À faire : Recevoir une note

Description | Remise | Modifier | Vue de la remise

Fichiers requis: main/Main.java, univ/Personne.java, univ/Prof.java, univ/Etudiant.java (Télécharger)
Nombre maximal de fichiers: 20
Type de travail: Travail individuel

Résumé de l'état du travail

Remises: 0
Dernière soumission: Aucun

La classe Personne

Vous avez à votre disposition la classe `Personne` déjà réalisée lors d'un exercice précédent.

Des classes dérivées

Il existe des catégories de personnes. Par exemple, les étudiants sont une certaine catégorie de personnes. Les profs en sont une autre. Elles ne se présentent pas de la même manière. Elles ne se saluent pas de la même manière.

Les classes `Prof` et `Etudiant` que vous allez écrire vont spécialiser la classe `Personne`.

ATTENTION : TRES IMPORTANT. La classe `Personne` est donnée. Elle est totalement écrite et n'a pas à être modifiée. Les classes `Prof` et `Etudiant` sont partiellement écrites et sont à compléter. **Vous allez constater à l'initiation de l'exercice que celui-ci ne compile pas**. C'est normal. Compte-tenu de la structure des classes, quelque chose manque dans les classes `Prof` et `Etudiant`.

Lisez bien les messages d'erreur de compilation et faites ce qu'il faut pour que ça compile avant de continuer cet exercice.

Le prof

Un prof est une personne. Mais en plus, il enseigne un cours. Il se présente un peu différemment d'une personne normale car il annonce le cours qu'il enseigne. En plus, quand il parle, il emploie toujours des mots désuets qui confèrent à son discours un air suranné. Enfin, quand il donne son nom, c'est en majuscule.

La classe Prof

Un prof est une personne. Je répète.

Attribut

Elle doit encapsuler un attribut `cours`, de type `String`.

Constructeur

Son constructeur doit définir les mêmes choses que celui d'une personne ordinaire, mais en plus, il doit définir son cours.

Getters

Comme l'attribut `cours` est privé, il doit exister un getter pour cet attribut.

Par ailleurs, un prof dit toujours son nom en majuscule. Il faut donc redéfinir le getter correspondant en conséquence.

Présentation

Attention de bien lire ce qui suit.

- La présentation familière (non soutenue) d'un prof est la présentation familière d'une personne ordinaire.
  Par exemple
  `"Yvan Maillot"`

?

- La présentation soutenue d'un prof est la présentation familière d'une personne ordinaire à laquelle il rajoute le cours qu'il enseigne.
  Par exemple
  `"Yvan Maillot. J'enseigne Java"`

Politesse

Un prof ne parle pas comme une personne ordinaire. Il emploie un langage cool et suranné.

Bonjour

Il ne dit pas "Bonjour" mais "Salut" à la place. Ensuite, il se présente dans son langage soutenue.

Par exemple, le bonjour de Yvan Maillot donne
`"Salut, je suis Yvan MAILLOT. J'enseigne Java."`

Réponse à un bonjour

Pour répondre à un bonjour, il dit

- `"Hello "`
- suivi du prénom de la personne à qui il répond
- suivi de `", moi c'est "`
- suivi de sa présentation familière
- suivi de `" et j'enseigne "`
- suivi de son cours

Utilisez les getters plutôt que les attributs.

Ainsi, la réponse de Bruno Adam à Yvan Maillot donne
`"Hello Yvan, moi c'est Bruno ADAM et j'enseigne les bases de données"`

Ca va

Pour dire ça va, le prof, un peu vieillot dit `"Ca get'z, "` suivi du prénom de celui à qui il s'adresse.

Ça peut alors donner quelque chose comme :
`"Ca get'z, Yvan"`

Et toi

Et pour y répondre, il se lâche et sort les jeux de mots de grand-père en lançant très fièrement `"Ca roule, Raoul."` (même si celui à qui il dit ça ne s'appelle pas Raoul).

Veillez à respecter scrupuleusement les messages retournées pour pouvoir valider correctement l'exercice.

L'étudiant

Un étudiant est une personne. Mais il est affublé d'un surnom (par exemple "poussin") et suit une année de formation (par exemple "L3 Miage").

