Support de cours

Cours:

Initiation à la programmation (en Java)

Vidéo:

Init-JAVA-06-1-Intro-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Aide de vos propres fonctions. Nom de la variable scorejoueur. Premier joueur. Code suivant. Score du premier joueur. Déroulement de nos fonctions. Nombre de points. Type double. Variable scorejoueur. Partie des aspects de traitements. Nom de ma future fonction. Objectif de cette nouvelle leçon. Corps de ma future fonction. Score du deuxième joueur. Réutilisation de portions de programmes.

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

vers la vidéo

EPFL

Fonctions : introduction
(Partie 1)

Initiation à la programmation (Java)

Jamila Sam, Vincent Lepetit et Jean-Cédric Chappelier

...

notes

résumé

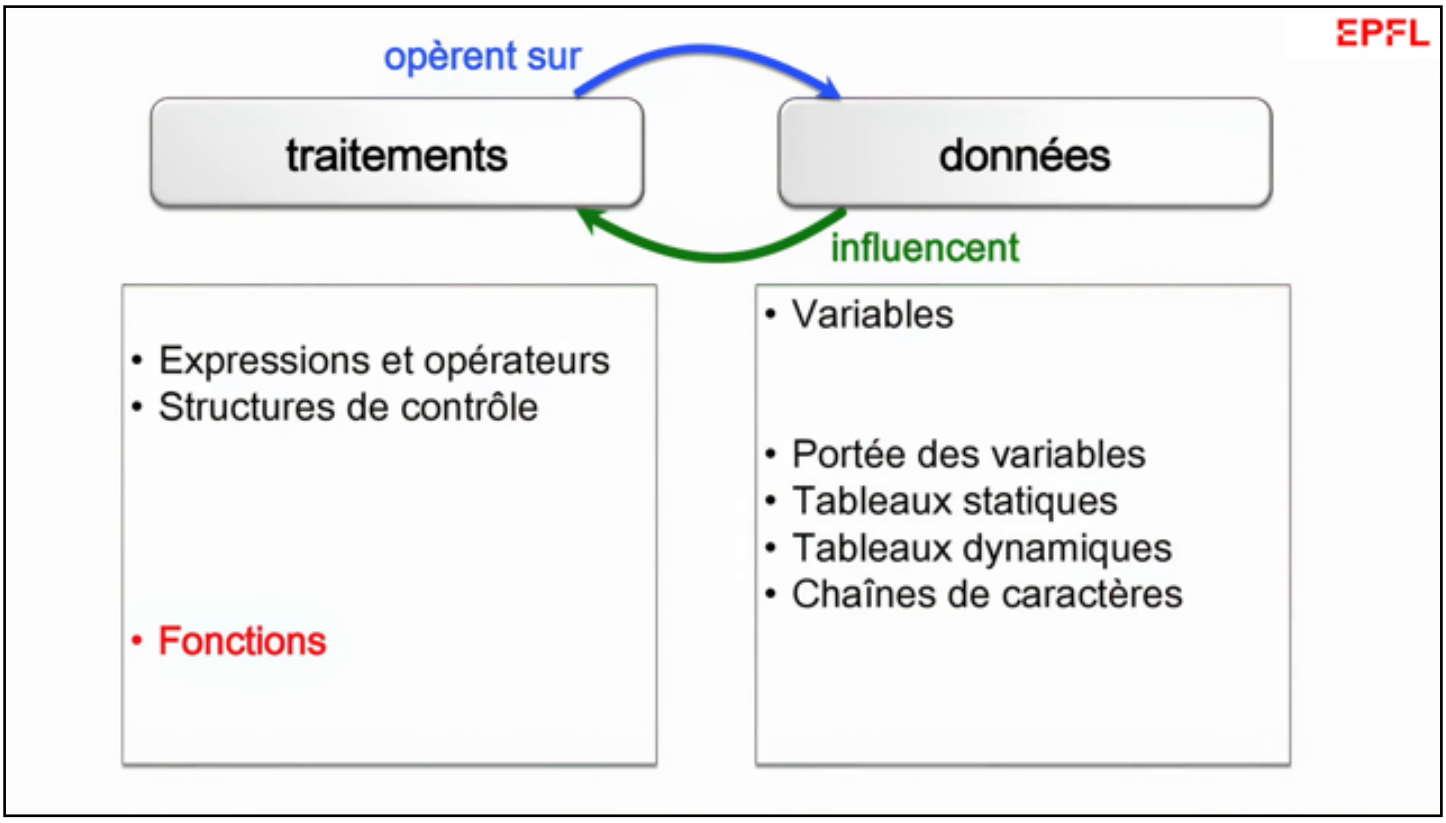
0m 0s

Bonjour, bienvenue à cette nouvelle leçon

notes

résumé

0m 1s



de notre cours d'introduction à la programmation Java. L'objectif de cette nouvelle leçon est de vous apprendre à organiser votre code en le modularisant à l'aide de vos propres fonctions.

