Le mer. 22 avr. 2020 à 21:32, [redacted] a écrit :

Merci , monsieur cest quoi la difference entre
Chaine=new String ();
Dans le monde Java cette écriture est fausse, car le terme **C**haine (avec C majuscule)
désigne le nom d'une classe et non pas le nom d'une référence (sorte de pointeur en C) . Donc l'écriture est fausse si l'on s'en tient aux règles de programmation Java. Cependant si cette écriture a été faite par un débutant, il faudrait que je dispose d'un code plus complet.
Peut etre que Chaine aurait éyé déclaré de type String comme ceci est le cas pour la 2ème instruction
Et
String Chaine=new.nextLine();
Ici il y a violation des règles de programmation java: Il faut plutôt écrire String **c**haine)=...
maintenant l'écriture new.nextLine() peut etre vu comme la demande de lecture d'une chaine de caractère. Dans ce cas **new** serait la source de cette lecture (l'objet qui fait la lecture.
Cependant **new** est un mot clé de java et ne peut pas être utilisé de la sorte. Le **new** est utilisé pour créer des objets

Je sais que la 2eme est pour lecture cest tout

Un exemple de code plus correct:
Scanner sc =new Scanner(System.in);
String chaine = sc.nextLine()

[redacted]

22 avr. 2020
01:18

Salut monsieur, jai qlq questions a vous poser si vous avez le temps bien sur
 vous avez sitez dans votre pdf que quon declare une methode faut utiliser public static mais Jen trouve trop dexemple qui utilise que le public cest quoi la difference ?
Qs2: class constructeurs a quoi sert et esq son existence est nécessaire au point que le programme sans elle ne vas pas marcher?
Qst3 Le this cest quoi par exemple this .nom aquoi sert?
Scanner Sc= new Scanner (System.in) sert a quoi est esq on lecrit que dans la classe main
Désolé pour le dérangement et bonne nuit !.

Salut monsieur, jai qlq questions a vous poser si vous avez le temps bien sur
Question 1 : vous avez sitez dans votre pdf que quon declare une methode faut utiliser public static mais Jen trouve trop dexemple qui utilise que le public cest quoi la difference ?
En java tout se trouve défini à l'intérieur de classe.
Une classe peut servir à réaliser les fonctions à la C (comme celle du C) et permet aussi de déclarer des types de données structurée à la C.

Les fonction reprises du C en Java sont appellées des fonctions de classes. Elle doivent commencer par static. C'est aussi le cas de variable globales. Elle doivent etre déclarées comme static dans une classe.
Pour les classes qui représentent les types de données C, les champs ne sont pas précédées par static. Ce sont des champs de variable structurées. Une variable structurées est appelées objet en Java et en POO
Nous allons voir par la suite (prochaine question du forum) que les structures de données en Java peuvent aussi contenir des fonctions . Une variable structurée en Java est une sorte de variable structurée intelligente car elle peut réaliser des fonctions. Les fonctions sont appelées méthodes. Ces fonctions et ces champs sont appelées des champs ou fonction d'objet et non pas de classe. Ces fonctions ne sont pas précédé de static. Ils ne peuvent etre exécuté que dans le contexte d'un objet. C'est à dire qu'il faut créé un objet puis demander à cet objet d'exécuter une des ses fonction(on utilise plutôt le terme méthode au lieu de fonction).

Question 2: class constructeurs a quoi sert et esq son existence est nécessaire au point que le programme sans elle ne vas pas marcher?
Pour class tu viens d'avoir une idée avec la réponse précédente. Pour constructeur, c'est une fonction spéciale d'objet. Son rôle est l'initialisation de l'objet: la Création de l'objet est réalisée par l'opérateur new. L'initialisation des divers attributs (champs) d'un objet se fait en général avec des valeurs qui peuvent etre spécifiée comme paramètres au constructeur. On utilise les constructeurs de manière intense lorsque on fait de la POO.

Qst3 Le **this** cest quoi par exemple this .nom aquoi sert?

