

# TP02 Partie 3: Mini réseau social

## Exercices

### 1. Mini réseau social : la classe DateNaissance

Le but de cet exercice (et des suivants) est de simuler un mini réseau social.

Les aspects éthiques d'un réseau social qui n'associe à chaque personne qu'un seul ami et un seul meilleur ami ne seront pas discutés au cours de ce TP, ces restrictions ayant été choisies pour simplifier l'énoncé du TP.

On considère la classe `DateNaissance` définie par le diagramme UML suivant:

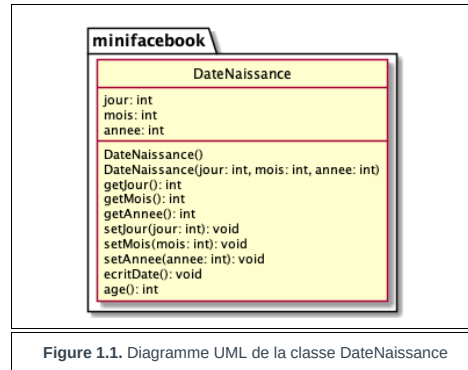


Figure 1.1. Diagramme UML de la classe `DateNaissance`

La classe `DateNaissance` sert à gérer les dates de naissance des membres du réseau.

## Les Attributs

### Question 1.1

Combien la classe `DateNaissance` contient-elle d'attributs?

- ☐ 0
- ☐ 2
- ☒ 3
- ☐ 10

Résultat: Votre réponse est juste.

La classe `DateNaissance` contient 3 attributs: `jour`, `mois` et `annee`.

### Question 1.2

Ajoutez ses attributs à la classe `DateNaissance`.

## Les constructeurs

La classe `DateNaissance` possède 2 constructeurs. Un constructeur *par défaut*, c'est-à-dire qui ne prend aucun paramètre, et un constructeur qui prend 3 paramètres.

Le constructeur de `DateNaissance` qui ne prend pas d'argument fixe la date de naissance arbitrairement au 23 juin 1912.

### Question 1.3

Quel est le type de retour des constructeurs?

- ☐ `int`
- ☐ `DateNaissance`
- ☐ `void`
- ☒ `Aucun`

Résultat: Votre réponse est juste.

Un constructeur n'a **Aucun** type de retour pas même `void` !

**Question 1.4**

Ecrire les 2 constructeurs de la classe `DateNaissance`.

Le but d'un constructeur est d'*initialiser* (c'est-à-dire donner une valeur initiale) aux attributs d'une classe.

Lorsqu'un *paramètre* a le même nom qu'un *attribut*, il peut être utile de faire appel au mot-clé Java pour l'autoréférence: `this`.

**Les Accesseurs / Modificateurs**

Dans cette partie, *Modificateurs* signifie modificateurs de classes et non modificateurs d'accès.

Un accesseur permet aux autre classes de connaître la valeur des attributs d'une classe. Ici, les accesseurs de la classe `DateNaissance` sont au nombre de 3: `getJour(): int`, `getMois(): int` et `getAnnee(): int`. Ils renvoient la valeur de l'attribut de la classe correspondant.

**Question 1.5**

Ecrire dans la classe `DateNaissance` les 3 accesseurs.

Un modificateur permet à une autre classe de modifier la valeur des attributs. Ici, les modificateurs de la classe `DateNaissance` sont au nombre de 3: `setJour(jour: int): void`, `setMois(mois: int): void` et `setAnnee(annee: int): void` et modifient les valeurs des attributs correspondant grâce à la valeur passée en paramètre.

**Question 1.6**

Ecrire dans la classe `DateNaissance` les 3 modificateurs.

**Ecrire la date**

On souhaite à présent pouvoir écrire de manière normalisée une date à l'écran.

L'écriture de la date aura la forme: `jour/mois/annee` par exemple `25/12/2015`.

Pour écrire à l'écran en Java, on utilise la méthode `System.out.print(s: String)`. Vous pouvez donc créer à l'intérieur de la méthode `ecritDate(): void` une *variable* `s` de type `String`. `s` prendra tout d'abord la valeur du `jour`, puis vous pourrez lui ajouter d'autres chaînes de caractères (comme `"/"`) grâce à l'opérateur de concaténation des chaînes de caractère: `"+"`.

