# Exercice 1 : nombre pair ou impair : condition if

Ecrivez un programme Java qui lit un nombre et indique s'il est positif, négatif ou s'il vaut zéro et s'il est pair ou impair.  
Exemple d'exécution:

Entrez un nombre entier: 5

Le nombre est positif et impair

Entrez un nombre entier: -4

Le nombre est négatif et pair

Entrez un nombre entier: 0

Le nombre est zéro (et il est pair)

# Exercice 2 : Héritage et interfaces

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 Euros.
* Ceux affectés à la Représentation. Leur salaire mensuel est également le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus 800 Euros.
* Ceux affectés à la Production. Leur salaire vaut le nombre d'unités produites mensuellement multipliées par 5.
* Ceux affectés à la Manutention. Leur salaire vaut leur nombre d'heures de travail mensuel multipliées par 65 euros.

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 :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **class** Salaires {  **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] 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](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Le salaire moyen dans l'entreprise est de " + p.salaireMoyen() + " euros.");  }  } |

Vous devriez obtenir quelque chose comme :

Le vendeur Pierre Business gagne 6400.0 euros.

Le représentant Léon Vendtout gagne 4800.0 euros.

Le technicien Yves Bosseur gagne 5000.0 euros.

Le manut. Jeanne Stocketout gagne 2925.0 euros.

Le technicien Jean Flippe gagne 5200.0 euros.

Le manut. Al Abordage gagne 3125.0 euros.

Le salaire moyen dans l'entreprise est de 4575.0 euros.

# Exercice 3

Vous trouverez ci-dessous le programme Alphabet qui implémente la hiérarchie de 3 classes (A, B et C). Indiquez l'affichage du programme (sans l'exécuter). Attention aux situations de shadowing.

**class** Alphabet {

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) args[]) {

A[] as = **new** A[3];

as[0] = **new** A(1);

as[1] = **new** B(2);

as[2] = **new** C(3);

**for** (**int** i = 0; i < as.length; i++) {

as[i].afficherClasse();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("-----");

}

**for** (**int** i = 0; i < as.length; i++) {

as[i].afficherVariables();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("-----");

}

}

}

**class** A {

**protected** **int** a = 5;

**public** A(**int** a) {

**this**.a = a;

}

**public** **void** afficherClasse() {

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Classe A");

}

**public** **void** afficherVariables() {

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("a = " + a);

}

}

**class** B **extends** A {

**protected** **int** b = 6;

**public** B(**int** b) {

**super**(2 \* b);

a = b;

}

**public** **void** afficherClasse() {

**super**.afficherClasse();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Classe B");

}

**public** **void** afficherVariables() {

**super**.afficherVariables();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("b = " + b);

}

}

**class** C **extends** B {

**protected** **int** b = 7;

**protected** **int** c = 8;

**public** C(**int** c) {

**super**(3 \* c);

b = c;

}

**public** **void** afficherClasse() {

**super**.afficherClasse();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Classe C");

}

**public** **void** afficherVariables() {

**super**.afficherVariables();

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("c = " + c);

}

}

# Exercice 4

Chacune des méthodes methode1 à methode8 dans le programme ErreursMethodes.java ci-dessous contient une seule erreur syntaxique. Il s'agit d'erreurs liées à l'utilisation des paramètres, des variables ou de la valeur retournée par la méthode. Il n'y pas d'erreurs dans la méthode main, ni dans la méthode methodeSansErreurs. Trouvez et corrigez les erreurs !

Si vous sollicitez l'aide du compilateur pour trouver les erreurs, sachez qu'il aura de la peine avec ce type de programme où il y a beaucoup d'erreurs de structure. Il peut vous montrer la ligne approximative où se trouve une erreur, mais la solution proposée n'est pas toujours correcte, la liste des erreurs n'est pas toujours complète et les erreurs ne sont pas présentées dans le bon ordre. Dans ce genre de situation, utilisez surtout votre bon sens et vos propres connaissances de la syntaxe de Java.

