Série A : Exercices de récapitulation

Table des matières

Exercice A.1 : Récapitulation des notions vues en 3e

Vous allez réaliser sous Unimozer la classe **`Fraction`** qui permet de représenter une fraction définie par un numérateur et un dénominateur.

Exemples: $\frac{1}{4}$, $\frac{2}{6}$, $\frac{-5}{7}$, $\frac{-3}{-9}$, $\frac{3}{1}$, $\frac{0}{6}$, $\frac{7}{0}$

Appeler le projet **`Fraction.`** Veillez à indenter votre code correctement et à documenter la classe par l'utilisation de Javadoc.

Répondez sur papier aux questions marquées du symbole  .

Classes et attributs

1)  Quelle est la différence entre la notion de classe et la notion d'objet (ou instance)? Donnez des exemples.

2) Définissez sous Unimozer la classe **`Fraction`** et ajoutez les attributs nécessaires.
Quelle visibilité définissez-vous pour ces attributs?
Quels types de données attribuez-vous aux attributs de la classe **`Fraction`**?

3)  Expliquez les visibilités **`public`** et **`private`**.

4)  Remplissez le tableau avec les types de données que vous connaissez.

Type	Explication

Méthodes

Accesseurs (« Getters ») et manipulateurs (« Setters »)

5)  Pourquoi utilise-t-on des accesseurs et des manipulateurs ?

6) Réalisez des accesseurs utiles pour la classe **`Fraction`**.

7) Réalisez un manipulateur utile pour la classe **`Fraction`**.

8)  Créez une instance de la classe **`Fraction`** représentant le quotient $\frac{1}{4}$.
Comment procédez-vous ? Qu'est-ce qui permettrait de simplifier cette tâche ?

Constructeurs

9)  A quoi sert un constructeur ? Est-il nécessaire de définir un constructeur pour une classe donnée ? Une classe peut-elle avoir plusieurs constructeurs ?

10) Réalisez un constructeur pour la classe **`Fraction`**.

Conventions de noms

11)  Quelles conventions de noms sont utilisées en Java pour: les classes, les attributs, les méthodes, les paramètres et les variables locales ?

La structure alternative

12)  Donnez un exemple d'une expression booléenne (condition).

13)  Indiquez la condition qui permet de vérifier que la variable **`x`** contient un nombre compris dans l'intervalle [1, 60].

14)  Indiquez la valeur de la variable **`x`** après l'exécution des lignes de code suivantes :

```
int x=-5;
   if (x < 0)
         x = -x;
   else;
         x = 2 * x;
```

15) Réalisez la méthode **`toString`** qui retourne la représentation d'une fraction sous forme de chaînes de caractères.
Exemples :
Pour la fraction $\frac{4}{9}$ la méthode retourne **`"4 / 9"`**
Pour la fraction $\frac{6}{1}$ la méthode retourne **`"6"`**

16) Réalisez la méthode **`getDecimal`** qui retourne la représentation décimale de la fraction.

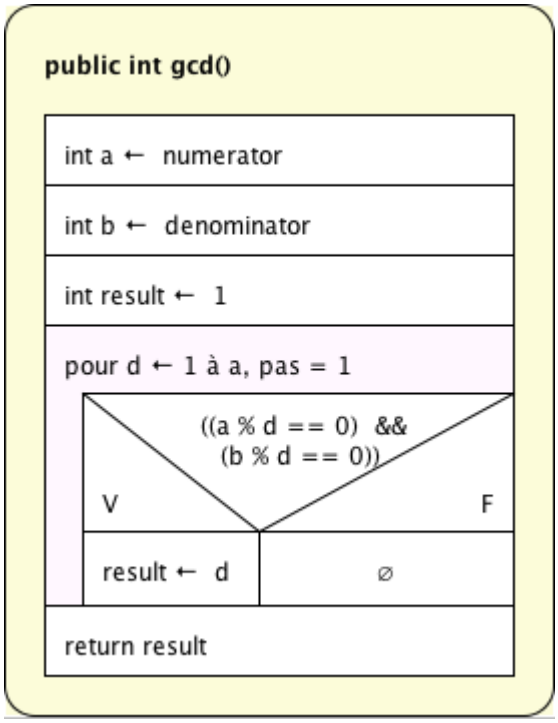
17) Réalisez la méthode **`isNegative`** qui indique si la fraction représente un nombre négatif ou non. Veillez à définir le type de données adéquat pour le résultat de la méthode !

La structure répétitive (boucles)

18)  Indiquez le code qui permet d'afficher à la console les nombres de 1 à 10 (par ordre croissant). Utilisez une boucle **`for`** !

19)  Indiquez le code qui permet d'afficher à la console les nombres de 10 à 1 (par ordre décroissant). Utilisez une boucle **`for`** !