**Question 1.7**

Le type de la méthode `ecritDate` est

- ☒ `void`  
☐ `String`  
☐ `Aucun type`

Résultat: Votre réponse est juste.

Le but de la méthode `ecritDate` est d'afficher quelque chose à l'écran et non de renvoyer une chaîne de caractères. Comme l'indique sa signature dans le diagramme UML, le type de la méthode `ecritDate` est `void`.

**Question 1.8**

Ecrire la méthode `ecritDate` de la classe `DateNaissance`.

**Question 1.9**

Quelle est l'erreur du compilateur Java si vous écrivez la ligne suivante: `String s = jour; ?`

- ☐ impossible de trouver le symbole `jour`  
☐ types incompatibles: `String` ne peut pas être converti en `int`  
☒ types incompatibles: `int` ne peut pas être converti en `String`  
☐ Il n'y a pas d'erreur

Résultat: Votre réponse est juste.

Java ne peut pas convertir l'entier `jour` (qui devrait être un symbole connu dans la méthode `ecritDate` puisque vous l'avez déclaré comme attribut de la classe `DateNaissance`) en chaîne de caractères pour l'affecter à `s` qui est une chaîne de caractère.

Pour résoudre ce problème, plusieurs solutions:

- convertir l'entier `jour` en chaîne de caractère grâce à `Integer.toString(jour)`
- utiliser la conversion forcée par la concaténation d'une chaîne de caractère vide et d'un entier: `" " + jour`

## Calcul de l'âge

La méthode `age` ne prend pas de paramètres et renvoie l'âge donnée par la date de naissance. On ne considère pour l'instant que le nombre d'années (et l'on ne cherchera pas à savoir si l'anniversaire est déjà passé ou pas). **L'année de référence sera 2015** (date à laquelle a été écrit cet énoncé).

### Question 1.10

Ecrire la méthode `age()` : `int`.

Le type de retour de la méthode `age` est `int`. Il peut être utile de créer, au début de la méthode `age` une *variable* `resultat` de type `int`. La dernière ligne de la méthode `age` sera alors `return resultat;`. Entre la déclaration de la *variable* `resultat` et la dernière ligne de la méthode `age`, il y aura évidemment le calcul de l'âge...

## Test de la classe DateNaissance

### Question 1.11

Exécutez le main de la classe `TestDateNaissance`.

### Question 1.12

A quoi servent les lignes `System.out.println();` ?

- ☐ A écrire la date
- ☒ A revenir à la ligne
- ☐ A rien

Résultat: Votre réponse est juste.

### Question 1.13

Quel est l'âge de Jack ?

- ☐ 41 ans
- ☒ 103 ans
- ☐ 70 ans
- ☐ 2012 ans

Résultat: Votre réponse est juste.

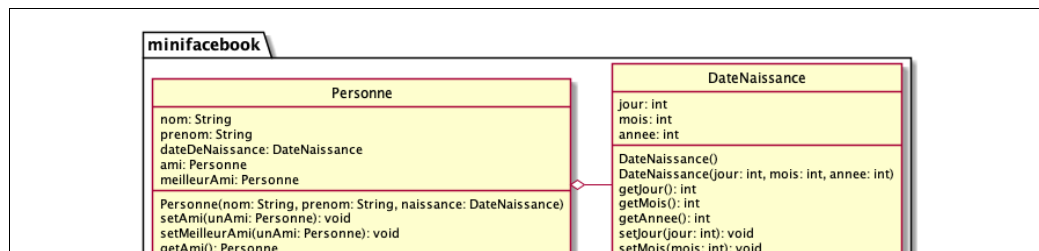
La date de naissance de Jack est instanciée avec le constructeur qui ne prend pas de paramètre. Ce constructeur, créé plus haut, initialise arbitrairement la date de naissance à 23 juin 1912.