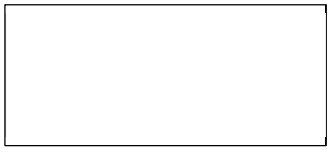
```
//Lors de l'exécution  d'une méthode sur un objet, celle-ci aurait été
//invoquée (appelée) dans le contexe de cet objet. Par exemple si on suppose
//la classe Complex suivante

public class Complex {
     private double r, i ;
     public Complex (double r, double i){this.r = r ; this.i = i;}
     public void setR(double r) {this.r = r ;}
     public void setI(double i) {this.i = i ;}
}
// Et si l'on suppose la classe Main suivante.
public class Main{
      public static void main(String[] args) {
            Complex c1 = new Complex (2, 3);
            Complex c2 = new Complex (2, 3);
            c1.setI(5);
            c2.setR(12);
      }
}
//AU niveau de la méthode de clase main  il y a dans un premier temps la
// création  de l'objet c1. Durant cette période de création
// l'objet en cours de traitemen est c1. La méthode qui s'exécute dans le
```

```
//contexte de cet objet localisé par c1_ c'est le constructeur  (Nous
//rappellons qu'un constructeur  est aussi une méthode mais particulière
//sa particularité c'est son objectif qui est l'initialisation de l'objet
//lors de sa création). Dans le contexte de java l'objet en cours localisé
// par c1 possède 2 référence: la référence c1  définie par le programmeur et
// la référence apellée this qui est défini par java et maintenue par java.
//Lorsque la construction de l'objet c1 est terminée, nous remarquons qu'il y
// a la création de l'objet c2. l'objet c2 devient alors l'objet courant. Il
// possède alors 2 référence: c2 et this. La référence this est mise à jour
// automatiquement par Java pour qu'elle puise localiser l'objet courant.
//lors de l'exécution de la méthode setI sur l'objet c1   (c1.setI(5);) c1
// redevient l'objet en cours et cette fois ci la référence this est mise à
// jour par Java pour localiser avec c1 le même objet. lors de l'exécution de
// la méthode setR sur l'objet c2   (c2.setR(12);) c2 redevient l'objet en
// cours et cette fois ci la référence this localise avec c2 le même objet.

//Pourquoi utiliser this: lors de l'écriture d'une classe, le programmeur n'a
// aucune idée des références qui seront utilisées pour instancier la classe
// (par exemple c1 et c2). Donc au lieu d'utiliser c1 ou c2 qui sont
// inconnues le programmeur utilisera la référence this
//Le this est utilisée pour faire une difference très claire entre atributs
// d'objets et variables locales (notament paramètre d'une méthode) possèdant
// les mêmes identifiants. Souvent les programmeurs veulent utiliser au
// niveau des paramètres les mêmes noms que que les attributs. Gràce à la
// référence this cette manière d'écrire les paramètre devient possible
```

Question : Scanner Sc= new Scanner (System.in) sert a quoi est esq on lecrit que dans la classe main

On l'écrit ou on veut: Si tu l'écrit à l'extérieur de main ou de toute autre fonction, précédé de static, ça devient un objet global (variable globale). Si tu l'écrit à l'intérieur d'une fonction (méthode) la référence sc devient locale, accessible seulement par la fonction et est détruite à la fin de l'exécution de la fonction

---



jeu. 4 juin 21:18 (il y a 5 jours)

Salut monsieur ,
pendant lapplication des tp jai remarqué que sur mes programme le System.out.println(); ne marche que a l'intérieur de methode aussi a l'intérieur de la

classe principale les autre classe ça ne marche pas apart si cetais a l'intérieur dune methode de classe esq cest normal ??

Qst2 dans le cours num 8 esq une relation faible on utilise les Arraylist et dans une relation forte on utilise tableau ou bien cest kifkif ???

System.out.println est une instruction et une instruction doit se trouver dans une méthode:C"est une demande à l"objet out d'executer la methode println

Qst2 : Il n y a pas de relation faible.. il y a une relation de composition faible que nous appelons aussi une relation d'agrégation.
L'utilisation de ArrayList ou de tableau est liée à la cardinalité de la relation. Si la limite supérieure de la cardinalité est plus grande que 1 et est connue (exemple : 5, 0..4, 2..9) on utilise un tableau. Si la limite supérieure est inconnue on utilise une liste (exemple : * , 0..*, 5..*, 10..*)

---

lun. 8 juin 17:22 (il y a 21 heures)

بالنسبة لـ CONCTRUCTEUR لازم كل Objet يكون عندو CONCTRUCTEUR خاص به
او يقدر CONCTRUCTEUR واحد يستعمل في plusieurs Objetes ?