Il ne parle pas comme une personne ordinaire. Il est assez peu loquace et ne se préoccupe guère de son interlocuteur. Seuls ses congénères le comprennent car il emploie des borborygmes comme "Wesh". On le reconnaît facilement car il termine toutes ses phrases par "Gros".

La classe Etudiant

Un étudiant est personne.

Attribut

Elle doit encapsuler les attributs `surnom` et `annee`, de type `String`.

Constructeur

Son constructeur doit définir les mêmes choses que celui d'une personne ordinaire, mais en plus, il doit définir son surnom et son année.

Getters

Comme les attributs sont privés, ils doivent être munis de getters.

Présentation

Attention de bien lire ce qui suit.

Pour l'exemple, nous imaginons l'étudiant "Alan" "Turing", dit "Enigma", étudiant en "L3 Miage"

- La présentation familière d'un étudiant est son surnom.
  Dans le cas de notre exemple d'étudiant :
  `"Enigma"`

?

- La présentation soutenue d'un étudiant est la présentation familière d'une personne ordinaire à laquelle il rajoute l'année qu'il suit
  Par exemple
  `"Alan Turing, étudiant en L3 Miage"`

Politesse

Un étudiant ne parle pas comme une personne ordinaire. Il emploie des borborygmes que seuls ses congénères comprennent et termine tous ses bredouillements par "Gros".

Bonjour

Il ne dit pas "Bonjour" mais `"'Lu Gros"` tout simplement.

Réponse à un bonjour

Pour répondre à un bonjour, il dit `"Ouai Gros"`

Ca va

Pour dire ça va, l'étudiant dit `"Bien ou quoi, Gros ?"`

Et toi

Et pour y répondre, il dit `"Wesh, Gros"`.

Veillez à respecter scrupuleusement les messages retournées (qui ne sont pas très compliqués) pour pouvoir valider correctement l'exercice.

Fichiers requis

main/Main.java

?

```java
/*
 * @author : Yvan Maillot (yvan.maillot@uha.fr)
 */
package main;

import java.io.IOException;
import univ.Personne;

// 0. Regler les import si necessaire.

public class Main {

    // Programme pour tester vos classes. Il suffit d'enlever les commentaires
    // au fur est à mesure de vos réalisations, comme indiqué.
    //
    // 1. Enlever les lignes commentaires /* 1 et 1 */ quand vous aurez
    // écrit le constructeur de la classe Personne. Bien sûr, ça ne suffit
    // pas. Mais vous pourrez exécuter le programme et voir ce qu'il reste
    // à faire. (C'EST DÉJÀ FAIT)
    //
    // 2. Enlever les lignes commentaires /* 2 et 2 */ quand vous aurez
    // écrit la classe Prof.
    //
    // 3. Enlever les lignes commentaires /* 3 et 3 */ quand vous aurez
    // écrit la classe Etudiant.
    //
    // 4. Enlever les lignes commentaires /* 4 et 4 */ quand vous aurez
    // tout écrit.
    //
    // ATTENTION : TRES IMPORTANT. Vous allez constater à l'initiation de l'exercice
    // que celui-ci ne compile pas. C'est normal. Il y a quelque chose qui manque
    // dans les classes Prof et Etudiant. Lisez bien les messages d'erreur de
    // compilation et faites ce qu'il faut pour que ça compile.
    public static void main(String[] args) throws IOException {
        System.out.println("\n--- Quelque part dans le farwest ---- \n");
        Personne lucky = new Personne("Luke", "Lucky", true);
        Personne ma = new Personne("Dalton", "Ma", false);
        lucky.ditBonjourA(ma);
        // Ce simple salut doit entraîner la discussion suivante :
        /*
        << Bonjour, je suis M. Lucky Luke >> dit Luke Lucky.
        << Bonjour M. Lucky Luke.
           Moi c'est Mme Dalton Ma >> répond Ma Dalton.
        << Comment allez-vous ? >> continue Ma Dalton.
        << Ca va bien. Merci >> termine Luke Lucky.
        */
        System.out.println("\n--- Pendant ce temps là, à l'est ---- \n");
        /* 2
        Prof bruno = new Prof("Adam", "Bruno", true, "les bases de données");
        Prof yvan = new Prof("Maillot", "Yvan", true, "Java");
        yvan.ditBonjourA(bruno);
         2 */
        System.out.println("\n--- À quelques pas de là ---- \n");
        /* 3
        Etudiant skell = new Etudiant("Maillot", "Simon", true, "Skell", "M2 MIAGE");
        Etudiant bisou = new Etudiant("Girardot", "Céline", false, "Cel", "M1 MIAGE");
        skell.ditBonjourA(bisou);
        3 */
```

