Programmation Orientée Objet

Valeur d'accueil et de reconversion en informatique (VARI1)

Daniel Porumbel (dp.cnam@gmail.com)

http://cedric.cnam.fr/~porumbed/vari1/

```
void setup(){
    float a = 1.0/60;
    float b = 1.0/30;
    float c = 1.0/20;
    if(10*(a+b+c)==1){
        println("10(a+b+c) vaut 1");
    }
    else{
        println("10(a+b+c) différent de 1");
    }
}
```

1. Que devrait afficher ce programme ?
2. Qu'affiche-t-il réellement ?

$\Longrightarrow$ On a besoin d'une nouvelle structure de données !

```
void setup(){
    float a = 1.0/60;
    float b = 1.0/30;
    float c = 1.0/20;
    if(10*(a+b+c)==1){
        println("10(a+b+c) vaut 1");
    }
    else{
        println("10(a+b+c) différent de 1");
    }
}
```

1. Que devrait afficher ce programme ?
2. Qu'affiche-t-il réellement ?

$\Longrightarrow$ On a besoin d'une nouvelle structure de données !

On définit la structure/classe de `Fraction`

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    Frac(int x, int y){         //fraction x/y
        num = x ;
        den = y ;
    }
    float valFloat(){           //renvoie une valeur
        return (float)num/den;//de type float
    }
}
void setup(){
   Frac a=new Frac(2,3); //initialise a=2/3
   println(a.valFloat());//affiche 0.6666667
}
```

On définit la structure/classe de `Fraction`

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    Frac(int x, int y){         //fraction x/y
        num = x ;
        den = y ;
    }
    float valFloat(){           //renvoie une valeur
        return (float)num/den;//de type float
    }
}
void setup(){
   Frac a=new Frac(2,3); //initialise a=2/3
   println(a.valFloat());//affiche 0.6666667
}
```

constructeur

Fonction qui peut s'appliquer sur des objets de type Frac

On définit la structure/classe de `Fraction`

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    Frac(int x, int y){         //fraction x/y
        num = x ;
        den = y ;
    }
    float valFloat(){           //renvoie une valeur
        return (float)num/den;//de type float
    }
}
void setup(){
   Frac a=new Frac(2,3); //initialise a=2/3
   println(a.valFloat());//affiche 0.6666667
}
```

attributs visibles dans `valFloat()`, car `valFloat()` s'applique sur des objets `Frac`

La possibilité d'inverser une `Frac`(tion)

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    Frac(int x, int y){         //fraction x/y
        num = x ;
        den = y ;
    }
    float valFloat(){           //renvoie une valeur
        return (float)num/den;//de type float
    }
    void inverser(){
        int tmp = num;
        num     = den;
        den     = tmp;
    }
}
```

Qu'affiche ce code ?

```
void setup(){
    Frac a=new Frac(2,3); //initialise a=2/3
    a.inverser();
    println(a.valFloat());
    Frac b = new Frac(4,5);
    Frac c = new Frac(1,2);
    b.inverser();
    c.inverser();
    c.inverser();
    println(b.valFloat());
    println(c.valFloat());
}
```

Qu'affiche ce code ?

```
void setup(){
    Frac a=new Frac(2,3);   //initialise a=2/3
    a.inverser();           //a=3/2
    println(a.valFloat());  //1.5
    Frac b = new Frac(4,5); //b=4/5
    Frac c = new Frac(1,2); //c=1/2
    b.inverser();           //b=5/4
    c.inverser();           //c=2/1
    c.inverser();           //c=1/2
    println(b.valFloat());  //1.25
    println(c.valFloat());  //0.5
}
```

Que affiche ce code ?

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...   ...
    ...   ...   ...
    boolean egal(int i){
        if(num==den*i)
            return true;
        return false;
    }
}
void setup(){
    Frac a=new Frac(6,3); //initialise a=6/3
    if(a.egal(2))
        println("6/3=2");
}
```

Quel est le role de `simplifier()` ?

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...
    void simplifier(){
        for(int div=2;div<num;div++)
            //si div est diviseur commun
            if((num%div==0)&&(den%div==0)){
                num = num/div;
                den = den/div;
            }
    }
}
```

Rappel : l'opérateur `%` renvoie le reste de la division

`toString()` convertit l'objet en chaîne de caractères

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...
    ...   ...
    String toString(){
        String s = num + "/"+den;
        return s;
    }
}
void setup(){
    Frac a=new Frac(6,4); //initialise a=6/4
    println(a.toString());
    println(a);//.toString() est appelé
               //automatiquement=>a.toString()
}
```

`toString()` convertit l'objet en chaîne de caractères

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...
    ...   ...
    String toString(){
        String s = num + "/"+den;
        return s;
    }
}
void setup(){
    Frac a=new Frac(6,4); //initialise a=6/4
    println(a.toString());
    println(a);//.toString() est appelé
               //automatiquement=>a.toString()
}
```

Pour tout type d'objet, `toString()` est appelé automatiquement lors des affichages comme `println(obj)`