اولا عليك ا ن تضعي في دماغة ان المشيد (constructeur) هدفه الوحيد هوو انشاء (او صناعة او تشييد) الأشياء (objets)، اي ا نه يستعمل فقط للتشييد، والمشيد يُعرف في ا طار تعريف الصنف (classe)، فهو وظيفة (fonctions, méthode) خاصة من الوظائف التي تُعرف في الصنف، وطبيعي جدا ان يُنشأ شيئا ما بمختلف الطرق، فيمكن لشئي ما من صنف ما ان تكون له طريقة واحدة في الإنشاء، ويمكن لشئي ما من صنف ما ان ينشأ بطريقتين، ويمكن لشئي ما من صنف ما ان ينشأ بأكثر من طريقتين، ولذا يمكن في اطار صنف ما ان نعرف بمشيد وا حد او اكثر، و كل مشيد يتبع طريقة ما في التشييد، والهدف هو انشاء شئ، اما القول ان "لكل شئ مشيد" ففيه بعض من اللبس، فالشيئ يصنع في اطار صنف باستعمال مشيد عرف في الصنف، فالصنف هو الذي يحتمل تعريف مشيد او اكثر و هذا حسب ما يراه مصمم وكاتب الصنف، ما قولك الآن هل فهمت نوعا ما

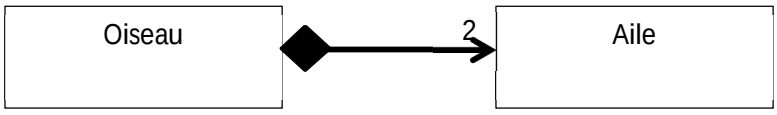
[redacted] 12:45 (il y a 1 heure)

À moi

bonjour..
monsieur si on a créé une classe et Nous n'avons pas été créés un constructeur pour cette class donc si nous créons un objet de cette class ,sera t'il crée de aniére normale ? et le constructeur par default inisialise tous les attributs a null ? et ici j'ai pas compri exactement dans cet constructeur de la class oiseaux le role des deux instructions
et merci d'avance

```
public class Oiseau {
   protected Aile[]  tabAile = new Aile[2];
   // Il faut lier l'objet Oiseau à 2 objets Aile
   // La liaison dans cet exemple est faite au niveau du constructeur par défaut
   public Oiseau(){
      tabAile[0] = new Aile();
      tabAile[1] = new Aile();
   }
}
```

En java, toute classe définie est associée automatiquement avec un constructeur qui ne possède aucun paramètre. C'est ce qu'on appelle le constructeur par défaut. Par exemple, la classe Aile possède le constructeur Aile() et la classe Oiseau possède le constructeur Oiseau().
Le constructeur par défaut **que java associe à une classe** ne fait rien. Il ne comporte aucune instruction. Le constructeur par défaut peut être redéfini (ré-écrit) dans une classe. Cette fois ci il pourrait être vide ou il pourrait comporter des instructions comme ceci est le cas dans la classe Oiseau.
Un constructeur, qu'il soit par défaut ou non (comporte au moins un parametre), est mis en œuvre par l'opérateur new. Avant l'exécution effective du constructeur, l'opérateur new reserve la zone mémoire nécessaire à l'objet et met à 0 tous les bits de la zone réservée l'objet. Ainsi, si l'objet dispose d'un attribut entier, sa valeur sera 0. Un attribut boolean aura la valeur false. Un attribut double aura la valeur 0.0D (ou 0.0). Un attribut float aura la valeur 0.0F (ou 0.0). Un attribut reference aura la valeur **null**.
Pour l'exemple, il apparait clair qu'un objet oiseau ne peut exister sans les 2 ailes. Donc llorsqueon crée un oiseauavec le constructeur par défaut, ce dernier cree automatiquement les 2 Ailes. Si je suppose que pour la classe Aile rien n'a été défini, le code corespondra au diagramme suivant :



12/06/2020 11:49 (il y a 1 heure)

À moi

bonjour monsieur

les objets sont stockés dans **la pile** d'éxécusion ou bien dans le **tas**?ou bien seulement les attributs des objets sont stockés dans l**e tas**

et que segnifie **class area**!!

question n02:

le variable statique count chaque objet peut modifier la valeur de cette variable ! donc les objet peut modifier les variable statique de la classe ??

```
//Java Program to illustrate the use of static variable which
//is shared with all objects.
class Counter2{
static int count=0;//will get memory only once and retain its value

Counter2(){
count++;//incrementing the value of static variable
System.out.println(count);
}

public static void main(String args[]){
//creating objects
Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();
}
}
```

Réponse Question 01 : Dans Java, tous les objets sans exception sont stocké dans le tas (souvent j'utilise les termes « mémoire dynamique » pour désigner le tas). Donc tous les attributs qui définissent un objet sont dans le tas.