univ/Personne.java

?

```java
/*
 * @author : Yvan Maillot (yvan.maillot@uha.fr)
 */
package univ;

/**
 * NE PAS MODIFIER CETTE CLASSE
 *
 * Une personne a un nom, un prénom et un genre (homme ou femme). Il faudra
 * déclarer des attributs pour ça, nom et prenom de type String et homme, de
 * type boolean (en respectant exactement cette orthographe).
 *
 * Il faudra écrire le constructeur qui initialise ces attributs.
 *
 * Il faudra aussi prévoir des getters pour ces attributs
 *  - getNom()
 *  - getPrenom()
 *  - isHomme()
 *
 * Une personne aime bien se présenter. Elle le fait de deux manières, soit
 * courte (ou familière), sous longue (ou soutenue).Pour ça, il faudra écrire le
 * corps de la méthode
 * protected String presentation(boolean courte).
 *
 * Une personne est polie. Quand elle rencontre une autre personne, elle dit
 * bonjour et se présente. Il s'en suit un petit dialogue, du style :
 *
 * << Bonjour, je suis M. Luke Lucky >> dit Lucky Luke.
 * << Bonjour Lucky Luke.
 *    Moi c'est Mme Dalton Ma >> répond Ma Dalton.
 * << Comment allez-vous ? >> continue Ma Dalton.
 * << Ca va bien. Merci >> termine Lucky Luke.
 *
 * Ce dialogue est établi à l'aide des quatre méthodes, mises à votre disposition,
 * suivantes :
 *       1. public final void ditBonjourA(Personne personne)
 *       2. private void repondAuBonjourDe(Personne personne)
 *       3. private void repondAuBonjourDe(Personne personne)
 *       4. private void demandeSiCaVaA(Personne personne)
 *
 * ELLES N'ONT PAS À ÊTRE MODIFIÉES EN QUOIQUE CE SOIT.
 *
 * Les 4 précédentes méthodes invoquent 4 autres méthodes, également mises à
 * votre disposition, qui retournent des chaînes de caractères.
 *
 *    1. protected String bonjour() qui retourne
 *              "Bonjour, je suis " suivi de sa présentation longue.
 *
 *    2. protected String reponseAuBonjourDe(Personne personne) qui retourne
 *              "Bonjour " + personne.presentation(true) + ".\n   Moi c'est " + presentation();
 *
 *    3. protected String caVa(Personne personne) qui retourne "Comment allez-vous ?";
 *
 *    4. protected String etVous(Personne personne) qui retourne "Ca va bien. Merci"
 *
 * VOUS POUVEZ MODIFIER LEUR CONTENU POUR VOUS AMUSER.
 *
 * L'exécution de la classe autonome main.Main devrait engendrer le petit
 * dialogue entre Lucky et Ma Dalton déjà évoqué. Ce dialogue peut varier en
 * changeant le retour comme précédement expliqué.
 *
 * D'autres dialogues sont pour le moment "en commentaires". Vous pourrez les
 * décommenter et les tester quand vous aurez réalisé l'exercice.
 *
 * NE PAS MODIFIER CETTE CLASSE
 *
 * ATTENTION : TRES IMPORTANT. Vous allez constater à l'initiation de l'exercice
 * que celui-ci ne compile pas. C'est normal. Compte-tenu de la structure des
 * classes, quelque chose manque dans les classes Prof et Etudiant.
 *
 * Lisez bien les messages d'erreur de compilation et faites ce qu'il faut pour
 * que ça compile.
 *
 * @author yvan
 */
public class Personne {

    /**
     * Affiche sur la sortie standard "<< **B** >> dit **P**. \n" où
     *        1. **B** est le résultat de l'invocation de bonjour()
     *        2. **P** est le résultat de l'invocation de presentation(false)
     * Puis invoque personne.repondAuBonjourDe(this) pour continuer le dialogue
     *
     * @param personne est la personne à qui s'adresse le bonjour.
     *
     * NE PAS MODIFIER CETTE MÉTHODE
     */
    public final void ditBonjourA(Personne personne) {
        System.out.printf("<< %s >> dit %s. \n", bonjour(), presentation(false));
        personne.repondAuBonjourDe(this);
    }

    /**
     * Affiche sur la sortie standard "<< **R** >> répond **P**. \n" où
     *        1. **R** est le résultat de l'invocation de reponseAuBonjourDe(personne)
     *        2. **P** est le résultat de l'invocation de presentation(true)
     * Puis invoque demandeSiCaVaA(personne);
     *
     * @param personne est la personne à qui répondre
     *
     * NE PAS MODIFIER CETTE MÉTHODE
     */
    private void repondAuBonjourDe(Personne personne) {
        System.out.printf("<< %s >> répond %s.\n", reponseAuBonjourDe(personne), presentation(false));
        demandeSiCaVaA(personne);
```