notes

résumé

0m 6s

Notion de réutilisabilité

Si une tâche, par exemple :

```
do {
    System.out.println("Entrez un nombre entre 1 et 100 : ");
    i = clavier.nextInt();
} while ((i < 1) or (i > 100));
```

doit être exécutée à *plusieurs* endroits dans un plus gros programme

Les fonctions font partie des aspects de traitements, au même titre que les expressions et opérateurs, et les structures de contrôle que nous avons vues précédemment. Les fonctions opèrent donc, en règle générale, sur des données, et en retour les données vont influencer le déroulement de nos fonctions. Jusqu'à maintenant, les programmes que vous avez écrits étaient constitués d'une séquence linéaire d'instructions, sans organisation globale, et sans partage des tâches répétées. Par exemple, si la tâche consistant à demander un nombre à l'utilisateur, comme le fait le code suivant, devait être exécutée plusieurs fois dans notre programme, par exemple, pour demander une fois un nombre de points,

notes

résumé

0m 17s

Notion de réutilisabilité (2)

Pourquoi ne jamais dupliquer du code (copier/coller) :

Cela rend le programme

- inutilement long

une fois un temps, une fois un âge, que feriez-vous ? Vous pourriez être tentés de recopier le code autant de fois que nécessaire aux endroits appropriés, mais évidemment, ceci est une très mauvaise solution. Il ne faut jamais dupliquer de code en programmant, ne soyez pas tentés par le copier-coller, ce que vous vous voudriez recopier doit être mis dans une fonction.

notes

résumé

1m 1s

```java
import java.util.Scanner;

class ExempleFlat
{
    private static Scanner clavier = new Scanner(System.in);

    public static void main(String[] args) {

        System.out.println("Calcul des scores d'une partie");

        System.out.println("  Saisie des donnees du premier joueur :");
        double nbPoints = 0.0;
        do {
            System.out.print("    Entrez le nombre de points (0-100) : ");
            nbPoints = clavier.nextDouble();
        } while ((nbPoints < 0.0) || (nbPoints > 100.0));

        double temps = 0.0;
        do {
            System.out.print("    Entrez le temps de jeu (1-60) : ");
            temps = clavier.nextDouble();
        } while ((temps < 1.0) || (temps > 60.0));

        double scoreJoueur1 = 0.0;
        if (temps != 0.0) {
            scoreJoueur1 = (int) (1000 * nbPoints / temps);
        }

        System.out.println("  Saisie des donnees du second joueur :");

        nbPoints = 0.0;
        do {
```

Pourquoi ne faut-il jamais dupliquer du code ? Cela rend le programme inutilement long, difficile à comprendre, et surtout difficile à maintenir. Il faudrait reporter chaque modification dans chacune des copies. C'est pour ça que tout bon langage de programmation fournit des moyens pour permettre la réutilisation de portions de programmes. Et en Java, ça se fait en utilisant ce qu'on appelle des fonctions. Considérons par exemple le programme suivant. Ce programme commence par demander le nombre de points obtenus par un premier joueur, et vous pouvez constater qu'il utiliser une boucle do..while pour forcer l'utilisateur à entrer une valeur entre 0 et 100. Ensuite, le programme demande le temps utilisé par le premier joueur, qui sera cette fois-ci une valeur entre 1 et 60. Et à partir du nombre de points et du temps, le programme calcule le score du premier joueur. Ensuite, le programme répète ces opérations pour un deuxième joueur, il demande un nombre de points, un temps, et calcule le score du deuxième joueur. Finalement, le programme compare le score du premier joueur et le score du deuxième joueur, pour afficher si c'est le premier joueur ou le deuxième qui a gagné. Vous pouvez constater que dans ce programme, le code qui consiste à demander un nombre de points, un temps, et à calculer un score à partir du nombre de points et du temps, est répété deux fois dans le programme, une fois pour chacun des deux joueurs. Dans un cas comme celui-là, il faut utiliser une fonction pour éviter la duplication du code.

