne");

  int colonne = clavier.nextInt();
  // les indices des tableaux commencent par 0 en Java:
  --colonne;

  boolean valide = joue(grille, colonne, couleurJoueur);
  if (!valide) {
    System.out.println(" > Ce coup n'est pas valide");
  }
}
```

void demandeEtJoue qui reçoit donc une grille et qui reçoit une couleur. Ensuite simplement on va, ici, dans le corps de la méthode, recopier tout ce qu'on avait couper depuis le main, pour justement en faire une méthode. A ce stade-là il faut bien faire attention de ne pas s'être trompé dans les noms. Par exemple, ici on a bien repris couleurJoueur et donc je vous encourage vraiment à sauvegarder, à compiler et à vérifier que le code est correct, qu'on n'a pas fait d'erreurs, bien sûr à le relire, et même, éventuellement, à le tester avec une petite méthode de test. Continuons dans ce même esprit de modulariser et d'éviter au maximum les copier-collers. Et moi je vois plusieurs copier-collers ici. Je vois que "Joueur" a été répété ici. Je vois que l'on a répété ici le message « entrez le numéro de colonne ». Donc tout ceci n'est pas correct et il va falloir modulariser. Pour cela, on va récupérer les parties communes et les mettre hors de la condition sur la couleur. Parce qu'au fond, quelle est la partie qui dépend de la couleur ? C'est uniquement cet affichage ici du X et du O. Donc les parties communes qui précèdent l'affichage du X, on va les mettre avant le if, et les parties communes qui sont après l'affichage du X et du O, nous allons les mettre après le if. Ce qui nous donne donc le code suivant : on commence par afficher ici le message commun Joueur espace... et ici attention on ne fait pas un println, on fait un print puisque la ligne n'est pas terminée, la ligne que l'on veut afficher. Ensuite on fait notre test sur la couleur et dans ce cas-là, si la couleur est jaune, on affiche le X sinon, on affiche le O.

notes

résumé

8m 11s

Puis on a donc ici de nouveau la partie commune qui est affichée avec un println qui affiche donc la fin du message. Et là, on voit encore une fois qu'on pourrait modulariser encore un petit peu plus et ce petit bout de code, ici, ces quelques lignes, pourrait être un affichage de la couleur qui pourrait être utile à la fois dans l'affichage du jeu et dans l'affichage du joueur comme ici.

notes

résumé

10m 1s

Méthode demandeEtJoue

```
void demandeEtJoue(int[][] grille, int couleurJoueur)
{
  System.out.print("Joueur ");
  if (couleurJoueur == JAUNE) {
    System.out.print("X");
  } else {
    System.out.print("O");
  }
  System.out.println(" :  entrez un numéro de colonne");

  int colonne = clavier.nextInt();
  // les indices des tableaux commencent par 0 en Java:
  --colonne;

  boolean valide = joue(grille, colonne, couleurJoueur);
  if (!valide) {
    System.out.println(" > Ce coup n'est pas valide");
  }
}
```

Profitons également qu'on est donc dans cette méthode demandeEtJoue pour améliorer l'aspect algorithmique cette fois-ci. Que se passe-t-il si le joueur a entré un coup qui n'est pas valide ? Dans la version précédente, on passait simplement à l'autre joueur, ce qui n'est certainement pas correct. Ici, on va devoir répéter tant que le coup n'est pas valide. Donc répéter, c'est-à-dire une boucle. De nouveau, on se pose la question: est-ce une boucle while ? Ou est-ce une boucle do while ? Donc est-ce qu'on commence par poser la question, puis vérifier la condition ? Ou est-ce qu'on commence par vérifier la condition, puis ensuite on exécute ? Donc ici, évidemment, on commence par exécuter, on commence par demander le coup avant de vérifier si c'était un coup valide.

notes

résumé

10m 23s

Donc c'est une boucle do while, et donc on va l'écrire. Ici, do, tout le bloc de notre méthode ici, et puis while. While quoi ? Je vous rappelle que ce qu'on veut faire c'est que si le coup n'est pas valide, alors on affiche le message, et on veut répéter l'entièreté de l'interaction ici tant que le coup n'est pas valide. Donc cela va s'écrire, while not valide. Voilà, on pense avoir terminé, on sauvegarde, on compile, et qu'est-ce qu'il se passe ? et qu'est-ce qu'il se passe ?

notes

résumé

11m 4s