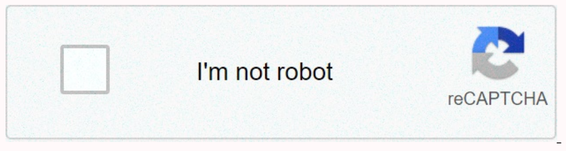


I'm not robot!

Exercice java heritage corrigé pdf

Cet exercice vous permettra de concevoir une hiérarchie de classes utilisant la notion d'interface. Il vous servira également de révision pour les notions d'héritage, de classes abstraites et de polymorphisme. Le directeur d'une entreprise de produits chimiques souhaite gérer les salaires et primes de ses employés au moyen d'un programme Java. Un employé est caractérisé par son nom, son prénom, son âge et sa date d'entrée en service dans l'entreprise. Dans un fichier Salaires.java, codez une classe abstraite Employe dotée des attributs nécessaires, d'une méthode abstraite calculerSalaire (ce calcul dépendra en effet du type de l'employé) et d'une méthode getNom retournant une chaine de caractère obtenue en concaténant la chaine de caractères « L'employé » avec le prénom et le nom. Dotez également votre classe d'un constructeur prenant en paramètre l'ensemble des attributs nécessaires. Calcul du salaire Le calcul du salaire mensuel dépend du type de l'employé. On distingue les types d'employés suivants : Ceux affectés à la Vente. Leur salaire mensuel est le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 400 Francs. Ceux affectés à la Représentation. Leur salaire mensuel est également le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 800 Francs. Ceux affectés à la Production.

Chapitre 3

L'héritage

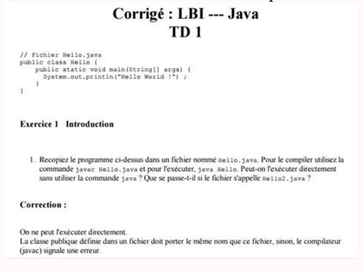
3.1 LE PRINCIPE DE L'HÉRITAGE

3.1.1 Les définitions

Le deuxième grand principe de la programmation objet après l'encapsulation (voir figure 2.1, page 11) est le concept d'héritage. Une classe B qui hérite d'une classe A hérite des attributs et des méthodes de la classe A sans avoir à les redéfinir. B est une **sous-classe** de A ; on trouve A est la **super-classe** de B ou la classe de base. On dit encore que B est une **classe dérivée** de la classe A, ou que la classe B étend la classe A. Une classe ne peut avoir qu'une seule super-classe ; il n'y a pas d'héritage multiple en Java. Par contre, elle peut avoir plusieurs sous-classes (voir figure 3.2).

Figure 3.1 — Le principe de l'héritage : la classe B hérite de la classe A.

Leur salaire vaut le nombre d'unités produites mensuellement multipliées par 5.



Corrigé : LBI --- Java
TD 1

```
// Fichier Hello.java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World !") ;
    }
}
```

Exercice 1 Introduction

1. Recopiez le programme ci-dessus dans un fichier nommé `Hello.java`. Pour le compiler utilisez la commande `javac Hello.java` et pour l'exécuter, `java Hello`. Peut-on l'exécuter directement sans utiliser la commande `java` ? Que se passe-t-il si le fichier s'appelle `Hello2.java` ?

Correction :

On ne peut l'exécuter directement.
La classe publique définie dans un fichier doit porter le même nom que ce fichier, sinon, le compilateur (javac) signale une erreur.