Concernant la référence de l'objet :

- Si cette référence est un attribut d'un objet, elle est alors avec l'objet dans le tas
- Si la référence est un paramètre de méthode ou une variable locale de méthode elle se trouve dans la pile d'exécution
- Si la référence a été déclaré comme attribut statique d'une classe (c'est un attribut de classe) alors elle se trouve dans la mémoire dite statique

Réponse Question 02 : (Une question similaire a été posé dans le forum de révision)

Une variable statique (ou variable de classe ou attribut de classe) publique est accessible de n'importe quel méthode de n'importe quelle classe. Rappelons qu'un constructeur est une méthode aussi. Il suffit de mettre le nom de classe suivi du nom de l'attribut dans la méthode. Si la méthode qui référencie un attribut de classe se trouve dans la même classe on peut enlever le nom de la classe. Une variable statique publique est une sorte de variable globale.

Pour être plus précis,Il vaut mieux dire « Méthode qui change la variable de classe » et non pas « objet qui change la variable de classe «

Il faut aussi voir le document « IMPORTANT » qui parle de l'utilisation des attributs et méthodes statique. **A titre d'exemple** si dans notre conception de diagramme de classe on remarque qu'une classe **A** est liée à tous les objets de la façons suivante : la flèche de la relation vient des autres classe vers la classe A et la cardinalité est de 1 : Dans ce cas on a un objet unique (appelé souvent singleton) connus par tous les autres objet. Dans ce cas on peut par exemple

- Interdire la création d'objet de la classe **A** en mettant à private le constructeur par défaut
- Ne définir aucun autre constructeur pour **A**
- Rendre toutes les méthodes et attributs comme static

De cette manière au niveau des autres classes, il n y aura pas de référence sur un objet de la classe **A** car les objets de la classe **A** ne vont pas exister. Les autres objets des autres classes vont accéder à la classe **A** (qui représente le singleton) en utilisant le nom de la classe suivi de la méthode ou de l'attribut si celui ci est accessible (public)

12/06/2020 14:48 (il y a 11 minutes)

À moi

monsieur si on a utilisé public et static les deux pour définir une methode comme la methode main donc cette methode peuvent etre accédé par l'objet et la classe les deux ?

Un attribut ou méthode "public static" est accessible par n'importe quelle méthode de n'importe quelle classe ou objet.
Pour y accèder il faut faire précéder le nom de la méthode ou de l'attribut par le nom de la classe.

Si celui qui accède (méthode de classe ou méthode d'objet) à un attribut ou méthode "public static" se trouve dans la même classe il n'est pas nécessaire de faire précéder le nom de l'attribut ou méthode par le nom de classe.. Pour ce dernier cas, il est plus sûr et préférable d'utiliser le nom de la classe pour accéder à l'attribut ou méthode

12/06/2020 21:53 (il y a 46 minutes)

À moi

استاذ methode get et set نستعملهم كيكون عندما Objet فيه constrecture فارغ بدون paramètre ?

لا توجد اي علاقة للمشيدات (constructeur) بالمُوصلات للسمات من امثال الموصل "خذ" او "تحصل على" (get) او الموصل "ثبت" او "ضع" (set)، فكل من المشيد والموصلات من الوظائف التي تعرف في الصنف و تستهدف الأشياء المصنعة على شكل الصنف (الأشياء التابعة للصنف)، فهدف المشيد هو انشاء الشيئ طبقا لصنفه، ولا يمكن استعمال المشيد الا في مرحلة انشاء الشيئ، و من بين المشيدات المشيد الإفتراضي (constructeur par defaut) الذي يعرف بدون اي معامل، اما الموصلات فهدفها الوصول الى سمات (attribut) الشيئ اما لـ"لإطلاع على قيمة السمة (الموصل حصل او "تحصل على" اي get) او لتغيير قيمة السمة (الموصل ثبت " اي set)، و في اداب كتابة البرا مج بلغة "جافا"، يكون اسم الموصل مكون من set او get متبوع مباشرة باسم السمة، مع مراعاة القاعدة ان او حرف يجب ان يكون من الأحرف اللاتينية الكبيرة فمثلا اذا كان اسم السمة هو nom يكون الموصل المغير للسمة setNom واسم الموصل المطلع على قيمة السمة هوgetNom

14 juin 2020 20:31 (il y a 18 heures)

À moi

سلام استاذ بالنسبة لـ Fonction في return متي نكتب this و متي لا نكتبها

الكلمة this تستعمل لإبراز التعامل مع سمات الشيء، ففي إطار كتابة نص لصنف ما، و في إطار كتابة نص وظيفة منتسبة للصنف (method) من الجيد ومن المستحب الإشارة إلى السمات باستعمال الكلمة this، ولا نستعمل البتة كلمة this مع سمات الصنف(attribut de classe) ، فللوصول إلى سمات الصنف نستعمل اسم الصنف

akila akila

16/06/2020 12:30 (il y a 36 minutes)