**class** ErreursMethodes {

**public** **static** **void** main ([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {

**int** i1 = methode1();

**int** i2 = methode2();

**int** i3 = methode3();

**int** i4 = methode4();

methode5();

methode6();

methode7();

methode8();

}

**static** **int** methode1 {

**int** a = 0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 1");

**return** a;

}

**static** **int** methode2 () {

**int** a = 0;

i1 = 10;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 2");

**return** a;

}

**static** **int** methode3 () {

**int** a = 0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 3");

}

**static** **int** methode4 () {

[String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) a = "0";

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 4");

**return** a;

}

**static** **void** methode5 () {

**double** a = 0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 5");

**return** a;

}

**static** methode6 () {

**double** a = 0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 6");

**return** a;

}

**static** **void** methode7 () {

**int** a = 0;

**double** b = 5.5;

methodeSansErreur(a);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 7");

}

**static** **void** methode8 () {

**int** a = 0;

[String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) b = "5.5";

methodeSansErreur(a, b);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Méthode 8");

}

**static** **void** methodeSansErreur (**int** a, **double** b) {

*// Cette méthode ne fait rien du tout*

}

}

# Exercice 5

Le programme Portee.java ci-dessous sert à illustrer un point important qui concerne la portée des variables dans un programme. Sans exécuter le programme, indiquez tout ce que le programme affichera à l'écran lors de l'exécution.

**class** Portee {

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {

**int** x = 11, y = 22;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("1. x = " + x);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("2. y = " + y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println();

calculer1(x, y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("5. x = " + x);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("6. y = " + y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println();

calculer2(x, y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("11. x = " + x);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("12. y = " + y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println();

**int** [] object = {5};

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("13. object [0] = " + object [0]);

calculer3(object);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("15. object [0] = " + object [0]);

}

**static** **void** calculer1(**int** x, **int** y) {

x = 878;

y++;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("3. x = " + x);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("4. y = " + y);

}

**static** **void** calculer2(**int** a, **int** b) {

**int** x = 33;

**int** y = 44;

a = 2 \* x;

b = 2 \* y;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("7. x = " + x);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("8. y = " + y);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("9. a = " + a);

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("10. b = " + b);

}

**static** **void** calculer3(**int** [] object) {

object [0] = 37;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("14. object [0] = " + object [0]);

}

}

# Exercice 6

Ecrivez un programme Rectangle.java, permettant de lire la largeur et la hauteur d'un rectangle. Ce programme devra afficher, à la demande, le périmètre ou la surface du rectangle. Exemples d'exécutions:

Entrez la largeur: ...

Entrez la hauteur: ...

Surface ('s/S') ou perimètre ('p/P')?: ...

La surface est ...

ou encore:

Entrez la largeur: ...

Entrez la hauteur: ...

Erreur: vous avez introduit une largeur ou une hauteur négative!

Le programme devra être bien modularisé de sorte à ce que la méthode main se présente comme suit:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) args[]) {  **double** largeur = lireDonnee("largeur");  **double** hauteur = lireDonnee("hauteur");    **boolean** donneesOk = testerDonnees(largeur, hauteur);    **if** (donneesOk) {  calculer(largeur, hauteur);  } **else** {  afficherErreur();  }  } |

Le test sur le type de calcul se fera au moyen d'un switch.

# Exercice 7

Implémentez un algorithme vous permettant de permuter les valeurs de deux variables.

Exemple d'exécution:

Entrez x: 1

Entrez y: 2

Avant permutation:

x : 1

y : 2

Après permutation:

x : 2

y : 1

# Exercice 8

**class** Moyenne1{

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {

**double** noteProgrammation = 6.0;

**double** noteAlgebre = 5.0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Ma moyenne est " + (noteProgrammation++ + noteAlgebre) / 2.0 );

}

}

**class** Moyenne2{

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {

**double** noteProgrammation = 6.0;

**double** noteAlgebre = 5.0;

[System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Ma moyenne est " + (++noteProgrammation + noteAlgebre) / 2.0 );

}

}

Expliquez pourquoi la valeur retournée est différente d’un programme à l’autre.

# Exercice 9

Écrivez un petit programme Java, Degre3.java, vous permettant d'évaluer un polynôme du 3ème degré de la forme:  
  
((*a*+*b*)/2)*x*3 + (*a*+*b*)2*x*2 + *a* + *b* + *c*   
  
Exemple d'exécution:

Entrez a (int) : 1

Entrez b (int) : 2

Entrez c (int) : 3

Entrez x (double) : 3.5

La valeur du polynôme est : 180.5625

Notez qu'il n'y a pas d'opérateur simple pour la puissance 3 en Java. Il faut utiliser des multiplications. Vous introduirez également au besoin des variables auxiliaires pour rendre le calcul plus efficace.