?

?

univ/Prof.java

?

```java
/*
 * @author : Yvan Maillot (yvan.maillot@uha.fr)
 */
package univ;

/**
 * Un prof est une personne. Mais, en plus, il enseigne un cours (par exemple
 * le Java).
 *
 * Il faudra donc rajouter, cours, un attribut de type String
 * (en respectant scrupuleusement l'orthographe et la casse).
 *
 * Il faudra créer un constructeur qui initialise un prof comme une personne
 * avec en plus, cours.
 *
 * Il faudra créer le getter pour cours (getCours())
 *
 * Un prof ne parle pas comme une personne ordinaire. Il emploie un langage cool
 * et suranné.
 *
 * Il faudra donc penser à redéfinir toutes les méthodes impliquées dans la
 * conversation. Dans ces méthodes, vous devrez utiliser les getters des attributs
 * plutôt que les attributs eux-mêmes quand vous en aurez besoin.
 *
 * (Toute ressemblance avec une personne connue ou fictive serait fortuite)
 *
 * ATTENTION : TRES IMPORTANT. Vous allez constater à l'initiation de l'exercice
 * que celui-ci ne compile pas. C'est normal. Compte-tenu de la structure des
 * classes, quelque chose manque dans les classes Prof et Etudiant.
 *
 * Lisez bien les messages d'erreur de compilation et faites ce qu'il faut pour
 * que ça compile.
 *
 * @author yvan
 */
public class Prof extends Personne {

    // TODO 1.01 Un prof enseigne un cours (prévoir un attribut à cet effet)
    //

    /**
     * TODO 1.02
     * Le constructeur initialise les attributs de ce prof (qui est une personne)
     * ainsi que son cours.
     *
     * @param nom (String)
     * @param prenom (String)
     * @param homme (true ou false)
     * @param cours de type String (par exemple "Java")
     */
    // TODO 1.02 Constructeur comme une personne normale avec le cours en plus
    //

    /**
     * TODO 1.03
     * Getter pour le cours
     * @return
     */
    public String getCours() {
        return null;
    // TODO 1.03
    }

    // TODO 1.04 Getter : un prof dit toujours son nom en majuscule
    //


    /**
     * TODO 1.05
     *
     * Attention de bien lire ce qui suit.
     *
     * Remarque : une présentation non soutenue et aussi dite familière.
     *
     * La présentation familière d'un prof est la présentation
     * familière d'une personne ordinaire.
     *
     * Par exemple "Yvan Maillot"
     *
     * La présentation soutenue d'un prof est la présentation familière d'une
     * personne ordinaire à laquelle il rajoute ". J'enseigne " + le cours
     * qu'il enseigne.
     *
     * Par exemple
     * "Yvan Maillot. J'enseigne Java."
     *
     * @param soutenue (true ou false)
     * @return la présentation soutenue ou pas de ce prof sous forme d'une String
     */
    @Override
    public String presentation(boolean soutenue) {
        return null;
    // TODO 1.05 Redéfinir presentation() pour un prof
    }

    /**
     * TODO 1.06
     * Un prof ne dit pas "Bonjour" mais "Salut" à la place. Ensuite, il se présente
     * dans son langage soutenue.
     *
     * Par exemple, le bonjour de Yvan Maillot (Java) donne
     * "Salut, je suis Yvan MAILLOT. J'enseigne Java."
     *
     * @return une chaîne de caractères contenant le bonjour de ce prof
     */
```

