

NOM

Prénom

Groupe

Éléments de programmation
par objets avec Java

Li230

Contrôle continu écrit

Durée : **0h30** environ - Tous les documents *personnels*, manuscrits ou imprimés, sont autorisés. Tous les appareils électroniques et téléphones portables sont interdits.

NE RIEN ÉCRIRE AU CRAYON NI À L'ENCRE ROUGE S.V.P.

A QCM

Pour chacune des questions qui suivent, plusieurs affirmations sont proposées en guise de réponses. Parmi celles-ci, **0, 1 ou plusieurs** peuvent être vraies. Vous devez cocher la case figurant devant chaque affirmation vraie. Toute rature susceptible de donner un caractère ambigu à votre réponse sera considérée comme une erreur. Chaque question est notée indépendamment des autres. Le barème appliqué sera le suivant :

1 point attribué **si tout est juste**,
0 point sinon.

a1 On considère la classe suivante

```
public class Test {
    public static void main(String[] a) {
        int k = 3;
        System.out.println("k = " + k.toString());
    }
}
```

Cocher les affirmations justes :

- Le programme affiche k = 3
- Il y a erreur de compilation
- System.out est une variable de la classe System

a2 On lance l'exécution du programme composé des deux classes suivantes :

```
public class Personne {
    private String nom;
    private String prénom;
    int nbPersonnes;

    public Personne(String nom, String prénom) {
        this.nom = nom;
        this.prénom = prénom;
        nbPersonnes++;
    }
}
```

```

    public String toString(){
        return nom + " " + prénom;
    }
    public String getNom(){
        return nom;
    }
}

public class Test {
    public static void main(String[] a) {
        Personne p1 ,p2 ,p3, p4;
        p1= new Personne("Bebo","Alin");
        p2= new Personne("Bobe","Alon");
        p3= new Personne("Bibu","Alan");
        System.out.println(p3.nbPersonnes);
    }
}

```

Cocher les affirmations justes :

- Le programme affiche 3
- Le programme affiche 1
- Il y a erreur de compilation
- `nbPersonnes` devrait être déclaré `static`

a3 On remplace dans la méthode `main` la dernière instruction par `System.out.println(p1)` ;

Cocher les affirmations justes :

- le programme affiche `Personne@f5da06`
- le programme affiche `Bebo Alin`
- il y a erreur de compilation

a4 On remplace la méthode `main` par

```

public static void main(String[] a) {
    Personne p1 ,p2;
    String s ;
    p1 = new Personne("Bebo","Alin");
    p2 = p1;
    // Instruction manquante-----
    System.out.println(s);
}

```

Sachant que le programme affiche `Bebo` , quelle peut être l'instruction manquante ?

- `s= p2.getNom()` ;
- `s= p1.nom`;
- `s= p2.nom`;
- `s= p1.getNom()` ;

Contrôle continu écrit

Durée : **1h30** environ - Tous les documents *personnels*, manuscrits ou imprimés, sont autorisés. Tous les appareils électroniques et téléphones portables sont interdits.

NE RIEN ÉCRIRE AU CRAYON NI À L'ENCRE ROUGE S.V.P.

B Déclarations de classes : micro-simulation de comptes bancaires

On considère la banque CJ, qui gère des comptes de clients. Un compte est caractérisé par son numéro et son solde, et la banque par l'ensemble des comptes qu'elle gère. Un compte comporte également le nom du client, et la banque le numéro du prochain compte à créer.

Les champs de la classe `Compte` sont `nomClient` `numero` `solde`, et ceux de la classe `Banque` *un tableau de `Compte` et l'index du « nouveau compte à créer »*.

Les numéros de compte sont successifs, 1 2 3 4 5 ... Ils sont automatiquement attribués lors de la création d'un nouveau compte; on utilisera à cette fin un champ de classe de `Compte`, `nouvNumero`.

N.B. Vous pouvez écrire la classe `Tests` en vous contentant des signatures et des fonctionnalités des méthodes de la classe `Banque`: vous n'avez pas besoin de connaître le détail de leurs instructions.

Classe `Compte`

Elle comporte trois champs privés, deux constructeurs, trois méthodes publiques d'accès aux données, les deux méthodes `depot` et `retrait`, et la redéfinition de la méthode `toString()` pour décrire le compte sous forme de chaîne de caractères.

