4 avril 2012

Contrôle d'informatique no 3

Durée: 1 heure 45'

Nom:					• • • • • • • • • • • • • • • • • • • •	Groupe:		
Prénom:								
No sujet		1		2		3		
Nombre points		47 points	47 points		36 points		37 points	
_		_						

Remarque générale : toutes les questions qui suivent se réfèrent au langage de programmation Java (à partir du JDK 5.0) et les réponses doivent être rédigées à l'encre et d'une manière propre sur ces feuilles agrafées.

Remarques initiales:

- Il est conseillé de lire les sujets jusqu'à la fin, avant de commencer la rédaction de la solution.
- Les trois sujets peuvent être résolus (éventuellement) séparément, mais après avoir lu et compris l'ensemble des énoncés.
- Si vous ne savez pas implémenter une méthode, écrivez son en-tête, réduisez son corps à un simple commentaire et passez à la suite.

On considère un projet Java qui contient plusieurs classes et interfaces regroupées dans un package nommé *cms_ctr3*, à savoir :

- la classe *Travailleur* dont l'en-tête et le corps sont indiqués ci-dessous ;
- les interfaces *IDirigeable* et *IEquipable* dont les en-têtes et les corps sont indiqués cidessous ;

- la classe *Chef* qui dérive de la classe de base *Travailleur*, implémente les interfaces *IDirigeable* et *IEquipable* et que vous devez définir en fonction des consignes précisées au point 1;
- la classe *GrandChef* qui dérive de la classe de base *Chef* et que vous devez définir en fonction des consignes précisées au point 2 ;
- la classe principale *CP_Ctr3* dont le code source est donné au point **3** et vous devrez indiquer quels seront les résultats affichés dans la fenêtre console suite à son exécution.

Une instance de la classe *Travailleur* est un objet qui correspond à un travailleur caractérisé par son *nom* et par sa capacité à *travailler*. Voilà le code source de la classe *Travailleur* :

Vous ne devez rien modifier dans la classe *Travailleur* (ni dans son en-tête ni dans son corps). Cette classe sera la classe mère de la classe *Chef* définie au point 1 (et la classe "grand-mère" de la classe *GrandChef* définie au point 2).

Voilà le code source des interfaces *IDirigeable* et *IEquipable* :

```
package cms_ctr3;

public interface IDirigeable
{
         String diriger();
}//fin de l'interface IDirigeable
```

2

```
package cms_ctr3;

public interface IEquipable
{
    int faireEquipe(Travailleur tr);
    void afficherEquipe();
}//fin de l'interface IEquipable
```

Vous ne devez rien modifier dans les interfaces *IDirigeable* et *IEquipable* qui seront implémentées par la classe non abstraite *Chef* définie au point 1.

1. Le but de ce premier sujet est d'écrire le code source de la classe non abstraite *Chef* qui permet l'instanciation d'objets correspondant à des chefs qui sont des travailleurs "spéciaux" dont chacun dirige une équipe formée de travailleurs qui ne sont pas eux-mêmes des chefs.

A la création d'un objet de type *Chef*, on indique le nom du chef et la taille de son équipe. Si la taille proposée est négative ou zéro, l'équipe aura un seul membre. Les membres de l'équipe associée à un chef sont stockés dans un tableau de travailleurs qui, tout au début, ne contient aucun membre.

Comme tout travailleur, un chef peut *travailler* mais son travail sera différent de celui d'un simple travailleur. De plus, un chef peut aussi *diriger*.

La méthode *faireEquipe* permet à un chef d'ajouter un travailleur (qui n'est pas lui-même un chef) à son équipe.

La méthode afficher Equipe permet à un chef d'afficher les noms des membres de son équipe.

Plus précisément, la classe *Chef* fait partie du package *cms_ctr3*, est <u>publique</u>, <u>dérive</u> directement de la classe *Travailleur* (qui fait partie du même package) et <u>implémente</u> les interfaces *IDirigeable* et *IEquipable* (qui font partie du même package).

Dans le corps de la classe *Chef*, on définit 4 champs propres <u>sans modificateurs d'accès</u> <u>explicites</u>:

- un champ d'instance *equipe* de type tableau d'objets *Travailleur*, sans valeur initiale explicite;
- un champ d'instance *noEquipe* de type nombre entier, sans valeur initiale explicite ;
- un champ d'instance *jauge* de type nombre entier, avec la valeur initiale explicite θ ;
- un champ <u>statique</u> *nombreChefs* de type nombre entier, avec la valeur initiale explicite θ .