À moi

صبح الخير استاذ , لم افهم ماذا تقصد بـ Fichier في جافا , Plusieurs classes dans un même fichier ؟

في الحالات الشائعة ، نضع في كل ملف **صنفا واحدا** ويكون الصنف **عام** متاح للجميع ويحمل الملف نفس اسم الصنف بزيادة الإضافة java.، مثلا الصنف العام Voiture يكون في الملف المسمى Voiture.java
وهكذا من خلال اسم الملف الذي يحترم قواعد كتابة اسماء الأصناف نستخلص اسم الصنف العام الموجود في الملف، فمثلا اذا كان اسم الملف هو Computer.java فان اسم الصنف الذي يحتويه هذا الملف هو public class **Computer** وقد عرف هذا الصنف انه صنف عام (public) متاح للإستعمال من قبل الأصناف الأخرى،
ومع هذا يمكننا ان نضع في ملف اكثر من صنف، وفي هذه الحالة يجب ان نراعي القواعد التالية:
1 - لا يمكن ان يحتوي الملف على اكثر من صنف واحد عام، فاذا كان في الملف صنف عام; وحيد، وجب على الملف ان يحمل نفس اسم الصنف متبوعا بالإضافة java.
2- يمكن عدم وجود اي صنف عام في الملف، و في هذه الحالة، يجب ان يكون اسم الملف مختلفا تماما عن اسم اي صنف فيه، بل اكثر من ذلك، يجب على اسم الملف ان لا يكون على شاكلة اسماء الأصناف التي يكون اول احرفها من الحروف الإنكليزية الكبيرة، فمثلا الملف lib.java والملف globals.java لايحتويان على اصناف عامة، اما الإسمين Global.java و Lib.java فهما غير مرغوب فيهما اذا لم يحتويا على صنف وحيد عام، ويوحيان ان الملفين يحتويان على الصنفين العامين Lib و Globals.

24 06 2020 ---10:34 (il y a 3 heures)

À moi

صبح الخير استاذ , لم افهم الفرق بين composition et d’agrégation ؟

Composition معناها "التكوين"، اي ان شيء مكون من اشياء أخرى، وهذه الأشياء الأخرى التي مكون منها الشيء داخلية له، ولا يعرفها الا هو، فهو من انشأها، وهناك قاعدة هامة اخرى: اي ال الشئ المكون لو اتلف لأتلفت معه كل مكوناته، ومن بين ما سبث يمكننا القول ان الأجزاء المكونة للشيئ لا يمكن مشاركتها، اذ لا يعلمها الا الشيئ المكون منها، مثال على هذا؟ بيت مكون من غرف، عمارة مكونة من منازل،

Agrégation معناها التركيب، و يمكن ايضا وصفها بالتكوين، لكن هذا التكوين ليس بقوة التكوين السابق، فهو تكوين مرن، ضعيف، فالأجزاء يمكن ان توجد دون وجود التركيب، فالأجزاء يمكن ان يعرفها غير الشيئ المركب بها، والقاعدة الأساسية التي تفرق التركيب بالتكوين القوي : اذا اتلف المركب لا تتلف مكوناته

متى نستعمل هذا او ذاك، فهذا راجع الى مصمم البر نا مج و كيف يرى الحل

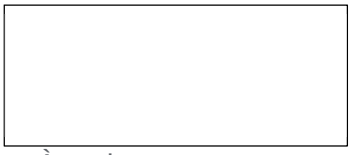
mer. 24 juin 20:54 (il y a 22 heures)

À moi

استاذ , objet متي يكتب بهذا الشكل ;()Stock stock=new Stock ومتي يكتب بهذا الشكل ;Stock stock عندما قرأة diagramme de class en java ?

في ال برمجة با لأشياء على مستوى لغة جافا يجب علينا ان نفرق بين: الصلات (reference) و بين الشيئ (object)، فالأشياء عبارة عن متغيرات طارئة (dynamic variable) التي تنشأ وقت تنفيذ البرنامج، والمتغيرات الطارئة، اي الأشياء في لغة جافا، لا يمكن الوصول اليها الا عبر الصلات، تنشأ الأشياء باستعمال العملية new

فالكتابة : ;Stock stock معناها انشا ء صلة يمكن ربطها بأشياء من الصنف Stock



26 06 2020 -- 17:39 (il y a 1 heure)

À moi

مساء الخير أستاذ , السوال 1 Constructeure de supperclass dans héritage بما انه لا يمكن ان يكون Public هل يكون Private او Protected ؟

اما السوال 2 / عندما يعطي لنا في diagramme de class الصنف الأم SupperClass فيها متغيرات علي شكل public عندما نحولها الي java هل يجب ان نكتبها Protected أم نتركها Public و أيضا نفس السوال عندما يكون عندنا supperinterface . و شكرا .

