La programmation par Objet

Les classes et les Objets

Exercice 1 — *Utilisation d'une classe*

Voici le texte d'une classe représentant de façon sommaire un compte bancaire et les opérations bancaires courantes.

```
public class Compte{
  int solde = 0;

  void depot(int montant){
    solde = solde + montant;
  }

  void retrait(int montant){
    solde = solde −montant;
  }

  void virement(int montant, Compte autre){
    autre.retrait(montant);
    this.depot(montant);
   }

  void afficher(){
    System.out.println("solde: "+ solde);
  }
}
```

1. Comment fonctionne la méthode `virement` ? Combien d'objets `Compte` fait-elle intervenir ?
2. Créez deux comptes que vous affecterez à deux variables. Ecrivez le code correspondant aux opérations suivantes :
   — dépot de 500 euros sur le premier compte.
   — dépot de 1000 euros sur le second compte.
   — retrait de 10 euros sur le second compte.
   — virement de 75 euros du premier compte vers le second.
   Vous mettrez le code java correspondant à cette question dans la méthode `main` d'une nouvelle classe appelée `TesteCompte`.
3. Créez un tableau de dix objets `Compte`. Pour cela, notez bien qu'il faut d'abord créer le tableau puis créer successivement les dix objets `Compte` à mettre dans les dix cases de ce tableau.
   Dans chaque case, faites un dépot de 200 euros plus une somme égale à 100 fois l'indice du compte dans le tableau.
   Enfin, vous afficherez les soldes de tous les comptes.

Exercice 2 — *Constructeurs*

Cet exercice reprend la classe Compte de l'exercice précédent.

1. Complétez la classe `Compte` avec une information supplémentaire : le nom du titulaire du compte (type `String`). Vous modifierez la méthode d'affichage pour qu'elle affiche cette information.
2. Créez un constructeur pour la classe `Compte`. Ce constructeur doit prendre en paramètre le nom du titulaire du compte. Donnez le code de création d'un objet `Compte` qui appelle ce constructeur.

Exercice 3 — *Compréhension de programme*

Q'affiche le programme suivant ?

```
public class Exercice8 {
  static int var = 0;

  public static void main(String[] args){
    Exercice8 obj = new Exercice8();
    obj.methode(); }

  public Exercice8(){
    this(15); }

  public Exercice8(int var){
   this.var = var; }

  void methode(){
    System.out.println("var d'instance var=" + var); }
}
```

Exercice 4 — *Compréhension de programme*

Q'affiche le programme suivant ?

```
public class Exercice9 {
   public static void main(String[] args){
     ClasseA objA = new ClasseA();
     ClasseB objB = new ClasseB();
     objA.enregistrer(objB);
     objB.rappeler();
     objA.println();
  }
}

class ClasseA{
  int var;
  void enregistrer(ClasseB obj){
    obj.enregistrer(this); }
  void rappeler(int var){
    this.var = var; }
  void println(){
    System.out.println("variable d'instance var = " + var); }
}

class ClasseB{
  ClasseA ref;
   void enregistrer(ClasseA var){
     ref = var; }
   void rappeler(){
     ref.rappeler(15); }
}
```

Exercice 5 — *Compréhension de programme*

Q'affiche le programme suivant ?

```
class Compteur {
   int x ;

   Compteur ( int n ) {
     x = n ; }

   Compteur incremente ( ) {
     x++ ;
     return this; }

   int getValeur ( ) {
     return x ; }
}

class Exercice10 {
   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␣" ) ; }
   }
}
```

Exercice 6 — *Conception d'un zoo*

Dans cet exercice, on va créer un zoo avec différents animaux. Chaque animal est caractérisé par son `espece` et son `nom` qui sont des chaînes de caractères.

(1) Créez la classe `Animal` qui comporte les deux attributs décrits, de type `String`. Vous coderez un constructeur qui prend en paramètre l'espèce et le nom de l'animal, ainsi que les accesseurs et transformateurs des deux attributs. Dans cette classe, vous créerez une méthode d'affichage `public void affichage()`, qui affiche à l'écran l'`espece` et le `nom` d'un `Animal`.

(2) Créez une classe `Zoo` qui caractérise un zoo qui est composé d'un tableau `zoo` dont les éléments sont de type `Animal`, d'un `nom`, d'une capacité maximale `capacite`, et d'une capacité courante `nbAnimaux`. Vous coderez les méthodes suivantes :
- `public Zoo(String nom, int capacite)` qui crée une instance de `Zoo`,
- `public void ajouterAnimal(Animal a)` qui ajoute un l'`Animal a` au `zoo`, et qui renvoie un message d'erreur si la capacité maximale du zoo est atteinte,
- `public void afficherZoo()` qui affiche chaque `Animal` présent dans `zoo`.

(3) Créez une classe `ZooMain`, où vous exécuterez les instructions suivantes :
   — Déclarez 4 variables de type `Animal` que vous instancierez pour chacune des espèces `Chien`, `Vache`, `Perroquet` et `Herisson`.
   — Déclarez une variable de type `Zoo`, instanciez la et ajoutez lui les 4 animaux que vous venez de créer.
   — Affichez le `Zoo` ainsi créé.

Exercice 7 — *Agenda*

Un agenda est vu comme un tableau de 52 semaines, chaque semaine étant composée de 7 jours eux mêmes divisés plages horaires, le nombre de plages horaires n'étant pas déterminé. Chaque plage peut contenir un message sous la forme d'une chaîne de caractères. Définir la classe agenda qui a les caractéristiques suivantes :

1. Le nombre de plages horaires est déterminé à la construction de chaque instance
2. Inclure une méthode pour noter un message sur cet agenda
3. Ajouter une méthode pour dupliquer un même message chaque semaine le même jour à la même plage.
4. Ajouter une méthode pour afficher le planning d'une semaine donnée
5. Ajouter une méthode pour afficher le planning d'un jour donné et d'une semaine donnée
6. coder une méthode `main` qui :
   — Crée une instance d'agenda avec 6 plages horaires
   — Note un "RDV" en plage 4 le 6ème jour de la 26 ème semaine
   — Duplique un message un jour et une plage donnée de chaque semaine
   — Affiche le planning d'une semaine donnée
   — Affiche le planning d'un jour donné
   — Affiche le message d'une plage donnée