# Exercice 10

Soit le programme suivant :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | **class** ConcatIncorrecte  {  **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args)  {  [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) s = "China Blue";  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println(s);  concat(s, " Express");  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println(s);    }      **public** **static** **void** concat([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) s, [String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) s2)  {  s +=s2;  }  } |

1. Expliquez pourquoi la méthode concatener ne parvient pas à modifier la chaîne s du main (en y concaténant " Express")
2. Corriger le codage de la méthode concatener et son utilisation dans le main de sorte à ce que l'exécution du programme affiche :

China Blue

China Blue Express

au lieu de :

China Blue

China Blue

# Exercice 11

Cet exercice, permet de travailler les classes et les méthodes abstraites, les conversions de type explicites et votre compréhension générale des hiérarchies de classes.   
  
Votre grand-mère a besoin d'un programme Java pour gérer sa collection d'ustensiles de cuisine anciens, plus précisément des assiettes (rondes ou carrées) et des cuillères. Vous trouverez ci-dessous une version incomplète de son programme Collection. Les 5 objets créés sont stockés dans le tableau us qui est de type Ustensile[].  
  
Il vous est demandé de compléter le programme selon les indications dans les quatre questions ci-après. Pensez à profiter au maximum des possibilités offertes par Java pour éviter la duplication inutile d'instructions dans les classes et les méthodes.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | **class** [Collection](http://www.google.com/search?hl=en&q=allinurl%3Acollection+java.sun.com&btnI=I%27m%20Feeling%20Lucky) {    **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {  Ustensile[] us = **new** Ustensile[5];  us[0] = **new** AssietteRonde(1926, 8.4);  us[1] = **new** Cuillere(1881, 7.3);  us[2] = **new** AssietteCarree(1935, 5.6);  us[3] = **new** Cuillere(1917, 8.8);  us[4] = **new** AssietteRonde(1837, 5.4);    afficherCuilleres(us);  afficherSurfaceAssiettes(us);  afficherValeurTotale(us);  }    **static** **void** afficherCuilleres(Ustensile[] us) {  **int** nbCuilleres = 0;  **for** (**int** i = 0; i < us.length; i++) {  *// A compléter*  }  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Il y a " + nbCuilleres + " cuillères.");  }    **static** **void** afficherSurfaceAssiettes(Ustensile[] us) {  **double** somme = 0;  **for** (**int** i = 0; i < us.length; i++) {  *// A compléter*  }  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Surface totale des assiettes : " + somme);  }    **static** **void** afficherValeurTotale(Ustensile[] us) {  **double** somme = 0;  **for** (**int** i = 0; i < us.length; i++) {  *// A compléter*  }  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Valeur totale de la collection : " + somme);  }  } |

### Hiérarchie de classes

Il convient de modéliser les objets de la collection avec une hiérarchie de 5 classes comme indiqué dans la liste ci-dessous. Ecrivez le code de cette hiérarchie de classes, y compris les variables d'instance et les constructeurs. Vous pouvez ajouter les nouvelles classes au fichier Collection.java, si vous voulez, ou bien utiliser des fichiers séparés.   
  
Evitez de dupliquer inutilement des instructions. La méthode main du programme Collection ci-dessus vous montre la façon d'appeler les constructeurs des classes qui sont instanciables.

1. Chaque Ustensile a une annee de fabrication qui est une valeur int. Un Ustensile est soit une Assiette, soit une Cuillere. Il n'est pas possible d'instancier la classe Ustensile.
2. Une Assiette est soit une AssietteRonde, soit une AssietteCarree. Il n'est pas possible d'instancier la classe Assiette.
3. Chaque AssietteRonde à un rayon qui est une valeur double.
4. Chaque AssietteCarree à un cote qui est une valeur double.
5. Chaque Cuillere à une longueur qui est une valeur double.

### Comptage

Complétez le code de la méthode afficherCuilleres pour qu'elle calcule et affiche le nombre d'objets de type Cuillere qui sont stockés dans le tableau us. Par exemple, le code de la méthode main ci-dessus donne lieu à l'affichage suivant:

Il y a 2 cuillères.

### Surface totale

Complétez le code de la méthode afficherSurfaceAssiettes pour qu'elle calcule et affiche la somme des surfaces de toutes les assiettes stockées dans le tableau us, c'est-à-dire les assiettes rondes et les assiettes carrées. Il sera nécessaire de compléter la hiérarchie des classes avec des méthodes de calcul de surface. Evitez de dupliquer inutilement des instructions.  
  