الجواب عل ى السؤال الأول: لا ادري حقيقة من اين اتيت بالفكرة "1 Constructeure de supperclass dans héritage بما انه لا يمكن ان يكون Public"، فالمشيدات (Costructors) يمكن ان تكون عامة اوخاصة او محمية و هذا حسب ما يريده مصمم الصنف، في العموم تكون المشيدات عامة، و في بعض الحالات التي لا نريد صناعة اشياء لا معنى لها، يقوم مصمم الصنف من منع استعمال المبا شر لبعض المشيدات خاصة المشيد الإفتراضي، فمثلا لا يمكن ان نتصور شخص بدون اسم او لقب، فهنا يقوم مصمم الصنف "شخص" (Person) بمنع استعمال المشيد الإفتراضي كون هذا الأخير ينشأ شخصا بدون اسم ا و ل قب، و كل هذه الأمور ناتجة عن رأي المصمم و ما يريده من ال صنف الذي يصممه.

هذا الأمر ينطبق على كل المشيدات، اكانت هذه المشيدة في الصنف او في الأصنا ف التي يرثها

السؤال الثاني: عندما نريد في صنف ما ان نعاود استعمال ما في صنف آخر (اتغلال صنف اعلى بعلاقة الوراثة)، فاذا كانت سمات الصنف الأعلى قد جعلت عامة فتبق في الصنف الأسفل عامة، ولا وجود (حسب علمي ، فانا لا اعرف كل شئ) لواجهة اعى من واجهة اخرى، فكل ما في الواجهة عام

30 06 2020 -17:33 (il y a 5 heures)

À moi

bonjour monsieur .....j'ai pas compris le role de ces trois constructeurs ...est ce que sont les meme !! et est ce que on doit lutilisée au meme temps.

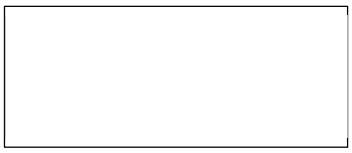
```
public class Ordinateur {
   ArrayList<Processeur> listProcesseur =
            new ArrayList<Processeur>();

// Méthodes ajoutées en plus du code  incomplet
//issu de la transformation UML→ Java
   Ordinateur(ArrayList<Processeur> listProcesseur){
     this.listProcesseur.addAll(listProcesseur);
   }
   Ordinateur(Processeur processeur){
     this.listProcesseur.add(processeur);
   }
   void addProcesseur(Processeur processeur){
     this.listProcesseur.add(processeur);
   }
```

Réponse : LOrsque on construit une maison on peut disposer de plusieurs plan de construction. Certains plan de constructions traitent de tous les aspects de la maison. Certains plan ne font qu'une construction partielle(La maison est construite mais non achevée.. elle nécessite par exemple la peinture, le carrelage etc...).Lorsque on veut réellement construire on la construit selon un des plans complet ou avec un plan partiel ou avec un plan complet qui est basé sur des plans partiels... L'essentiel:: Lorsque on fait le new on ne peut citer qu'un constructeur qu'il soit complet ou partiel ou réalisé à base de constructeurs partiel. Donc on a le choix.. Nous ne sommes pas obligé d'utiliser tous les constructeurs.

Maintenant concernant la classe Ordinateur de l'image : Cette classe offre la possibilité de créer des objet de la classe Ordinateur à l'aide d'un des 2 constructeurx cités. Le premier permet de créer un ordinateur pouvant avoir plusieurs processeur une fois construit (Ordinateur multiprocesseur). Je dis bien permet de créer un Multiprocesseur. Mais si la liste donnée contient un seul processeur, ce constructeur

devient équivalent avec le 2ème qui permet de créer un ordinateur avec un seul processeur.

En conclusion: si l'on désire créer un objet, on utilise un et un seul constructeur. Le constructeur peut se suffire à lui même pour construire l'objet comme il peut se faire aider par d'autres constructeurs et méthode dans la construction de l'objet. Mais pour le new il faut donner un seul constructeur



30 06 2020 18:47 (il y a 4 heures)

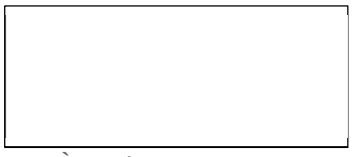
À moi

سلام عليكم ، استاذ هل interface تحتوي على ثوابتس فقط أو يمكن أن تحتوي علي متغيرات أيضا ؟

الواجهة (interface) لا تحتوي على متغيرات، تحتوي فقط على ثوابت عامة تابعة لها، اي للوصول الى ثابتة ما من الواجهة علينا كتابة اسم الواجهة ثم نقطة ثم اسم الثابتة، والثابتة تكتب اجباريا بالحروف الكبيرة،

اما الوظائف التي تظهر في الواجهة فهي فقط رؤوس لوظائف تعرف في الأصناف التي تنجز الواجهة، وهي وظائف عامة ايضا، ويمكن لكل صنف ان ينجز الوظيفة حسب اهدافه.

