

Chapitre 2

Classes et objets

- Classes et objets, exemples
- Utilisation de classes prédéfinies
Vector, Date, GregorianCalendar
- Définition de classes
Rationnel..., Equation...
- Utilisation de classes définies pour ce cours
Clavier, Terminal, Fenetre, DessinsDeTetes

Programmation Java 2003/2004 - D.Pastre

Chap.2.1

Classes et objets

Les classes et les objets sont des entités informatiques correspondant respectivement à des concepts et à des instances de ces concepts, reliés par la relation binaire *est-un* (*isa*)

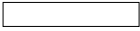


Les classes sont

- prédéfinies (en Java dans l'API - Application Programming Interface)
- ou - définies par le développeur
- définies par un autre développeur et mises à disposition

Programmation Java 2003/2004 - D.Pastre

Chap.2.2

Exemples

<u>objets</u>		<u>classes</u>	<u>en Java</u>
6	<i>est-un</i>	entier	Integer prédéfinie
7/5	<i>est-un</i>	rationnel	définie plus loin
abcd	<i>est-une</i>	chaîne de caractères	String prédéfinie
jean	<i>est-un</i>	étudiant	à définir
$3x^2 + 5x - 1$	<i>est-une</i>	équation du 2nd degré	définie plus loin
c1	<i>est-un</i>	compte bancaire	à définir
	<i>est-un</i>	rectangle	à définir
	<i>est-un</i>	cercle	à définir
rep1	<i>est-un</i>	répertoire	à définir
poo	<i>est-une</i>	UE	à définir
[6, jean, 	<i>est-un</i>	vecteur	Vector prédéfinie
2 février 2004	<i>est-une</i>	date	Date prédéfinie

Programmation Java 2003/2004 - D.Pastre

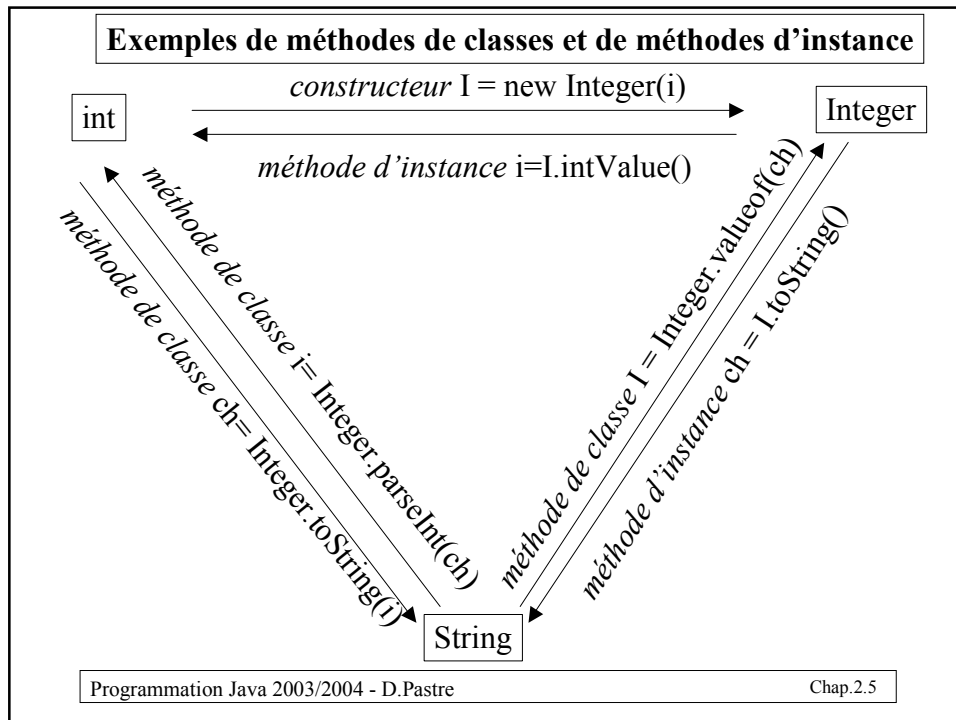
Chap.2.3

Dans une classe on trouve :