Ceux affectés à la Manutention. Leur salaire vaut leur nombre d'heures de travail mensuel multipliées par 65 francs. Codez dans votre fichier Salaires.java une hiérarchie de classes pour les employés en respectant les conditions suivantes : La super-classe de la hiérarchie doit être la classe Employe. Les nouvelles classes doivent contenir les attributs qui leur sont spécifiques ainsi que le codage approprié des méthodes calculerSalaire et getNom, en changeant le mot « employé » par la catégorie correspondante. Chaque sous classe est dotée de constructeur prenant en argument l'ensemble des attributs nécessaires.
N'hésitez pas à introduire des classes intermédiaires pour éviter au maximum les redondances d'attributs et de méthodes dans les sous-classes Employés à risques Certains employés des secteurs production et manutention sont appelés à fabriquer et manipuler des produits dangereux. Après plusieurs négociations syndicales, ces derniers parviennent à obtenir une prime de risque mensuelle. Complétez votre programme Salaires.java en introduisant deux nouvelles sous-classes d'employés. Ces sous-classes désigneront les employés des secteurs production et manutention travaillant avec des produits dangereux. Ajouter également à votre programme une interface pour les employés à risque permettant de leur associer une prime mensuelle fixe de 200.-. Collection d'employés Satisfait de la hiérarchie proposée, notre directeur souhaite maintenant l'exploiter pour afficher le salaire de tous ses employés ainsi que le salaire moyen. Ajoutez une classe Personnel contenant une « collection » d'employés. Il s'agira d'une collection polymorphique d'Employe – regardez le cours si vous ne voyez pas de quoi il s'agit. Définissez ensuite les méthodes suivantes à la classe Personnel : void ajouterEmploye(Employe)qui ajoute un employé à la collection. void calculerSalaires()qui affiche le salaire de chacun des employés de la collection. double salaireMoyen()qui affiche le salaire moyen des employés de la collection. Testez votre programme avec le main suivant : class Salaires { public static void main(String[] args) { Personnel p = new Personnel(); p.ajouterEmploye(new Vendeur("Pierre", "Business", 45, "1995", 30000)); p.ajouterEmploye(new Representant("Léon", "Vendtout", 25, "2001", 20000)); p.ajouterEmploye(new Technicien("Yves", "Bosseur", 28, "1998", 1000)); p.ajouterEmploye(new Manutentionnaire("Jeanne", "Stocketout", 32, "1998", 45)); p.ajouterEmploye(new TechnARisque("Jean", "Flippe", 28, "2000", 1000)); p.ajouterEmploye(new ManutARisque("Al", "Abordage", 30, "2001", 45)); p.afficherSalaires(); System.out.println("Le salaire moyen dans l'entreprise est de " + p.salaireMoyen() + " francs."); } } Vous devriez obtenir quelque chose comme : Le vendeur Pierre Business gagne 6400.0 francs. Le représentant Léon Vendtout gagne 4800.0 francs. Le technicien Yves Bosseur gagne 5000.0 francs. Le manut. Jeanne Stocketout gagne 2925.0 francs. Le technicien Jean Flippe gagne 5200.0 francs. Le manut. Al Abordage gagne 3125.0 francs. Le salaire moyen dans l'entreprise est de 4575.0 francs. La correction exercice Java (voir page 2 en bas) Avec des exercices corrigés en Java sur les classes et l'héritage, vous pratiquerez divers concepts du langage Java.
Vous commencerez par des exercices Java de base à des exercices plus avancés. La solution est fournie pour chaque exercice. Vous devez essayer de résoudre chaque problème par vous-même avant de vérifier la solution. Si vous avez des questions concernant chaque problème, nous vous encourageons à les poster sur notre forum. Vous pouvez utiliser l'éditeur Java suivant pour résoudre les exercices suivants: (Cliquez sur l'onglet input si vous souhaitez entrer des valeurs, cliquez sur Run pour exécuter votre programme, le résultat sera affichée sur l'onglet output). Exercice 1:Écrivez une classe « Rectangle » ayant deux variables « a » et « b » et une fonction membre « surface() » qui retournera la surface du rectangle.import java.util.