كل ما يكتب في الواجهة عام، يمكن لأي وظيفة من اي صنف الوصول اليها.



ven. 3 juil. 19:23 (il y a 15 heures)

À moi

مساء الخير استاذ عندما نحول من diagramme de class إلي JAVA كيكون عندي في نهاية السهم 1 متي اعتبره Objet ومتي أعتبره reference ? وشكرا

اولا لا يمكن الكلام عن شيئ بدون الصلة (reference)، فلا يمكن الوصول الى شيئ ما الا عبر صلة، والصلة تنشأ قبل الشيئ ثم تربط به حسب ما يريده كاتب البرنامج،

عندما يكون العدد المحاذي للسهم هو واحد، فمعنى هذا ان الشئ التابع للصنف (نسميه الشيئ الأول) الذي انطلق منه السهم يجب ان يكون مرتبط بشئ واحد (لا اقل و لا كثر) من الصنف المستهدف من السهم (نسميه الشيئ الثاني)، والسؤال هنا هو: متى يتم ربط الشيئ الأول بالثاني: يتم هذا عند الإنتهاء من صناعة الشيئ الأول،

12:40 (il y a 3 heures)

À moi

صبح الخير استاذ , ماهي قواعد كتابة Prototype ?

يتكون نموذج اي وظيفة من : 1- اسم الوظيفة، 2- متبوع بين قوسين بنمط كل معامل، ولا يدخل في تعريف النموذج نمط القيمة التي ترجعها الوظيفة، فمثلا اذا كان راس الوظيفة هو int max(int a, int b) فإن النموذج هو max(int a, int b) ،

او باختصار max(int, int) ، و هذا هو التعريف المطلق.

في بعض الأحيان، حسب تفاهم الأستاذ مع الطلبة، فانه مجازا نستعمل كلمة نموذج لنعني بها رأس الوظيفة كاملا، اي ان نمط القيمة التي ترجعها الوظيفة يجب ان يذكر، و هذا ما نراه عندما نصرح بالوظائف الإفتراضية في الأصناف الإفتراضية

(Abstract method in Abstract classes) او في الواجهات (interfaces) ففي هذه الحالة، و زيادة على رأس الوظيفة نذكر ايضا رؤية الوظيفة هل هي عامة، خاصة محمية الخ

05/07/2020 13:07 (il y a 19 minutes)

À moi

صباح الخير استاذ , أردت التأكد إذا كان في class Diagramme de هل كل attribut أو Méthode مسطر عليها تكون public static أم فقط main .?

وايضا بالنسبة لـinterface هل كل Méthode فيها تكون abstract ?

واخر شئ هل الثوابت تكتب دائما Majuscule أم فقط في interface ؟ وشكرا .

الجواب 1: نعم كل سمة او وظيفة تكتب مسطرة فهي سمة او وظيفة ساكنة (static)، اما رؤية السمة او الوظيفة فتكون حسب الرمز الذي يذكر في بداية السطر الذي فيه يشار الى السمة او الوظيفة: فالرمز + معناه عام (public) والرمز - معناه خاص الخ...

الجواب 2: نعم، على مستوى واجهة ما (interface) كل ما يصرح به من رؤوس لوظائف فهي افتراضية (abstract) وايضا عامة (دون اللجوء الى كتابة الكلمة public فهذه الكلمة لا تذكر في كتابة محتوى الواجهات من رؤوس وظائف او اسماء لقيم ثابتة).

الجواب 3: في اي موقع كان، صنف او واجهة، تكتب اسماء الثوابت دائما بالحروف اللاتينية (الإنجليزية بالتحديد) الكبيرة

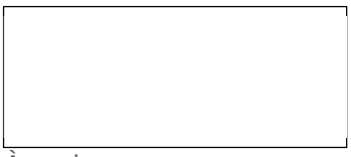
07/07/2020 16:07 (il y a 40 minutes)

À moi

السلام عليكم ، أستاذ بالنسبة لـ Méthode static هل يمكن استدعائها داخل Main بذكر اسمها فقط في عوض استدعائها بهذا الشكل NomDeClass.NonDeMéthode حدث لي اشكال لانه في lesTestes كتبت لنا داخل System.out.println مباشرتا إسم Méthode .