De plus, dans le corps de la classe *Chef*, on définit un constructeur public avec 2 arguments :

- le premier argument *nomChef* est de type chaîne de caractères (et il correspond au nom du chef);
- le deuxième argument *tailleEquipe* est de type nombre entier (et il correspond à la taille de l'équipe associée au chef).

Ce constructeur:

- stocke l'adresse de la chaîne de caractère correspondant à son premier argument dans le champ hérité *nom* par un appel au constructeur de la classe mère ;
- crée un tableau d'objets *Travailleur* et stocke son adresse dans le champ *equipe*, en procédant ainsi :
 - o si la valeur de son deuxième argument *tailleEquipe* est plus petite ou égale à zéro, le tableau créé aura la taille *1*;
 - o autrement, le tableau créé aura la taille indiquée par l'argument taille Equipe ;
- incrémente d'une unité la valeur du champ *nombreChefs* ;
- copie la nouvelle valeur du champ *nombreChefs* dans le champ *noEquipe*;
- affiche dans la fenêtre console le message :

XXX est un chef!

où XXX est la chaîne de caractères référencée par son premier argument.

En outre, dans le corps de la classe *Chef*, on redéfinit :

- la méthode d'instance *travailler* qui doit afficher dans la fenêtre console le message :

Le chef XXX dirige des travailleurs!

- où *XXX* est la chaîne de caractères correspondant au champ *nom* de l'objet appelant la méthode :
- la méthode d'instance *diriger* qui doit retourner la chaîne de caractères

Que l'équipe numéro YYY travaille!

- où YYY est la valeur stockée dans le champ no Equipe de l'objet appelant la méthode ;
- les méthodes *faireEquipe* et *afficherEquipe* selon les indications données ci-dessous.

La méthode d'instance redéfinie *faire Equipe* doit respecter les consignes suivantes :

- si l'argument de la méthode est une instance de la classe *Chef*, on affiche dans la fenêtre console le message

Pas de place pour deux chefs!

et on retourne la valeur -1;

- autrement, si la valeur du champ *jauge* de l'objet appelant est plus grande ou égale à la longueur du tableau désigné par le champ *equipe* de l'objet appelant, on affiche le message :

Equipe numéro YYY complète!

- où YYY est la valeur du champ noEquipe de l'objet appelant, et on retourne ensuite la valeur 1;
- autrement, on procède ainsi :
 - o on copie la valeur de l'argument de la méthode dans le tableau désigné par le champ *equipe* de l'objet appelant à la position précisée par le champ *jauge* de l'objet appelant ;
 - o on incrémente d'une unité le champ jauge de l'objet appelant ;
 - o on affiche dans la fenêtre console le message :

ZZZ ajouté à l'équipe numéro YYY!

- où **ZZZ** est la chaîne de caractère correspondant au champ *nom* de <u>l'objet</u> <u>argument</u> et **YYY** est la valeur du champ *noEquipe* de l'objet appelant ;
- o on retourne la valeur θ .

La méthode d'instance <u>redéfinie</u> *afficherEquipe* doit respecter les consignes suivantes :

- si la valeur du champ *jauge* de l'objet appelant est égale à zéro, on affiche dans la fenêtre console le message :

L'équipe numéro YYY est vide!

où YYY est la valeur du champ no Equipe de l'objet appelant ;