*; class Rectangle { public int a,b; public int surface(){ return a*b; } } public class Test{ public static void main(String []args){ Rectangle rectangle = new Rectangle(); Scanner in = new Scanner(System.in); System.out.println("Entrez la largeur(a) du rectangle :"); rectangle.a = in.nextInt(); System.out.println("Entrez la longueur(b) du rectangle :"); rectangle.b = in.nextInt(); System.out.println("Surface : "+ rectangle.surface()); in.close(); } } Exercice 2:Écrivez une classe « Somme » ayant deux variables « n1 » et « n2 » et une fonction membre « sum() » qui calcule la somme. Dans la méthode principale main demandez à l'utilisateur d'entrez deux entiers et passez-les au constructeur par défaut de la classe « Somme » et afficher le résultat de l'addition des deux nombres.import java.util.*; class Somme { public int n1, n2; //constructeur par défaut Somme(int nbr1, int nbr2){ n1 = nbr1; n2 = nbr2; System.out.println("Les nombres sont initialisés."); } public int sum(){ return n1 + n2; } } public class Main { public static void main(String []args){ Scanner in = new Scanner(System.in); System.out.println("Entrez le premier nombre :"); int n1 = in.nextInt(); System.out.println("Entrez le deuxième nombre :"); int n2 = in.nextInt(); //appeler le constructeur par défaut. Somme s = new Somme(n1,n2); System.out.println("Le résultat de l'addition est : "+ s.sum()); in.close(); } } Exercice 3:Écrivez une classe Java appelée « Student » avec les membres suivant :nom (de type String),note1, note2 (de type int)Le programme demande à l'utilisateur d'entrer le nom et les notes. calc_moy() calcule la note moyenne et show() affiche le nom et la note moyenne.import java.util.*; class Student { public String nom; public int note1, note2; Student(String nom, int note1, int note2){ this.nom = nom; this.note1 = note1; this.note2 = note2; } public int calc_moy(){ return (note1 + note2)/2; } public void show(){ System.out.println("Étudiant: "+ nom +" moyenne: "+ calc_moy()); } } public class Main{ public static void main(String []args){ Scanner in = new Scanner(System.in); System.out.println("Entrez le nom: "); String nom = in.nextLine(); System.out.println("Entrez les notes de deux matiéres: "); int note1 = in.nextInt(); int note2 = in.nextInt(); Student s = new Student(nom, note1, note2); s.show(); in.close(); } } Exercice 4:Effectuez une opération d'addition sur des nombres complexes à l'aide d'une classe Java appelée « Complex ». Le programme doit demander la partie réelle et imaginaire de leur somme.Exemple:Premier nombre Entrez la partie réelle: 1 Entrez la partie imaginaire: 4 Deuxième nombre Entrez la partie réelle: 2 Entrez la partie imaginaire: 3 La somme est 3 + 7iimport java.util.*; public class Complex{ double real, img; Complex(double r, double i){ this.real = r; this.img = i; } public static Complex somme(Complex c1, Complex c2) { Complex c = new Complex(0, 0); c.real = c1.real + c2.real; c.img = c1.img + c2.img; return c; } public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Premier nombre"); System.out.println("Entrez la partie réelle: "); double r1 = in.nextDouble(); System.out.println("Entrez la partie imaginaire: "); double i1 = in.nextDouble(); Complex c1 = new Complex(r1, i1); System.out.println("Deuxième nombre"); System.out.println("Entrez la partie réelle: "); double r2 = in.nextDouble(); System.out.println("Entrez la partie imaginaire: "); double i2 = in.nextDouble(); Complex c2 = new Complex(r2, i2); // calculer la somme des deux nombres complexes c1 et c2 Complex c = somme(c1, c2); System.out.printf("La somme est: "+ c.real+" + "+ c.img +"i"); } } Exercice 5:Créez une classe appelée « Point ». Cette classe doit avoir 2 entiers (x, y) en tant que membres privés. Le constructeur doit définir les valeurs de x et y via des paramètres. La classe doit avoir une méthode publique appelée « distance ».