فيما يخص وضائف الصنف، اي تلك التي نصرح بها انها ساكنة (static)، فعند طلب تنفيذها نكتب اسم الصنف متبوعا باسم الوظيفة، و في حالة واحدة يمكن ان لا نذكر اسم الصنف، وهي الحالة التي يوجد النداء من قبل وظيفة صنف او وظيفة شيء كتبتا في نفس الصنف، كما نراه في المثالين التاليين:

```
public class Example {
     static   void cm1(){}   // A class Method
     void om1() {}     // this is an instance (or object) method

     static void cm2(){   // Another class Method
     // Calling the m1 class Method using class name. Not necessary  but recommended
        Example.cm1();
     }
     static void cm3(){   // A Third class Method
     // Calling the m2 class Method directly without specifying class name. correct
        Example.cm2();
     // Calling the m1 class Method directly without specifying class name. correct
        Example.cm1();
     }

     void om2(){
        //   Calling a class method without using class name: Correct
        cm1();
        //   Calling a class method  using class name: Correct and recommended
        Example.cm2();
     }
}

public class SecondExample{
      // to use  Example class Method  we must specify the class name
       void   anObjectMethod(){
             // Calling Example class Methods cm1 and cm3
             Example.cm1();
             Example.cm3();
       }

        static void aClassMethod(){
             // Calling Example class Methods cm2 and cm3
             Example.cm2();
             Example.cm3();
       }
}
```



06 09 2020 14:42 (il y a 4 heures)

À moi

Bonjour monsieur

Pour cette instruction en java par exemple

Complexe q= new complexe();

Donc c'est un référence d'une variable de type Complexe

La référence q est stocké dans le tas??

Et la variable de type complexe aussi stocké dans le tas?? Et merci .

L'écriture Complexe q= new complexe(); est une déclaration et non pas une instruction.

Cette déclaration crée effectivement un objet de type Complexe et q est la référence de cet objet. L'objet est automatiquement crée dans le tas (ce que j'appelle souvent mémoire dynamique).

Quand à la référence q sa création et position en mémoire dépend de la position de la déclaration Complexe q= new complexe();

Il y a 2 endroits où cette déclaration peut etre faite :

- Au niveau d'une classe : ici q est un attribut d'objet. La référence **q** est ainsi créée lorsque l'objet est crée vu que **q** est un attribut (ou champ) de l'objet.
- Au niveau d'une méthode : dans ce cas **q** est une **variable locale**. La référence **q** est crée dans **la pile d'exécution** et non pas dans le **tas**

07/08/2020 14:53 (il y a 45 minutes)

A moi

Bonjour monsieur

•On a dit que le contenu d'une référence java n'est pas un nombre entier (adresse mémoire) donc le contenu c'est quoi ??

• exemple :

Ona deux références p et q :

Complexe p= new complex();

Complexe q = new complexe ();

On a cette instruction :

P=q;

Je sais que après cette instruction p va pointé à q et q pointe à son objet ou bien p pointe sur l'objet de q directement ??

J'ai pas compris le sens réellement de cette instruction

Et merci .

Réponse:

En java on n'aime pas dire qu'une reference est un pointeur ou que le contenu d'une référence comporte une adresse mémoire (donc un entier), comme ceci est le cas en C et C++. Personnellement j'ignore exactement ce que contient comme information une référence, mais elle contient **au moins** l'adresse mémoire de l'objet et cette adresse est un entier. Une référence **pourrait** (je dis bien **pourrait**) comporter d'autres informations, notament le nom de la classe associée (c'est juste une idée).

Pour la programmation et la compréhension de ce qui se passe on peut voir la référence exactement comme un pointeur en respectant la grande difference entre pointeur et référence et qui réside dans ce qui suit :

- Avec un pointeur on peut réaliser toutes les opérations arithmétique et logique qu'on peut réaliser sur un entier (=, ==, >, <, +, -, /, *, % etc..).
- Avec une référence on ne peut utiliser que 2 opérateurs : l'Assignation (=) et la comparaison à égalité (==).

Après l'opération p=q, les 2 références p et q (ou pointeurs si tu désires) localisent le même objet : C'est l'objet qui a été crée par Complexe q = new complexe (); L'objet qui a été crée avec Complexe p = new complexe (); se trouve maintenant sans aucune référence. Cet objet sera détruit et sa place mémoire dans le tas libérée.