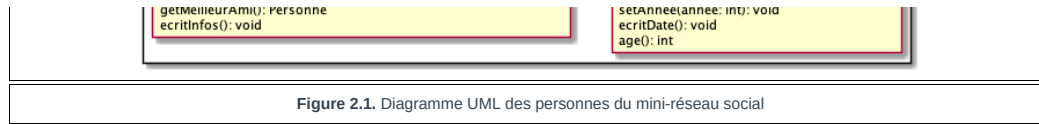
Au fait, de quel personnage célèbre de l'informatique né à cette date ?

[Fermer]

## 2. Mini réseau social : la classe Personne

On considère à présent les classes `Personne` et `DateNaissance` définies par le diagramme UML suivant:





La classe `Personne` stocke les données d'une personne: son nom, son prénom, sa date de naissance ainsi que 2 de ses amis. Elle ne contient qu'un seul constructeur qui prend en paramètre le nom, le prénom et la date de naissance de la personne.

#### Question 2.1

Créer la classe `Personne` avec ses attributs, son constructeur, ses accesseurs et ses modificateurs.

La méthode `ecritInfos()` écrit les informations sur la personne sous la forme suivante (du moins pour le moment):

```

1  -----
2  prenom nom
3  Né(e) le : jour/mois/annee (age ans)
4  -----
  
```

Elle utilise les méthodes suivantes:

- `System.out.print();` et `System.out.println();`
- `System.out.println("-----");`
- La méthode `ecritDate` de la classe `DateNaissance`, sous la forme `dateDeNaissance.ecritDate()`

Un exemple de résultat est:

```

1  -----
2  Patty Bullaire
3  Né(e) le : 10/5/1968 (41 ans)
4  -----
  
```

#### Question 2.2

Ecrire la méthode `ecritInfos()` de la classe `Personne`.

#### Question 2.3

Exécutez la méthode `main` de la classe `TestPersonne` et vérifiez que vous avez bien le résultat attendu.

[Fermer]

### 3. Mini réseau social: ce qui se passe à l'exécution

#### Question 3.1

Dessiner ce qui se passe dans la mémoire au cours de l'exécution de la méthode `main` de l'exercice précédent.

On souhaite à présent ajouter dans la méthode `main`, à la suite de la création de l'instance `jack` et **avant la ligne `patty.ecritInfos();`** du code existant, le code ci-dessous:

```

1  patty.setAmi(jack);
2  patty.setMeilleurAmi(jack);
3
4  jack.setAmi(patty);
5  jack.setMeilleurAmi(patty);
  
```

#### Question 3.2

Dessiner ce qui se passe dans la mémoire pour chacune de ces lignes (diagramme APO).

#### Question 3.3

Des nouvelles instances d'amis et de meilleurs amis ont été créées.

- ☐ *vrai*  
☒ *faux*

Résultat: Votre réponse est juste.

Si vous n'avez pas la bonne réponse, avez-vous réellement dessiné le diagramme APO ? Si oui, demandez de l'aide à votre enseignant, si non, c'est peut être le moment de s'y mettre...

On souhaite à présent pouvoir afficher le nom et le prénom de l'ami et du meilleur ami de chaque personne dans la méthode `ecritInfos` de la classe `Personne`. Le résultat devrait être le suivant:

```

1  -----
2  Patty Bullaire
3  Né(e) le : 10/5/1968 (41 ans)
4  Meilleur ami : Jacques Pottes
5  Ami : Jacques Pottes
6  -----
7
8  Jacques Pottes
9  Né(e) le : 1/1/1970 (39 ans)
10 Meilleur ami : Patty Bullaire
11 Ami : Patty Bullaire
12 -----

```

#### Question 3.4

Compléter la méthode `ecritInfos` de la classe `Personne` pour obtenir le résultat de l'exemple ci-dessus et vérifier en exécutant la classe `TestPersonne` avec le code décommenté.