?

univ/Etudiant.java

?

```java
/*
 * @author : Yvan Maillot (yvan.maillot@uha.fr)
 */
package univ;

/**
 * Un étudiant est une personne. Mais il est affublé d'un surnom (par exemple
 * "poussin") et suit une année de formation (par exemple "L3 MIAGE").
 *
 * Il faudra donc rajouter, surnom et annee, deux attributs de type String
 * (en respectant scrupuleusement l'orthographe et la casse). Comme d'habitude
 * ces attributs doivent être encapsulés.
 *
 * Il faudra créer un constructeur qui initialise un étudiant comme une personne
 * avec en plus, surnom et annee.
 *
 * Il faudra créer les getters pour surnom (getSurnom()) et annee (getAnnee())
 *
 * Un étudiant ne parle pas comme une personne ordinaire. Il est assez peu
 * loquace et ne se préocupe guère de son interlocuteur. Seuls ses congénères le
 * comprennent car il emploie des borborygmes comme "Wesh". On le reconnaît
 * facilement car il termine toutes ses phrases par "Gros".
 *
 * ATTENTION : TRES IMPORTANT. Vous allez constater à l'initiation de l'exercice
 * que celui-ci ne compile pas. C'est normal. Compte-tenu de la structure des
 * classes, quelque chose manque dans les classes Prof et Etudiant.
 *
 * Lisez bien les messages d'erreur de compilation et faites ce qu'il faut pour
 * que ça compile.
 *
 * @author yvan
 */
public class Etudiant extends Personne {
    // TODO 2.00 Les attributs surnom et annee d'un étudiant
    //

    /**
     * TODO 2.01
     * Le constructeur initialise les attributs de cet étudiant (qui est une personne)
     * ainsi que son surnon et son année.
     *
     * @param nom (String)
     * @param prenom (String)
     * @param homme (true ou false)
     * @param surnom (String) Par exemple "poussin"
     * @param annee (String) Par exemple "L3 Miage"
     */
    // TODO 2.01 Le constructeur comme une personne mais avec un surnom et l'année en plus
    //

    // TODO 2.02 Écrire un getter pour le surnom
    //

    // TODO 2.03 Écrire un getter pour l'année
    //

    /**
     * TODO 2.04
     *
     * Remarque : une présentation non soutenue et aussi dite familière.
     *
     * La présentation familière d'un étudiant est son surnom.
     *
     *   - Par exemple pour "Alan" "Turing", dit "Enigma", étudiant en "L3 Miage"
     *     sa présentation familière est "Enigma"
     *
     * La présentation soutenue d'un étudiant est la présentation familière d'une
     * personne ordinaire à laquelle il rajoute ", étudiant en " et son année.
     *
     *   - Par exemple la présentation soutenue de ce même étudiant est
     *     "Alan Turing, étudiant en L3 Miage"
     *
     * Cette méthode ne doit pas pouvoir être redéfinissable.
     *
     * @param soutenue (true ou false)
     * @return la présentation soutenue ou pas de ce prof sous forme d'une String
     */
    // TODO 2.04 Redéfinir presentation(boolean soutenue) - Lire Javadoc 2.04

    /**
     * TODO 2.05
     * Un étudiant ne dit pas "Bonjour" mais "'Lu Gros".
     *
     * La méthode ne doit pas être redéfinissable.
     *
     * @return "'Lu Gros"
     */
     // TODO 2.05 Redéfinition de bonjour() pour un étudiant (la méthode ne doit pas être redéfinissable) - Lire Javadoc 2.05

   /**
     * TODO 2.06
     * Un étudiant répond à un bonjour par un "Ouai Gros"
     *
     * La méthode ne doit pas être redéfinissable.
     *
     * @return "Ouai Gros"
     */
    // TODO 2.06 Redéfinition de reponseAuBonjourDe() pour un étudiant (la méthode ne doit pas être redéfinissable) - Lire Javadoc 2.06

    /**
     * TODO 2.07
     * Un étudiant demande si ça va en disant "Bien ou quoi, Gros ?"
     *
     * La méthode ne doit pas être redéfinissable.
     *
```

?

Webservice

VPL 4.2.4

?