- les **caractéristiques** d'un ensemble d'objets définis par cette classe
exemples : - le numérateur et le dénominateur de ce rationnel
 - les coefficients de cette équation
 - les centre et le rayon de ce cercle
- des méthodes applicables à ces objets (dites **méthodes d'instances**)
exemples : - faire une opération sur ce rationnel
 - calculer les racines de cette équation
 - calculer le périmètre de ce rectangle ou de ce cercle
 - ajouter une personne à ce répertoire téléphonique
 - inscrire un étudiant de cette UE
- des méthodes statiques (dites **méthodes de classes**)
exemples : - calculer $n!$ où n est une variable de type entier (`int`)
 - faire la somme de deux rationnels
 - donner la liste de toutes les UE

Programmation Java 2003/2004 - D.Pastre

Chap.2.4



En Java, il y a aussi des **classes** dites **statiques** qui ne sont pas instanciables :

- classes prédéfinies contenant des fonctions ou des constantes utiles
exemple la classe `Math` avec
 - les fonctions `Math.sqrt`, `Math.abs`, ...
 - les constantes `Math.PI`, `Math.E`
- la classe `Clavier` (de la bibliothèque `bibliDP`)
- les classes de test définies par chaque développeur

Membres d'une classe

- attributs
 - **attributs d'instance**
exemples : - les coefficients d'une équation du 2nd degré
- le titulaire d'un compte bancaire
 - **attributs de classe** *exemples* : Math.PI, System.out
- fonctions
 - **constructeurs**
exemples : new Integer(6) représente l'entier 6
new Rationnel(3,4) représente le rationnel 3/4
new Compte("Dupond") crée un compte pour Dupond
 - méthodes
 - **méthodes d'instance**
exemple : r1.somme(r2)
 - **méthodes de classe** (mot-clé static)
exemple : somme(r1,r2)

Programmation Java 2003/2004 - D.Pastre

Chap.2.7

Exemples

- Utilisation de classes prédéfinies
- Définitions de classes
- Utilisation de classes définies pour ce cours

Programmation Java 2003/2004 - D.Pastre

Chap.2.8

Utilisation de classes prédéfinies

1. La classe Vector

```
...
import java.util.Vector;
/** Le programme Bonjour avec les notes dans un Vector */
public class BonjourVector {
    public static void main(String[] args) {
        ...
        Vector notes = new Vector();
        double note=Clavier.readDouble();
        while (note>0) {notes.addElement(new Double(note) );
            note = Clavier.readDouble();}
        ...
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.9

```
...
System.out.println("notes : "); System.out.println(notes);
...
int nbNotes = notes.size();
double somme = 0;
for (int i=0;i<nbNotes;i++)
    somme += ((Double)notes.elementAt(i)).doubleValue();
double moyenne = somme/nbNotes;
System.out.println("\nmoyenne : "+ moyenne);

}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.10

Variantes

```
...
/** Le programme Bonjour avec toutes les données dans un Vector */
Public class BonjourVectorVariante;
...
Vector donnees = new Vector();
donnees.addElement(nom); donnees .addElement(prenom);
donnees.addElement(new Integer(age)); donnees .addElement("ans");
// ou mieux donnees .addElement(age+" ans");
donnees.addElement("notes ");
int n = donnees.size();
...
for (double note = Clavier.readDouble(); note>0;
      note=Clavier.readDouble())
    donnees.addElement(new Double(note));
System.out.println(" donnees : " + donnees.toString());
...

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.11

```
...
int nbNotes = donnees.size()-n;
double somme = 0;
for (int i=n;i<nbNotes+n;i++)
    somme += ((Double)donnees.elementAt(i)).doubleValue();
double moyenne = somme/nbNotes;
System.out.println("\nmoyenne : "+ moyenne);

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.12

Comparaison

- **tableaux**

```
x = t[i]; // x est un objet ou une
// variable de type élémentaire
t[i]=x;
```

- **Vector**

```
x = v.elementAt(i); // x est un
// objet
v.setElementAt(x,i);
```

Avantages des *Vector*

- Pas de dimension fixe.
- Le *Vector* et tous ses éléments sont des objets, donc tout ce qui est applicable aux objets leur est applicable.
- On peut *retirer* ou *insérer* un élément sans devoir faire de décalage.

```
v.removeElement(i); // indice i
v.removeElement(x); // objet x (première occurrence)
insertElementAt(x, i);
```