20) Réalisez la méthode **`gcd`** (pour ***g**reatest **c**ommon **d**ivisor*) qui calcule le **p**lus **g**rand **c**ommun **d**iviseur (ou PGCD) du numérateur et du dénominateur d'une fraction en traduisant le structogramme suivant :



21) Dessinez un tableau des variables (tableau d'exécution) pour l'appel à **`gcd`** pour chacune des fractions suivantes :

i. 4 / 6

ii. 6 / 4

iii. -4 / 6

22) Modifiez la méthode **`gcd`** pour qu'elle fonctionne avec des nombres négatifs.

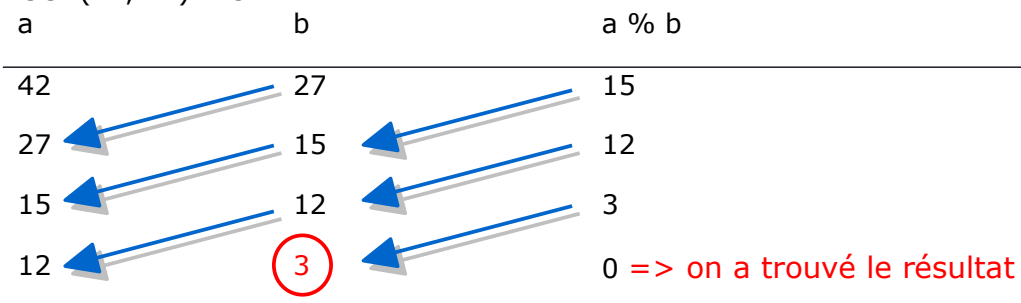
23) Indiquez quand on utilise une boucle **`for`** et quand on utilise une boucle **`while`**.

24) Donnez un exemple d'une boucle vide (aucune itération) et d'une boucle infinie (infinité d'itérations).

25) En analysant les deux premiers cas d'exécution de la méthode **gcd** du point 21, réalisez une méthode **gcd** plus performante.

26) Réalisez une méthode **gcd** encore plus performante en utilisant la méthode de la division euclidienne.

Exemple :
PGCD(42, 27) = 3



27) Réalisez la méthode **lcm** (pour ***l**east **c**ommon **m**ultiple*) qui permet de calculer le **p**lus **p**etit **c**ommun **m**ultiple (ou PPCM) du numérateur et du dénominateur d'une fraction sachant que:

$$PPCM(a,b)=\frac{|a\cdot b|}{PGCD(a,b)}$$

28) Réalisez la méthode **reduce** qui permet de simplifier la fraction. Veillez à ce que la méthode fonctionne dans tous les cas !

29) Réalisez la méthode **reciprocal** qui inverse la fraction.

Par exemple : la fraction $\frac{5}{11}$ est transformée en $\frac{11}{5}$

Utilisation d'une classe comme paramètre

30) Réalisez la méthode **add** qui permet d'additionner à la fraction une fraction passée comme paramètre. Réalisez également les méthodes **subtract**, **multiply** et **divide**.

Pour avancés

31) Vous voulez pouvoir créer des fractions à partir de nombres décimaux.

Exemple: 0,25 => $\frac{1}{4}$

Comment pouvez-vous procéder ? Effectuez les réalisations nécessaires.
Indications:

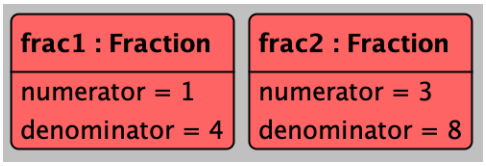
- Procédez par étapes ! Exemple: 0,25 => $\frac{25}{100}$ => $\frac{1}{4}$
- Pour obtenir la partie entière d'un nombre décimal de type **double** vous pouvez utiliser la conversion suivante : **(int) decimal**
- Testez votre constructeur avec beaucoup de valeurs différentes. Est-ce qu'il fonctionne toujours correctement? Si non essayez de déterminer une explication ou même une solution.

Exercice A.2 : Addition de deux fractions

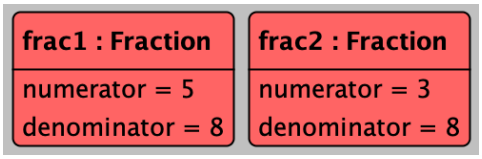
Copier la solution de l'exercice A.1 et renommer le projet (le dossier) en **A02_Addition_Fractions_new**.

Dans cet exercice vous allez créer des objets de la classe **`Fraction`** à l'aide de l'instruction **new** et effectuer l'addition suivante : $\frac{1}{4}+\frac{3}{8}$

1. Créez manuellement dans Unimozer des objets de la classe **`Fraction`** représentant les deux fractions.

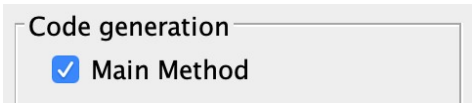
   

   Utilisez ensuite la méthode **`add`** de la fraction **`frac1`** pour additionner la fraction **`frac2`**.

2. Maintenant vous allez programmer les actions que vous avez effectuées sous le point 1.

   Ajoutez tout d'abord la classe **`Test`** au projet. Lors de la création de la classe sélectionnez dans le dialogue l'option de génération d'une méthode **`main`**.

   C'est dans cette méthode que vous allez effectuer vos développements.