La surface d'une assiette se calcule comme suit:

1. AssietteRonde : 3.14 \* rayon \* rayon
2. AssietteCarree : cote \* cote

Par exemple, le code de la méthode main ci-dessus donne lieu à l'affichage suivant:

Surface totale des assiettes : 344.48080000000004

### Valeur totale des ustensiles

Complétez le code de la méthode afficherValeurTotale pour qu'elle calcule et affiche la somme des valeurs de tous les ustensiles stockés dans le tableau us. Il sera nécessaire de compléter la hiérarchie des classes avec des méthodes de calcul de valeur. Evitez de dupliquer inutilement des instructions.  
  
La valeur d'un ustensile se calcule comme suit:

1. Cuillere et AssietteRonde : Si l'ustensile a moins de 50 ans, il vaut 0 euros. Sinon, il vaut 1 franc pour chaque année qu'il a de plus de 50 ans, c'est-à-dire (2009 - annee - 50). Par exemple, une AssietteRonde ou une Cuillere fabriquée en 1943 vaut 16 euros.
2. AssietteCarree : 5 fois la valeur qu'elle aurait eue en étant ronde (car les assiettes carrées sont plus rares). Par exemple, une AssietteCarree fabriquée en 1943 vaut 5 \* 16 euros.

Par exemple, le code de la méthode main ci-dessus donne lieu à l'affichage suivant:

Valeur totale de la collection : 395.0

# Exercices 12

Vous vous intéressez dans cet exercice à décrire les données d'un jeu simulant des combats de magiciens.

Dans ce jeu, il existe trois types de cartes : les terrains, les créatures et les sortilèges.

* Les terrains possèdent une couleur (parmi 5 : blanc('B'), bleu ('b'), noir ('n'), rouge ('r') et vert ('v').)
* Les créatures possèdent un nom, un nombre de points de dégâts et un nombre de points de vie.
* Les sortilèges possèdent un nom et une explication sous forme de texte.

De plus, chaque carte, indépendamment de son type, possède un coût. Celui d'un terrain est 0.

Dans un programme Magic.java, proposez (et implémentez) une hiérarchie de classes permettant de représenter des cartes de différents types.

Chaque classe aura un constructeur permettant de spécifier la/les valeurs de ses attributs. De plus, chaque constructeur devra afficher le type de la carte.

Le programme doit utiliser la conception orientée objet et ne doit pas comporter de duplication de code.

Ajoutez ensuite aux cartes une méthode afficher() qui, pour toute carte, affiche son coût et la valeur de ses arguments spécifiques.

Créez de plus une classe Jeu pour représenter un jeu de cartes, c'est-à-dire une collection de telles cartes.   
Cette classe devra avoir une méthode piocher permettant d'ajouter une carte au jeu. On supposera qu'un jeu comporte au plus 10 cartes. Le jeu comportera également une méthode joue permettant de jouer une carte. Pour simplifier, on jouera les cartes dans l'ordre où elles sont stockées dans le jeu, et on mettra la carte jouée à null dans le jeu de cartes.

Pour finir, dans la méthode main, constituez un jeu contenant divers types de cartes et faites afficher le jeu grâce à une méthode afficher propre à cette classe.

Par exemple la méthode main pourrait ressembler à quelque chose comme cela  :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **class** Magic {  **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky)[] args) {  Jeu maMain = **new** Jeu(10);    maMain.piocher(**new** Terrain('b'));  maMain.piocher(**new** Creature(6, "Golem", 4, 6));  maMain.piocher(**new** Sortilege(1, "Croissance Gigantesque",  "La créature ciblée gagne +3/+3 jusqu'à la fin du tour"));    [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println("Là, j'ai en stock :");  maMain.afficher();  maMain.joue();  }  } |

qui produirait quelque chose comme :

On change de main

Un nouveau terrain.

Une nouvelle créature.

Un sortilège de plus.

Là, j'ai en stock :

Un terrain bleu

Une créature Golem 4/6

Un sortilège Croissance Gigantesque

Je joue une carte...