notes

résumé

1m 22s

```
        System.out.print("--> ");
        if (scoreJoueur1 < scoreJoueur2) {
            System.out.print( "Joueur 2 gagne");
        } else if (scoreJoueur1 != scoreJoueur2) {
            System.out.print("Joueur 1 gagne");
        } else {
            System.out.print("egalite");
        }
        System.out.println(" !");
    }

    static double saisieEtCalcul()
    {
        double nbPoints = 0.0;
        do {
            System.out.print("    Entrez le nombre de points (0-100) : ");
            nbPoints = clavier.nextDouble();
        } while ((nbPoints < 0.0) || (nbPoints > 100.0));

        double temps = 0.0;
        do {
            System.out.print("    Entrez le temps de jeu (1-60) : ");
            temps = clavier.nextDouble();
        } while ((temps < 1.0) || (temps > 60.0));

        double scoreJoueur = 0.0;
        if (temps != 0.0) {
            scoreJoueur = (int) (1000 * nbPoints / temps);
        }

        return scoreJoueur;
```

Pour ceci, je vais tout d'abord mettre de côté le code qui m'intéresse, c'est-à-dire le code qui demande un nombre de points, un temps, et qui calcule un score. Je vais mettre ce code à la fin du programme, mais avant la dernière accolade fermante. Je vais mettre ce code entre une accolade ouvrante et une accolade fermante. Ce code va constituer ce qu'on appelle le corps de ma future fonction. Ce corps, il va falloir le nommer. Je vais l'appeler ici, saisieEtCalcul, ce qui permettra d'y faire référence dans le reste du programme, et ça deviendra le nom de ma future fonction. Ma fonction doit fournir le score qu'elle aura calculé au reste du programme. Pour cela, je vais tout d'abord enlever les références au premier joueur, puisque ma fonction va être valable pour n'importe quel joueur. Je vais rajouter à la fin de ma fonction, le mot clé return, suivi de nom de la variable scoreJoueur, pour indiquer que ma fonction doit renvoyer la valeur contenue dans la variable scoreJoueur. Ne vous inquiétez pas, nous verrons tout cela en détails dans la suite du cours. La variable scoreJoueur a été définie de type double, la fonction va donc fournir une valeur de type double, ce qu'il faut indiquer au reste du programme, et ça se fait en ajoutant double avant le nom de la fonction. Un autre aspect des fonctions est qu'elles peuvent utiliser des valeurs fournies par le reste du programme pour pouvoir fonctionner. Notre fonction saisieEtCalcul est relativement simple, elle n'a pas besoin de telles valeurs. Pour indiquer cela, il faut ajouter après le nom de la fonction, une parenthèse ouvrante et une parenthèse fermante. Il me reste à rajouter le mot clé static au début de cette ligne,

notes

résumé

3m 1s

```java
import java.util.Scanner;

class ExempleFlat
{
    private static Scanner clavier = new Scanner(System.in);

    public static void main(String[] args) {

        System.out.println("Calcul des scores d'une partie");

        System.out.println("  Saisie des donnees du premier joueur :");
        double scoreJoueur1 = saisieEtCalcul();

        System.out.println("  Saisie des donnees du second joueur :");

        nbPoints = 0.0;
        do {
            System.out.print("    Entrez le nombre de points (0-100) : ");
            nbPoints = clavier.nextDouble();
        } while ((nbPoints < 0.0) || (nbPoints > 100.0));

        temps = 0.0;
        do {
            System.out.print("    Entrez le temps de jeu (1-60) : ");
            temps = clavier.nextDouble();
        } while ((temps < 1.0) || (temps > 60.0));

        double scoreJoueur2 = 0.0;
        if (temps != 0.0) {
            scoreJoueur2 = (int) (1000 * nbPoints / temps);
        }
```

pour une raison que nous verrons dans la suite du cours. Et voilà, nous avons crée notre première fonction. Je vais maintenant utiliser ma fonction saisieEtCalcul pour demander un nombre de points, un temps, et calculer le score du premier joueur. Pour ceci, je vais retourner à l'endroit où j'avais extrait le code que j'ai utilisé pour créer ma fonction, c'est-à-dire, ici. Je vais déclarer une variable scoreJoueur1, que je vais initialiser en utilisant la valeur fournie par la fonction saisieEtCalcul. Pour cela, je vais indiquer le nom de la fonction, suivi d'une parenthèse ouvrante, et d'une parenthèse fermante. Cette instruction initialise donc la variable scoreJoueur1 à la valeur fournie par la fonction saisieEtCalcul, et cette utilisation de la fonction saisieEtCalcul, s'appelle un appel à la fonction. Je peux faire exactement la même chose