   ```
   public static void main(String[] args)
   {

   }
   ```

   Ajoutez dans cette méthode le code effectuant les mêmes actions que sous le point 1. en déclarant et en utilisant les variables **`frac1`** et **`frac2`**.

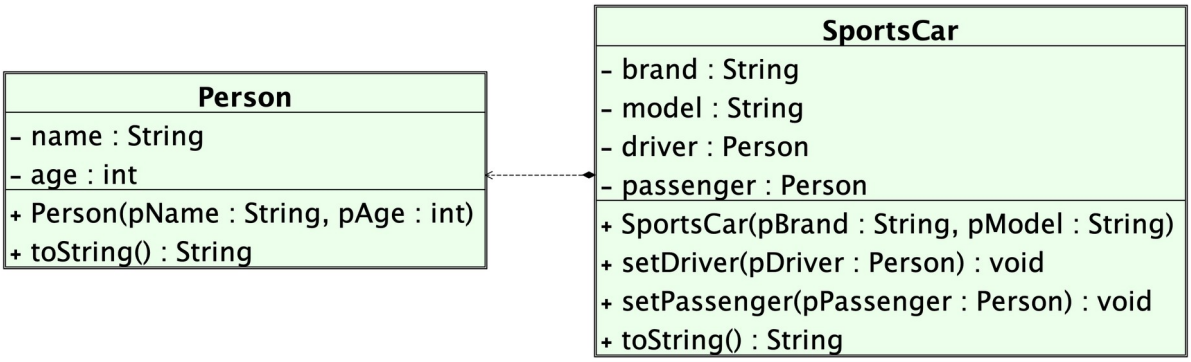
   Vous pouvez utiliser la méthode **`System.out.println()`** pour afficher le contenu des variables dans la console ou la méthode **`Unimozer.monitor()`** pour afficher les objets des variables dans le panneau des objets (indiquer la variable et son nom comme arguments p.ex. **`Unimozer.monitor(frac1, "frac1")`**).

3. Représenter l'exécution de la méthode à l'aide d'un tableau des variables en y dessinant les objets.

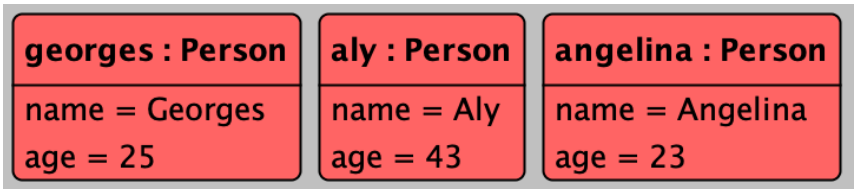
Notions requises : Création de nouveaux objets avec l'instruction **new**. Tableaux des variables et diagrammes objet.

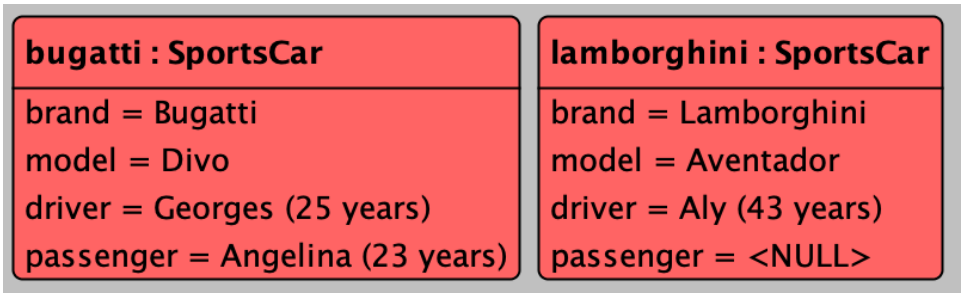
Exercice A.3 : Voitures de sport

Ouvrez le projet **`A03_Sportscars`**. Ce projet contient les deux classes suivantes :



1. Créez une classe test et développez la méthode pour créer les objets suivants :





2. Représenter l'exécution de la méthode à l'aide d'un tableau des variables en y dessinant les objets.

Notions requises : Création de nouveaux objets avec l'instruction **new**. Tableaux des variables et diagrammes objet.

Exercice A.4 : Exercice de compréhension

Voici le code d'une méthode créant des instances (objets) de la classe **`Fraction`**.

```
public static void main(String[] args)
{
     Fraction f1;

     f1 = new Fraction(0.75);

     System.out.println("1: f1="+f1.toString());

     Fraction f2 = new Fraction(4, 12);

     System.out.println("2: f2="+f2.toString());

     f1 = f2;

     System.out.println("3: f1="+f1.toString()+" ,f2="+f2.toString());

     f1.reduce();

     System.out.println("4: f1="+f1.toString()+" ,f2="+f2.toString());

     f2 = null;

     System.out.println("5: f2="+f2.toString());

}
```

Complétez le tableau des variables suivant en dessinant les objets de la classe **`Fraction`**.

Notions requises : Création de nouveaux objets avec l'instruction **new**. Références vers les objets. Tableaux des variables et diagrammes objet.