[Fermer]

#### 4. Mini réseau social : décodage et reverse engineering (rétro-ingénierie)

On souhaite obtenir le résultat suivant dans la mémoire:

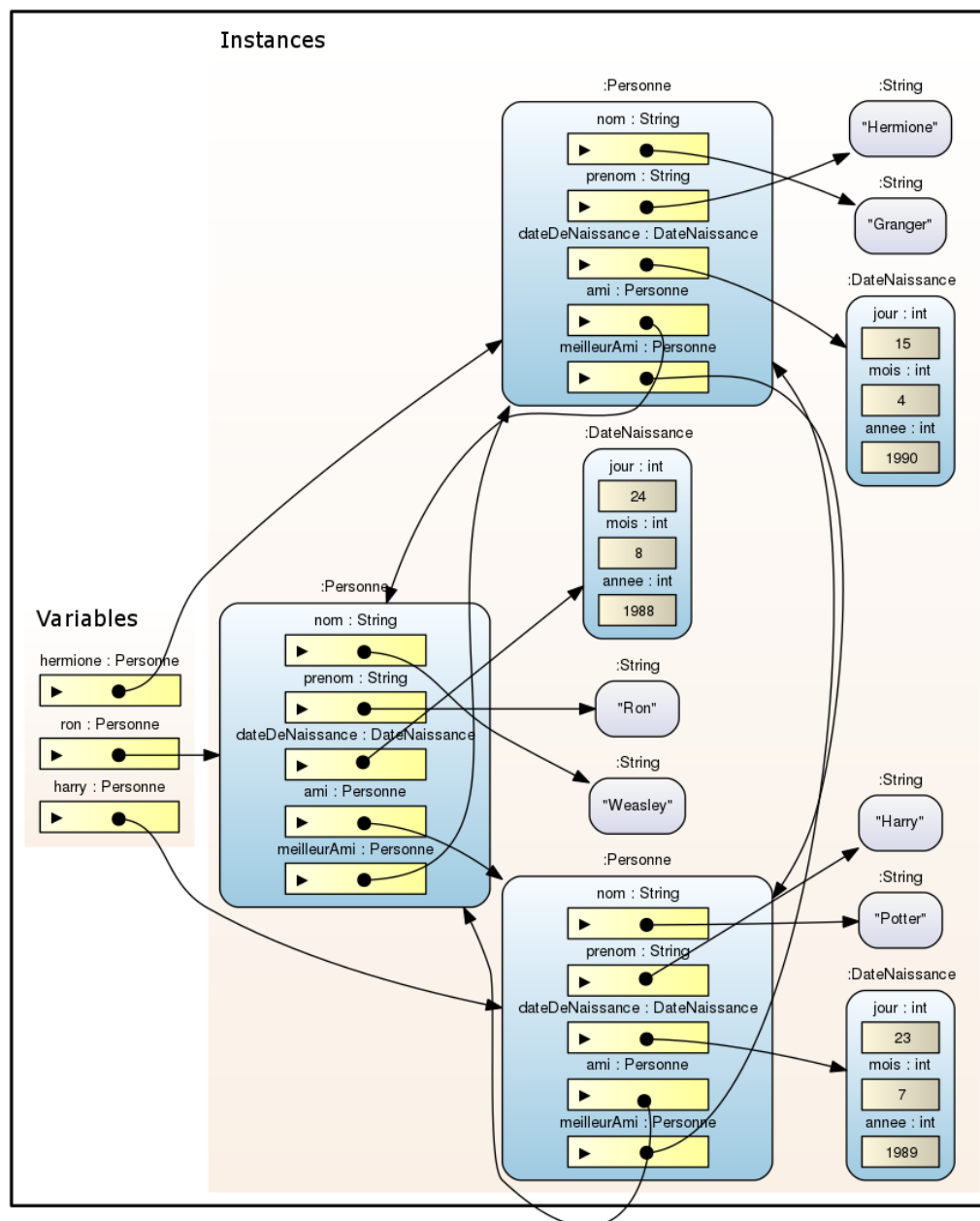


Figure 4.1. Résultat de l'exécution dans la mémoire.

**Question 4.1**

Ecrire dans la méthode `main` de la classe `HarryPotterSocial` ce qu'il faut pour obtenir cet état de la mémoire.

**Question 4.2**

Combien de fois le mot clé `new` apparaît-il dans votre méthode `main` ?

- ☐ 3
- ☒ 6
- ☐ 9
- ☐ 12

Résultat: Votre réponse est juste.

Le mot clé `new` permet la création d'une nouvelle instance. Votre code doit comporter:

- 3 instances de `Personne` (Harry, Ron et Hermione)

- 3 instances de `DateNaissance` (celles de Harry, Ron et Hermione)

Vous avez donc dû invoquer le mot-clé `new` 6 fois.