notes

résumé

5m 1s

```java
import java.util.Scanner;

class ExempleFlat
{
    private static Scanner clavier = new Scanner(System.in);

    public static void main(String[] args) {

        System.out.println("Calcul des scores d'une partie");

        System.out.println("  Saisie des donnees du premier joueur :");
        double scoreJoueur1 = saisieEtCalcul();

        System.out.println("  Saisie des donnees du second joueur :");

        System.out.println("  Resultats :");
        System.out.println("    Joueur 1 : " + scoreJoueur1);
        System.out.println("    Joueur 2 : " + scoreJoueur2);
        System.out.print("--> ");
        if (scoreJoueur1 < scoreJoueur2) {
            System.out.print( "Joueur 2 gagne");
        } else if (scoreJoueur1 != scoreJoueur2) {
            System.out.print("Joueur 1 gagne");
        } else {
            System.out.print("egalite");
        }
        System.out.println(" !");
    }

    static double saisieEtCalcul()
    {
        double nbPoints = 0.0;
```

pour initialiser le score du deuxième joueur. Je peux remplacer tout ce code,

notes

résumé

6m 1s

```java
public static void main(String[] args) {

    System.out.println("Calcul des scores d'une partie");

    System.out.println("  Saisie des donnees du premier joueur :");
    double scoreJoueur1 = saisieEtCalcul();

    System.out.println("  Saisie des donnees du second joueur :");
    double scoreJoueur2 = saisieEtCalcul();

    System.out.println("  Resultats :");
    System.out.println("    Joueur 1 : " + scoreJoueur1);
    System.out.println("    Joueur 2 : " + scoreJoueur2);
    System.out.print("--> ");
    if (scoreJoueur1 < scoreJoueur2) {
        System.out.print( "Joueur 2 gagne");
    } else if (scoreJoueur1 != scoreJoueur2) {
        System.out.print("Joueur 1 gagne");
    } else {
        System.out.print("egalite");
    }
    System.out.println(" !");
}

static double saisieEtCalcul()
{
    double nbPoints = 0.0;
    do {
        System.out.print("    Entrez le nombre de points (0-100) : ");
        nbPoints = clavier.nextDouble();
    } while ((nbPoints < 0.0) || (nbPoints > 100.0));
```

par un simple appel à ma fonction saisieEtCalcul. Et vous pouvez constater que le programme est maintenant beaucoup plus concis. Créer la fonction saisieEtCalcul m'avait permis d'éviter de dupliquer du code, mais c'est aussi intéressant de créer une fonction, pour pouvoir se concentrer sur un aspect un peu difficile du programme,

notes

résumé

6m 7s

```
        } else if (scoreJoueur1 != scoreJoueur2) {
            System.out.print("Joueur 1 gagne");
        } else {
            System.out.print("egalite");
        }
        System.out.println(" !");
    }

    static double saisieEtCalcul()
    {
        double nbPoints = 0.0;
        do {
            System.out.print("    Entrez le nombre de points (0-100) : ");
            nbPoints = clavier.nextDouble();
        } while ((nbPoints < 0.0) || (nbPoints > 100.0));

        double temps = 0.0;
        do {
            System.out.print("    Entrez le temps de jeu (1-60) : ");
            temps = clavier.nextDouble();
        } while ((temps < 1.0) || (temps > 60.0));


        return scoreJoueur;
    }

    double score
    {
        double scoreJoueur = 0.0;
        if (temps != 0.0) {
            scoreJoueur = (int) (1000 * nbPoints / temps);
        }
```

comme par exemple ici, le calcul du score. Comme avant, je vais extraire le code qui m'intéresse, le mettre entre accolades, ma nouvelle fonction va renvoyer le score du joueur, donc une valeur de type double. Je vais appeler ma nouvelle fonction tout simplement score, pour l'instant je vais faire suivre le nom de ma fonction

notes

résumé

6m 35s

par une parenthèse ouvrante et une parenthèse fermante. Comme avant, ma fonction va renvoyer le score du joueur, donc pour cela je vais utiliser le mot clé return. Mais la différence, par rapport à avant, est que cette fonction a besoin maintenant de la variable nbPoints et de la variable temps. Donc dans ce cas, je vais rajouter ces variables entre les parenthèses ici, ce qui va constituer ce qu'on appelle les paramètres de la fonction. Comme avant, je vais retourner là où j'ai extrait le code, pour créer ma fonction, et appeler ma fonction à la place. Ça peut se faire par exemple ici, je vais appeler ma fonction en lui fournissant les valeurs de la variable nbPoints, et de la variable temps, et ça se fait de cette façon là. et ça se fait de cette façon là.

notes

résumé

7m 1s