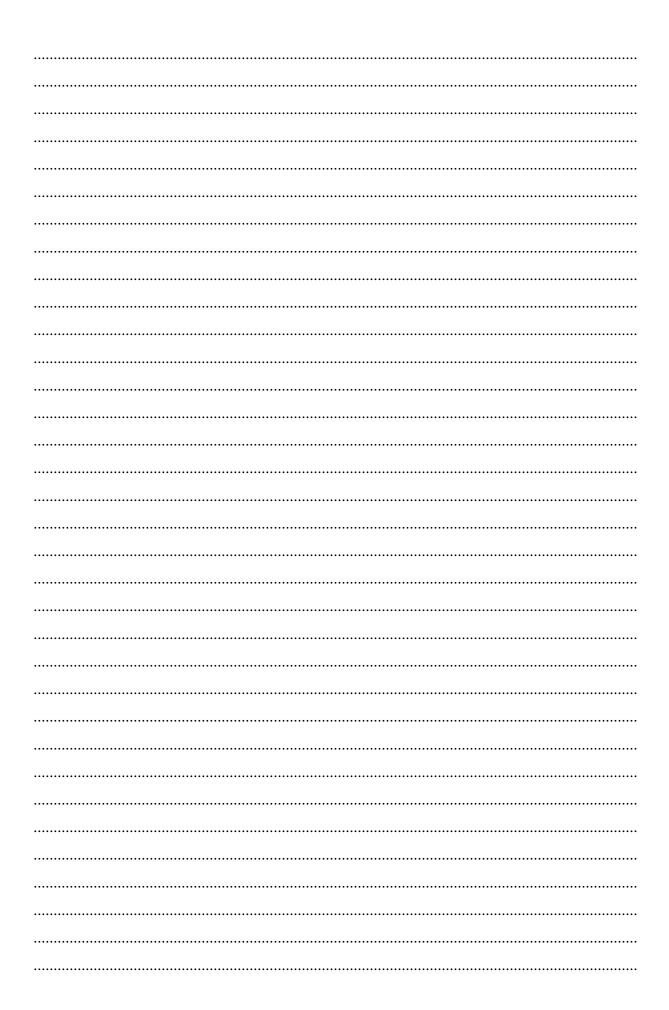
- autrement, on affiche sur une seule ligne un message du genre :

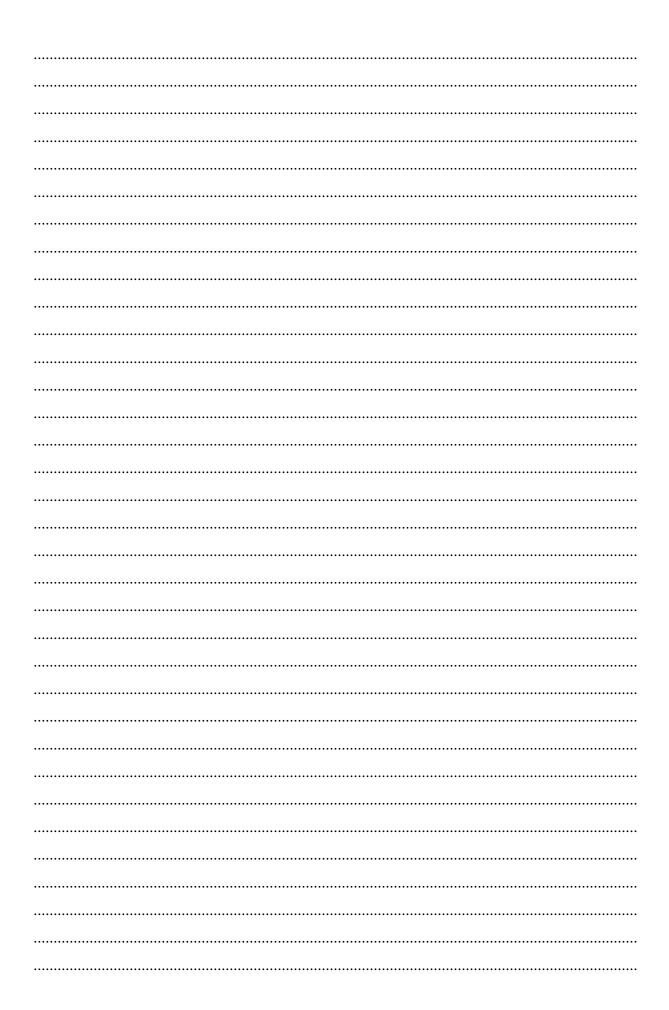
L'équipe numéro YYY formée de : AAA, BBB, CCC!