Cela prend un seul paramètre(Point) et renvoie la distance entre les 2 points.Exemple:P1 (5,6) P2 (3,2) La distance entre P1 et P2 est : 4.47214import java.lang.Math; class Point { private double x; private double y; Point(double x, double y) { this.x = x; this.y = y; } public double getX() { return x; } public double getY() { return y; } public double distance(Point p) { double px = this.getX() - p.getX(); double py = this.getY() - p.getY(); return Math.sqrt(px * px + py * py); } } public class Test { public static void main(String []args){ Point p1 = new Point(5, 6); Point p2 = new Point(3, 2); System.out.println("P1 ("+ p1.getX() +","+ p1.getY() +")"); System.out.println("P2 ("+ p2.getX() +","+ p2.getY() +")"); System.out.println("La distance entre P1 et P2 est : "+p1.distance(p2)); } }report this ad report this ad Java exercices corrigés pour maitriser le langage Java, conçue pour les informaticiens, vous pratiquerez divers concepts du langage de programmation Java. Vous commencerez par des exercices Java de base à des exercices plus avancés. La solution est fournie pour chaque exercice.
Vous devez essayer de résoudre chaque problème avant de vérifier la solution. Si vous avez des questions concernant chaque problème, n'hésitez pas à nous écrire. Vous pouvez utiliser l'éditeur Java suivant pour résoudre les exercices suivants: (Cliquez sur l'onglet input si vous souhaitez entrer des valeurs, cliquez sur Run pour exécuter votre programme, le résultat sera affichée sur l'onglet output).Écrivez une classe « Rectangle » ayant deux variables « a » et « b » et une fonction membre « surface() » qui retournera la surface du rectangle. Corrigé/preview/button/#27ae60 Écrivez une classe « Somme » ayant deux variables « n1 » et « n2 » et une fonction membre « sum() » qui calcule la somme. Dans la méthode principale main demandez à l'utilisateur d'entrez deux entiers et passez-les au constructeur par défaut de la classe « Somme » et afficher le résultat de l'addition des deux nombres.Corrigé/preview/button/#27ae60 Écrivez une classe Java appelée « Student » avec les membres suivant :nom (de type String),note1, note2 (de type int)Le programme demande à l'utilisateur d'entrer le nom et les notes. calc_moy() calcule la note moyenne et show() affiche le nom et la note moyenne.Corrigé/preview/button/#27ae60Effectuez une opération d'addition sur des nombres complexes à l'aide d'une classe Java appelée « Complex ». Le programme doit demander la partie réelle et imaginaire de deux nombres complexes et afficher les parties réelle et imaginaire de leur somme.Entrez la partie réelle: 1Entrez la partie imaginaire: 4Entrez la partie réelle: 2Entrez la partie imaginaire: 3Corrigé/preview/button/#27ae60 Créez une classe appelée « Point ». Cette classe doit avoir 2 entiers (x, y) en tant que membres privés. Le constructeur doit définir les valeurs de x et y via des paramètres. La classe doit avoir une méthode publique appelée « distance ». Cela prend un seul paramètre(Point) et renvoie la distance entre les 2 points.La distance entre P1 et P2 est : 4.47214Corrigé/preview/button/#27ae60Créez une classe « Person »Créez une classe « Student » et une autre classe « Teacher », les deux héritent de la classe « Person ».La classe « Student » aura une méthode publique « GoToClasses », qui affichera à l'écran « I'm going to class.
».La classe « Teacher » aura une méthode publique « Explain », qui affichera à l'écran « Explanation begins ». En plus, il aura un attribut privé « subject » de type string.La classe « Person » doit avoir une méthode « SetAge(int n) » qui indiquera la valeur de leur âge (par exemple, 15 years old).La classe « Student » aura une méthode publique « DisplayAge » qui écrira sur l'écran « My age is: XX years old ».Vous devez créer une autre classe de test appelée « Test » qui contiendra « Main » et:Créez un objet Person et faites-lui dire « Hello »Créer un objet Student, définir son âge à 15 ans, faites-lui dire « Hello », « I'm going to class. » et afficher son âgeCréez un objet Teacher, 40 ans, demandez-lui de dire « Hello » puis commence l'explication.Corrigé/preview/button/#27ae60 Créez une classe « House », avec un attribut « surface », un constructeur qui définit sa valeur et une méthode « Display » pour afficher « Je suis une maison, ma surface est de XXX m2 » (XXX: la valeur de surface).
Incluez aussi des getters et des setters pour la surface.La classe « House » contiendra une porte (Door). Chaque porte aura un attribut « color » (de type String), et une méthode « Display » qui affichera « Je suis une porte, ma couleur est bleu » (ou quelle que soit la couleur). Inclure un getter et un setter. Créez également la méthode « GetDoor » dans la classe « House ».La classe « Apartment » est une sous-classe de la classe « House », avec une surface prédéfinie de 50m2.Créez également une classe Person, avec un nom (de type String). Chaque personne aura une maison. La méthode « Display » pour une personne affichera son nom, les données de sa maison et les données de la porte de cette maison.Écrivez un Main pour créer un Apartment, une personne pour y vivre et pour afficher les données de la personne.Je m'appele Thomas.Je suis un appartement, ma surface est 50 m2Je suis une porte, ma couleur est blue. des notions UML à savoir : La composition peut être considérer comme une relation "fait partie de", c'est à dire que si un objet Y fait partie d'un objet X alors Y ne peut pas exister sans X. Ainsi si X disparaît alors Y également. L'agrégation peut être considérer comme une relation de type "a un", c'est à dire que si un objet X a un objet Y alors Y peut vivre sans X. Corrigé/preview/button/#27ae60