- b1** Donner la déclaration des champs privés d'objet et de leurs méthodes d'accès.
- b2** Donner la déclaration de la variable de classe de `Compte`, initialisée à 1 pour tous les comptes.
- b3** Donner la déclaration du constructeur à deux paramètres permettant d'initialiser les trois champs d'objet du compte nouvellement créé, le numéro étant attribué à partir de la variable de classe.
- b4** Donner la déclaration du constructeur à un paramètre permettant d'initialiser le solde à 0.

Sur le compte, on peut bien sûr effectuer des dépôts (augmentation du solde) et des retraits, cette dernière opération (diminution du solde) n'étant possible que si le solde est supérieur ou égal au montant du retrait demandé.

- b5** Donner la déclaration de la méthode `depot` permettant d'effectuer sur le compte un dépôt d'un certain montant.
- b6** La méthode `retrait` renvoie `true` ou `false` selon que le retrait est ou non possible, et effectue l'opération dans le premier cas. Donner sa déclaration.

- b7** Il faut enfin pouvoir visualiser sous forme de texte (chaîne de caractères) les données du compte, avec la présentation à l'écran suivante :

```
4 Robert
solde : 1000.0
```

Donner la déclaration de la méthode `toString()` , qui fournit en résultat la chaîne de caractères décrivant le compte.

Classe Banque

Les divers comptes de la banque CJ sont stockés sous forme d'un tableau (à une dimension) de comptes, `clientele` . Un autre champ `nouvCompte` contient l'index dans ce tableau de la case apte à recevoir l'objet correspondant au prochain compte à créer. On suppose que la i^{e} case du tableau correspond au compte numéro i , et que l'on ne peut jamais supprimer de compte, seulement en ajouter de nouveaux.

- b8** Donner la définition complète des deux champs privés `clientele` et `nouvCompte` , sachant que `clientele` ne repère (désigne) aucun objet et que l'index du premier compte dans le tableau est 1.

- b9** On sait que tout objet tableau est de taille fixe tout au long de son existence (taille immuable), et que celle-ci doit être indiquée lors de la création de l'objet. Donner la définition du constructeur à un paramètre, la taille maximale du tableau de comptes, qui crée l'objet tableau de comptes.

La méthode publique `ajouteCompte` , crée « à la demande » un objet de type `Compte` et l'affecte à la bonne case du tableau.

- b10** On peut vouloir ouvrir un compte avec ou sans versement initial. Donner les deux définitions des deux méthodes `ajouteCompte` à un ou deux paramètres. Pour l'une des deux, penser à appeler l'autre méthode.

Lorsqu'un client arrive à sa banque CJ, il peut y prendre ou verser de l'argent., à condition d'indiquer son numéro de compte.

- b11** Donner la déclaration de la méthode `operation` , à trois paramètres, qui, pour un numéro de compte, effectue soit un dépôt soit un retrait en fonction de la valeur du paramètre booléen `verse` .

- b12** On peut aussi envisager une méthode `getCompte` , qui renvoie sous forme de chaîne de caractères tous les renseignements d'un compte indiqué en paramètre. Donner sa déclaration.

- b13** De nombreux clients arrivent sans connaître le numéro de leur compte : il convient d'ajouter à la classe `Banque` une méthode `getNum` , qui renvoie le numéro de compte d'un nom (de client) ou -1 lorsque le nom n'existe pas dans la liste des comptes existants. Donner la déclaration d'une telle méthode.

Classe Tests

Il convient de tester l'ensemble de ces outils avec une classe de lancement de l'exécution, qui essaiera les divers outils déclarés dans les classes `Compte` et `Banque`.

On se donnera un champ qui définisse la constante `MAXCLIENTS` comme ayant la valeur 10.

On crée un objet de type `Banque` ; on peut alors ajouter deux comptes, et on affiche alors à l'écran les caractéristiques de ces deux comptes. On ajoute alors deux nouveaux comptes. Après cela, on effectue successivement les opérations suivantes : versement de 3000 sur le compte numéro 2, versement de 200 sur le compte numéro 3, et retrait de 500 sur le compte dont le client a pour nom Albert. On affiche à l'écran les caractéristiques de ces quatre comptes (après les trois opérations).

On obtient l'affichage à l'écran suivant :

```
1 Albert
solde : 2000.0

2 Louise
solde : 0.0

1 Albert
solde : 1500.0

2 Louise
solde : 3000.0

3 Marie
solde : 4200.0

4 Robert
solde : 1000.0
```

b14 Donner la définition complète de la classe `Tests`.