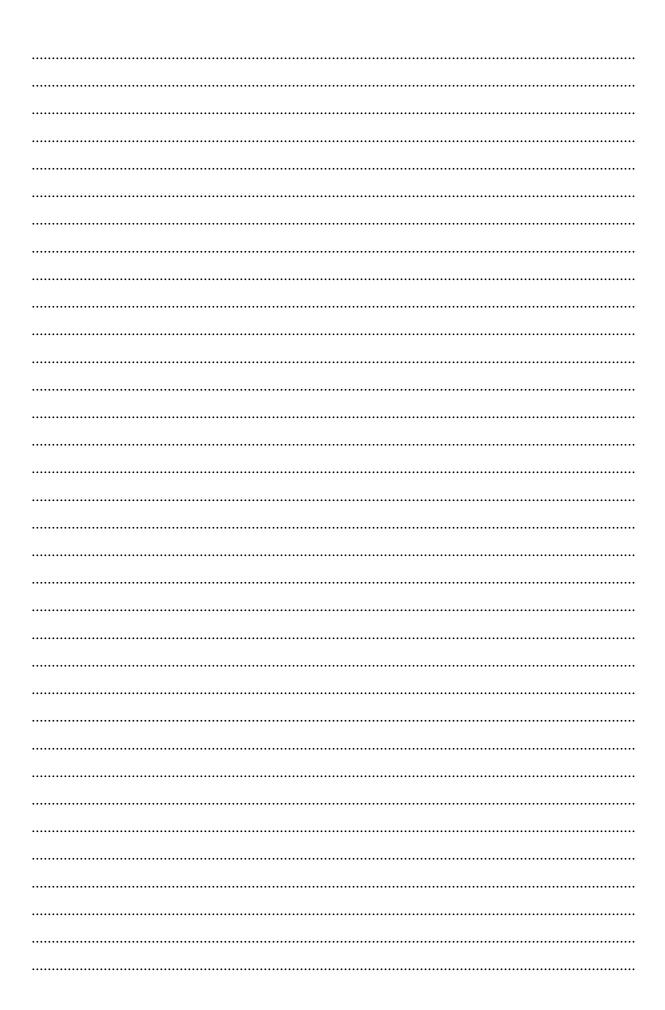
où YYY est la valeur du champ noEquipe de l'objet appelant et AAA, BBB, CCC sont les noms de tous les membres de l'équipe (formée dans notre exemple de trois travailleurs), c'est-à-dire les chaînes de caractères correspondant au champ nom de chaque élément qui n'est pas null dans le tableau dont l'adresse est stockée dans le champ equipe de l'objet appelant ; à l'affichage, ces noms doivent être séparés par des virgules et le dernier nom doit être suivi par un point d'exclamation.

Indication: Si le tableau *equipe* a des éléments de type *Travailleur*, le dernier élément qui contient une adresse qui n'est pas *null* est l'élément d'indice *jauge - 1*.

Indiquez ci-dessous le code source de la classe Chef (à savoir : la déclaration du package						
l'en-tête de la classe, la déclaration des 4 champs propres, la définition du constructeur et la						
redéfinition des méthodes travailler, diriger, faire Equipe et afficher Equipe).						







2. Le but de ce deuxième sujet est d'écrire le code source de la classe *GrandChef* qui permet l'instanciation d'objets correspondant à des chefs "extraordinaires" qui sont des chefs "spéciaux" dont chacun dirige une équipe formée de chefs "ordinaires".

Grâce aux méthodes héritées, un grand chef peut *travailler*, *diriger*, ajouter des membres à son équipe de direction (grâce à la méthode *faireEquipe*) et afficher les membres de son équipe de direction (grâce à la méthode *afficherEquipe*). Cependant, les prérogatives d'un grand chef sont différentes par rapport à un chef ordinaire et ces quatre méthodes doivent être redéfinies dans la classe *GrandChef*.

Plus précisément, la classe *GrandChef* fait partie du package *cms_ctr3*, est <u>publique</u> et <u>dérive</u> directement de la classe *Chef* (qui fait partie du même package).

La classe *GrandChef* n'a <u>aucun champ propre</u> et dans son corps on définit d'abord un constructeur public avec 2 arguments :

- le premier argument *nomGrandChef* est de type chaîne de caractères ;
- le deuxième argument *tailleEquipeDirection* est de type nombre entier.

Ce constructeur:

- appelle le constructeur de la classe mère en lui passant comme arguments ses deux arguments ;
- affiche dans la fenêtre console le message

XXX est un grand chef!

où XXX est la chaîne de caractères indiqué par son premier argument.

De plus, dans le corps de la classe *grandChef*, on <u>redéfinit</u> :

- la méthode d'instance *travailler* qui doit :
 - o afficher dans la fenêtre console le message :

Le grand chef XXX dirige des chefs!

- où *XXX* est la chaîne de caractères correspondant au champ *nom* de l'objet appelant la méthode ;
- o appeler la méthode *travailler* d'origine, c'est-à-dire de la classe mère ;
- la méthode d'instance *diriger* qui doit retourner la chaîne de caractères :

L'équipe de direction numéro YYY bosse!

- où YYY est la valeur stockée dans le champ no Equipe de l'objet appelant la méthode ;
- les méthodes faire Equipe et afficher Equipe selon les indications données ci-dessous.