**Question 4.3**

Appelez la méthode `ecritInfos()` sur chacune des instances de `Personne` que vous avez créée. Vérifier si l'affichage sur la sortie standard correspond bien à ce que vous avez prévu.

On s'intéresse à présent au morceau de code suivant :

```
1 | Personne hugo = new Personne("Histe", "Hugo", new DateNaissance());
2 | hugo.setMeilleurAmi(hugo);
3 | hugo.ecritInfos();
```

**Question 4.4**

Sur une nouvelle feuille, décrire ce qui va se passer dans la mémoire. Que va-t-on avoir comme résultat de ce code?

**Question 4.5**

Exécuter la méthode `main` de la classe `TestHugo`.  
Le résultat est-il celui que vous avez prévu ?

**Question 4.6**

Ce qui pose problème dans le code précédent, c'est le fait qu'Hugo Histe soit ami avec lui-même.

- ☐ vrai
- ☒ faux

Résultat: Votre réponse est juste.

Si vous avez fait le diagramme APO, vous avez vu que l'autoréférence (`this`) pour la référence de l'attribut `ami` ne pose pas de problème (à condition que vous ayez correctement écrit la méthode `ecritInfo` qui ne devrait que récupérer les attributs `nom` et `prénom` de l'instance référencée par `ami` et `meilleurAmi`, et non appeler la méthode `ecritInfo()` des instances en questions).

Par contre, la méthode `ecritInfo` récupère les `nom` et `prénom` de l'instance référencée par `meilleurAmi`. Or ici, il n'y a eu aucune instanciation de `meilleurAmi`, donc cette référence est `NULL`. On ne peut donc pas récupérer les attributs `nom` et `prénom` de cette instance puisqu'elle n'existe pas ! C'est ce qui produit une exception en Java.

[Fermer]

## 5. Bénéfices de l'encapsulation

Le principe d'encapsulation sera vu dans quelques semaines.

Pour cet exercice, vous pouvez retenir que ce principe expose le fait que la classe doit garantir l'intégrité de chacune des instances

Cela signifie que l'on ne doit pas être capable de créer une instance de `DateNaissance` dont les valeurs de `jour`, `mois` et `année` ne sont pas cohérent (par exemple négatifs, etc.).

On notera également que cet exercice nécessite le chapitre du cours sur les conditionnelles (à voir pour aujourd'hui).

Afin de garantir la cohérence de notre système d'information *Mini réseau social*, nous allons utiliser le principe de l'encapsulation pour garantir la validité des dates de naissances utilisées.

**Question 5.1**

Modifier la méthode `setAnnee` afin qu'elle ne modifie pas l'année si le paramètre donné n'est pas positif ou s'il est supérieur à l'année courante (ici 2015).

**Question 5.2**

Modifier la méthode `setMois` afin qu'elle ne modifie pas le mois si le paramètre donné n'est pas un numéro de mois valide.

Attention ! Lors de l'appel de `setMois` dans le constructeur qui prend 3 paramètres (vous n'alliez tout de même pas oublier de modifier le code de votre constructeur, sans réécrire le même code de ces méthodes ?!?!...), l'attribut `mois` aura été initialisé à 0 (qui n'est pas un numéro de mois valide) par l'opérateur `new`. Si l'utilisateur passe en paramètre un mois non valide, il faudra tout de même initialiser le mois à un numéro valide.

Plusieurs solutions existent pour résoudre le problème. Ici, il est demandé, lorsque le mois passé en paramètre n'est pas valide, de forcer l'attribut `mois` à la valeur 1.

**Question 5.3**

Idem pour la méthode `setJour` qui devra vérifier en fonction du mois, si le numéro du jour est valide.

*Note* : Une année est bissextile si elle est divisible par 4 mais pas divisible par 100, ou bien si elle est divisible par 400.

Dans votre constructeur (et vos tests en général), pensez à appeler `setJour` **après** avoir appelé `setMois` et `setAnnee`...

De même que pour le mois, si le numéro du jour passé en paramètre n'est pas valide, le jour sera mis à 1.

[Fermer]