Multiplication et ajout (somme)

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...
    ...   ...
    //multiplier la fraction par un factor
    void multiplier(int factor){
        num = num*factor;
    }
    //ajouter f à la fraction
    void ajouter(Frac f){
        num = num*f.den+den*f.num;
        den = den*f.den;
    }
}
```

Multiplication et ajout (somme)

```
class Frac{
    int num;                    //numérateur
    int den;                    //dénominateur
    ...   ...
    ...   ...
    //multiplier la fraction par un factor
    void multiplier(int factor){
        num = num*factor;
    }
    //ajouter f à la fraction
    void ajouter(Frac f){
        num = num*f.den+den*f.num;
        den = den*f.den;
    }
}
```

$$\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}$$

Qu'affiche ce code ?

```
class Frac{
    ......
}
void setup(){
    Frac a=new Frac(1,5);
    Frac b=new Frac(2,5);
    a.ajouter(b);
    println(a);
    a.simplifier();
    println(a);
    a.multiplier(10);
    println(a);
    a.simplifier();
    println(a);
}
```

Qu'affiche ce code ?

```
class Frac{
    ......
}
void setup(){
    Frac a=new Frac(1,5);
    Frac b=new Frac(2,5);
    a.ajouter(b);
    println(a);                 //15/25
    a.simplifier();
    println(a);                 //3/5
    a.multiplier(10);
    println(a);                 //30/5
    a.simplifier();
    println(a);                 //6/1
}
```

Calcul correct $10 \cdot \left(\frac{1}{60} + \frac{1}{30} + \frac{1}{20}\right) = 1$

```
class Frac{
    ....
}
void setup(){
    Frac a=new Frac(1,60);
    Frac b=new Frac(1,30);
    Frac c=new Frac(1,20);
    a.ajouter(b);              //  1/60+1/30
    a.ajouter(c);              //  1/60+1/30+1/20
    a.multiplier(10);          //  10(1/60+1/30+1/20)
    if(a.egal(1))
        println("10(a+b+c) vaut 1");
    else
        println("10(a+b+c) différent de 1");
}
```

La notion théorique d'objet

Un objet

- possède un état constitué de valeurs (attributs)
- possède des actions (méthodes) qui peuvent agir sur ce cet état pour le modifier
  - les méthodes définissent le comportement d'un objet

Un objet est une instance (un exemplaire) d'une classe

La notion théorique de classe

- Une classe est un modèle pour construire des objets
  - classe = moule à objets

Toute classe définit :

- les caractéristiques (attributs) de ses objets
  - chaque objet d'une classe possède une copie de l'ensemble de ces attributs
- l'ensemble des actions (méthodes) que l'on peut effectuer sur les objets
- Des constructeurs : des méthodes qui initialisent les attributs avec des valeurs

Autre exemple : quel est le résultat du code ?

```
class Compte{
    int solde;
    Compte(){           //constructeur
        solde = 0;      //sans arguments
    }
    public void ajouter(int montant){
        solde = solde + montant;
    }
}
void setup(){
    Compte c = new Compte();
    c.ajouter(10);
    println(c.solde);
}
```

Variables `static`

Une variable déclarée `static` n'appartient pas aux objets.

- Elle appartient à la classe où elle est déclarée
- Elle est partagée par tous les objets

```
class Frac{
    static int maxDenominateur = 1000000;
...
}
void setup(){
    println(Frac.maxDenominateur);
    println(Integer.MAX_VALUE);//affiche la plus
        grande valeur qu'un entier peut avoir
}
```

Méthodes `static`

- Une méthode `static` n'est pas associée à un objet
- Pour l'appeler, on utilise le nom de la classe

```
int i = Integer.parseInt("123");
double x = Math.round(6.6);
double d = Math.pow(x,2);
```

Contineur la classe `Compte` pour :

1. Ajouter un nom de titulaire à la classe `Compte`
   - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
2. Accorder par défaut un crédit de 10 euros
3. Pouvoir retirer de l'argent
4. Pouvoir verser tout l'argent d'un compte `c1` dans un compte `c2`

Contineur la classe `Compte` pour :

1. Ajouter un nom de titulaire à la classe `Compte`
   - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
2. Accorder par défaut un crédit de 10 euros
3. Pouvoir retirer de l'argent
4. Pouvoir verser tout l'argent d'un compte `c1` dans un compte `c2`

Contineur la classe `Compte` pour :

1. Ajouter un nom de titulaire à la classe `Compte`
   - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
2. Accorder par défaut un crédit de 10 euros
3. Pouvoir retirer de l'argent
4. Pouvoir verser tout l'argent d'un compte `c1` dans un compte `c2`

Contineur la classe `Compte` pour :

1. Ajouter un nom de titulaire à la classe `Compte`
   - Ajouter une deuxième constructeur qui reçoit comme argument le nom du titulaire
2. Accorder par défaut un crédit de 10 euros
3. Pouvoir retirer de l'argent
4. Pouvoir verser tout l'argent d'un compte `c1` dans un compte `c2`