La méthode d'instance <u>redéfinie</u> faire Equipe doit respecter les consignes suivantes :

si l'argument de la méthode est une instance de la classe *GrandChef*, on affiche dans la fenêtre console le message :

Pas de place pour deux grands chefs!

et on retourne la valeur 99 :

- autrement, si l'argument de la méthode est une instance de la classe *Chef*, on procède ainsi :
 - o si la valeur du champ *jauge* de l'objet appelant est plus grande ou égale à la longueur du tableau désigné par le champ *equipe* de l'objet appelant, on affiche le message :

Equipe de direction numéro YYY complète!

où *YYY* est la valeur du champ *noEquipe* de l'objet appelant, et on retourne ensuite la valeur *101*;

- o autrement, on procède ainsi :
 - on copie la valeur de l'argument de la méthode dans le tableau désigné par le champ *equipe* de l'objet appelant à la position précisée par le champ *jauge* de l'objet appelant;
 - on incrément d'une unité le champ *jauge* de l'objet appelant ;
 - on affiche dans la fenêtre console le message :

ZZZ ajouté à l'équipe de direction numéro YYY!

où **ZZZ** est la chaîne de caractère correspondant au champ *nom* de <u>l'objet argument</u> et **YYY** est la valeur du champ *noEquipe* de l'objet appelant;

- on retourne la valeur 100;
- autrement, (c'est-à-dire si l'argument de la méthode est un "simple" *Travailleur* qui n'est ni *Chef* ni *GrandChef*) on affiche dans la fenêtre console le message :

Le grand chef dirige seulement des chefs!

et on retourne la valeur entière 303.

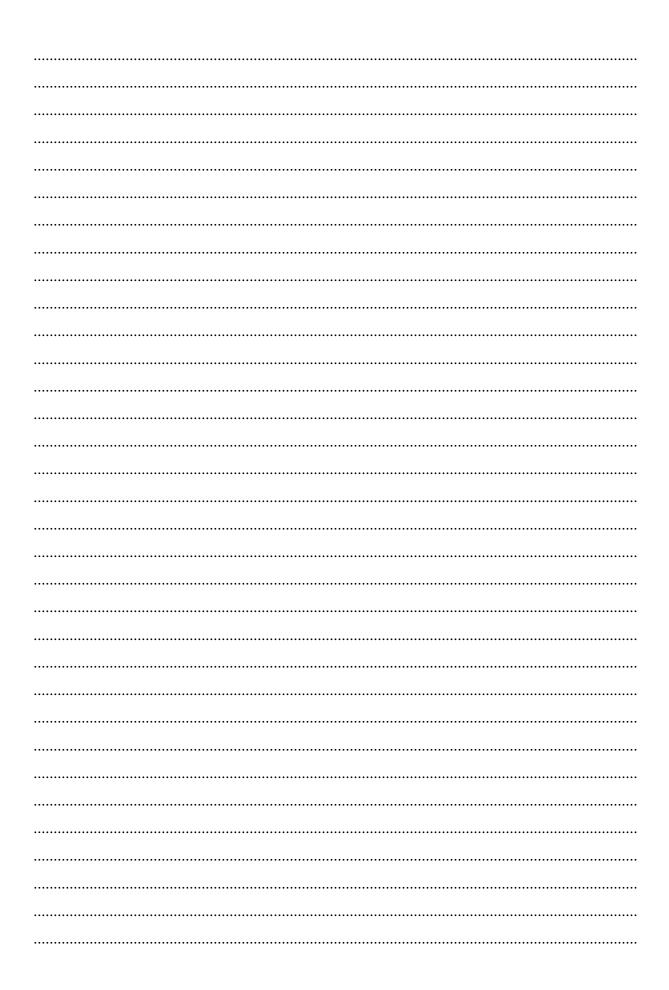
La méthode d'instance <u>redéfinie</u> *afficherEquipe* doit respecter les consignes suivantes :