La carte jouée est :

Un terrain bleu

# Exercice 13

Ecrivez un programme Geometrie qui permet à l'utilisateur d'entrer les coordonnées (*x*, *y*) des sommets d'un triangle. Le programme affiche ensuite le périmètre du triangle ainsi qu'un message indiquant s'il s'agit d'un triangle isocèle. Votre programme doit être orienté objets. Indications:

* Un triangle est isocèle si au moins deux côtés ont la même longueur.
* La formule pour calculer la distance entre deux points (*x*1, *y*1) et (*x*2, *y*2) est: racine carrée de (*x*1 - *x*2)2 + (*y*1 - *y*2)2.
* Java met à disposition la méthode Math.sqrt() pour calculer la racine carrée. Cette méthode prend un nombre non-négatif en paramètre. Exemple:
* double a = Math.sqrt(9.0); // la valeur 3.0 sera affectée à a

Exemple d'affichage du programme pour un triangle isocèle:

Construction d'un nouveau point

Veuillez entrer x : 0

Veuillez entrer y : 0

Construction d'un nouveau point

Veuillez entrer x : 2.5

Veuillez entrer y : 2.5

Construction d'un nouveau point

Veuillez entrer x : 0

Veuillez entrer y : 5

Périmètre : 12.071067811865476

Le triangle est isocèle

Dans cet exercice, vous élaborerez un programme orienté objets de manière indépendante pour la première fois. Si vous n'avez pas le temps de faire cet exercice, n'oubliez pas d'en étudier le corrigé.  
  
Voici quelques indications en vrac qui peuvent vous être utiles. Ne les lisez pas si vous voulez être complètement indépendant ...

* Réfléchissez aux objets que vous aimeriez utiliser dans le programme. Vous pourriez par exemple représenter le triangle par une classe Triangle et ses points par une classe Point. Une troisième classe Geometrie pourrait héberger la méthode main.
* Réfléchissez aux variables et méthodes d'instance qui seraient utiles pour les classes Triangle et Point.
* Un objet de type Point a typiquement les coordonnées x et y. Un objet de type Triangle a trois sommets qui peuvent être représentés par des objets de type Point.
* Les coordonnées des points peuvent par exemple être entrées dans la méthode constructeur de la classe Point à l'aide de la méthode scanner.nextDouble(). Dans ce cas, la méthode constructeur n'a pas de paramètres.
* Le périmètre d'un triangle peut être calculé comme la somme des distances entre les trois sommets.
* Une méthode calculerDistance() peut être utile pour calculer la distance entre deux points. La formule est indiquée ci-dessus. Si cette méthode se trouve dans la classe Point, le premier point est accessible sous forme de l'objet utilisé pour appeler la méthode (l'objet actuel this), mais il faut envoyer le deuxième point en paramètre. Si les variables d'instance x et y du deuxième point sont privées, il faut mettre à disposition des méthodes getX() et getY() afin de pouvoir accéder aux valeurs nécessaires pour le calcul.

# Exercice 14

**package** fr.formation.module1.exo14;

**public** **class** Egalite {

**public** **static** **void** main(String[] argv) {

Compteur c1, c2, c3;

c1 = **new** Compteur(0);

c1.incremente();

c2 = **new** Compteur(1);

c3 = c1;

**if** (c1 == c3) {

System.***out***.println(" c1 et c3 sont égaux ");

} **else** {

System.***out***.println(" c1 et c3 ne sont pas égaux ");

}

**if** (c1.getValeur() == c2.getValeur()) {

System.***out***.println(" c1 et c2 ont même valeur ");

} **else** {

System.***out***.println(" c1 et c2 n’ont pas la même valeur ");

}

**if** (c1 == c2) {

System.***out***.println(" c1 et c2 sont égaux ");

} **else** {

System.***out***.println(" c1 et c2 ne sont pas égaux ");

}

**if** (c1.getValeur() == c1.incremente().getValeur()) {

System.***out***.println(" c1 et c1 incremente ont même valeur ");

} **else** {

System.***out***.println(" c1 et c1 incremente n’ont pas la même valeur ");

}

**if** (c1 == c1.incremente()) {

System.***out***.println(" c1 et c1 incremente sont égaux ");

} **else** {

System.***out***.println(" c1 et c1 incremente ne sont pas égaux ");

}

}

}

**class** Compteur {

**int** x;

Compteur(**int** n) {

x = n;

}

Compteur incremente() {

x++;

**return** **this**;

}

**int** getValeur() {

**return** x;

}

}

Essayez de prédire le résultat de l’exécution de ce programme.