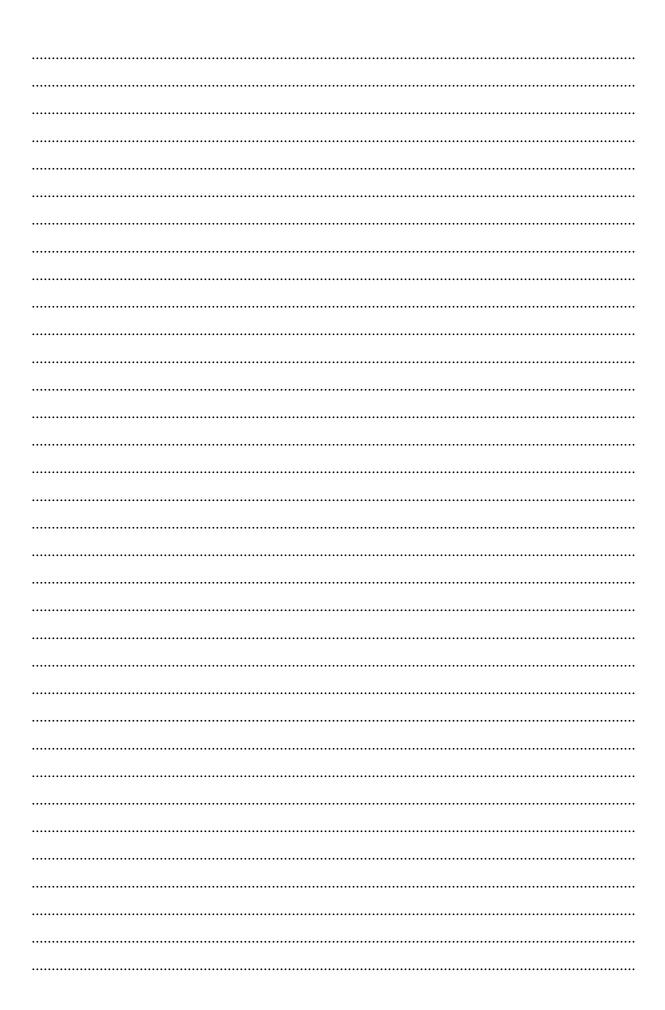
- on affiche, sur une ligne dans la fenêtre console, le message :

Equipe de direction!

- on appelle la méthode *afficherEquipe* d'origine, c'est-à-dire de la classe mère.

Indiquez ci-dessous le code source de la classe grandChef (à savoir : la déclaration du
package, l'en-tête de la classe, la définition du constructeur et la redéfinition des méthodes
travailler, diriger, faireEquipe et afficherEquipe).





3. Dans ce troisième sujet, on vous donne le code source de la classe principale *CP_Ctr3* qui fait partie du package *cms_ctr3* et qui utilise les classes et les interfaces présentées cidessus.

```
package cms ctr3;
public class CP Ctr3
     public static void main(String[] args)
     {
          System.out.println("Première partie");
          Travailleur tabTr[] = { new Travailleur("Dupont"),
                               new Travailleur("Toto"),
                               new Travailleur("Gigi"),
                               new Travailleur("Blanc"),
                               new Travailleur("Bonnet"));
          System.out.println("-----");
          tabTr[3].travailler();
          System.out.println("----");
          Chef chef_1 = new Chef("Patron", 3);
          Chef chef_2 = new Chef("Directeur", -1);
          Chef chef_3 = new Chef("ChefSection", 10);
          System.out.println("-----
          chef 2.travailler();
          System.out.println(chef_2.diriger());
          System.out.println("----");
          GrandChef theBoss = new GrandChef("PDG", 2);
          GrandChef bigBoss = new GrandChef("Président", 5);
          System.out.println("----");
          bigBoss.travailler();
          System.out.println(bigBoss.diriger());
          System.out.println("******************************);
          System.out.println("Deuxième partie");
          System.out.println(chef_1.faireEquipe(chef_2));
          System.out.println(chef_1.faireEquipe(theBoss));
          //Attention à la première valeur de j ci-dessous !
          for(int j=1; j< tabTr.length; j++)</pre>
               System.out.println(chef 1.faireEquipe(tabTr[j]));
          System.out.println("----");
          chef_1.afficherEquipe();
          chef 2.afficherEquipe();
          System.out.println("----");
```

```
System.out.println(theBoss.faireEquipe(bigBoss));
            System.out.println(theBoss.faireEquipe(tabTr[3]));
            System.out.println("-----
            System.out.println(theBoss.faireEquipe(chef_1));
            System.out.println(theBoss.faireEquipe(chef_2));
            System.out.println(theBoss.faireEquipe(chef_3));
            System.out.println("-----
            theBoss.afficherEquipe();
            bigBoss.afficherEquipe();
      }//fin de la méthode main
}//fin de la classe principale
Préciser ci-dessous quels sont les résultats affichés dans la fenêtre console suite à l'exécution
de la classe CP_Ctr3.
```