Utilisation de classes prédéfinies

2. Les classes **Date** et **GregorianCalendar**

```
...
import java.util.Date;
import java.util.GregorianCalendar;
public class BonjourDate {
    public static void main(String[] args) {
        ...
        Date dAuj = new Date(); // date d'aujourd'hui
        System.out.println(dAuj);
        System.out.println("Il s'est écoulé " + dAuj.getTime()
            + " millisecondes depuis le 1er janvier 1970 Oh GMT");
        GregorianCalendar gcAuj = new GregorianCalendar();
        System.out.println(gcAuj); // possible mais ...
        System.out.println(gcAuj.getTime()); // plus simple !
    }
}
```

```

int anneeAuj = gcAuj.get(GregorianCalendar.YEAR);
System.out.println("Nous sommes en " + anneeAuj);
int annee = Clavier.readInt("année de naissance : ");
int age = anneeAuj - annee;
System.out.println("vous avez (ou vous aurez cette année "+age+" ans");
int mois = Clavier.readInt("numéro du mois de naissance : ");
int jour = Clavier.readInt("numéro du jour de naissance dans le mois : ");
GregorianCalendar gc = new GregorianCalendar(annee, mois, jour);
System.out.println("date de naissance : " + gc.getTime());
GregorianCalendar gcAnniv =
    new GregorianCalendar(anneeAuj,mois,jour);
Date dAnniv = gcAnniv.getTime(); // date de l'anniversaire
if (dAnniv.after(dAuj)) age -=1;
System.out.println("vous avez "+age+" ans");
long nbMilliSec = dAuj.getTime() - gc.getTime().getTime();
System.out.println("Vous avez déjà vécu "+nbMilliSec+" millisecondes");
}
}

```

Programmation Java 2003/2004 - D.Pastre
Chap.2.15

Définitions de classes

1. La classe Rationnel
2. Les classes Equation...
3. La classe Etudiant

Documentation de la classe **Rationnel** avec méthodes statiques (ou méthodes de classe)

```
public class Rationnel {  
    /** crée un rationnel connaissant le numérateur et le dénominateur  
     * non nul */  
    public Rationnel(int n,int d){}  
    /** renvoie une chaîne de caractères représentant ce rationnel avec la  
     * notation habituelle sous forme de fractions irréductibles */  
    public String toString(){  
        /** renvoie le pgcd de deux entiers */  
        private static int pgcd(int a,int b) {}  
        /** normalise ce rationnel en le mettant sous forme  
         * irréductible avec un dénominateur positif */  
        private void simplifier(){}  
    }
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.17

```
// on peut choisir de définir des méthodes statiques  
// (ou méthodes de classe)  
/** crée et renvoie un rationnel somme de deux rationnels */  
public static Rationnel somme(Rationnel a,Rationnel b){}  
/** crée et renvoie un rationnel produit de deux rationnels */  
public static Rationnel produit(Rationnel a,Rationnel b){}  
/** crée et renvoie un rationnel inverse d'un rationnel*/  
public static Rationnel inverse(Rationnel a){}  
// ou des méthodes d'instance  
/** crée et renvoie la somme du rationnel 'a' et de ce rationnel */  
public Rationnel somme(Rationnel a){}  
/** crée et renvoie le produit du rationnel 'a' et de ce rationnel */  
public Rationnel produit(Rationnel a){}  
/** crée et renvoie un rationnel inverse de ce rationnel */  
public Rationnel inverse(){} }
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.18

Classe de Test pour la classe **Rationnel**

```
package rationnels;
/** classe de test pour la classe rationnel.Rationnel */
public class TestRationnel {
    public static void main(String[] args) {
        Rationnel r1= new Rationnel(15,-4); System.out.println("r1 = "+r1);
        Rationnel r2= new Rationnel(-7,-6); System.out.println("r2 = "+r2);

        System.out.println("\nméthodes statiques (ou méthode de classe)")
        System.out.println(r1+" + "+r2+" = "+Rationnel.somme(r1,r2));
        System.out.println(r1+" * "+r2+" = "+Rationnel.produit(r1,r2));
        System.out.println("1/("+r1+")= "+Rationnel.inverse(r1));
        System.out.print("si r="+r1+"    "+"(1/r)*r= "
                        +Rationnel.produit(Rationnel.inverse(r1),r1));
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.19

```
System.out.println(" et r*(1/r)= "
                  +Rationnel.produit(r1,Rationnel.inverse(r1)));
```

```
System.out.println("\nméthodes d'instance");
```

```
System.out.println("ajouter "+r2+" à "+r1+" donne "+r1.somme(r2));
System.out.println("multiplier "+r1+" par "+r2+" donne "
                  +r1.produit(r2));
```

```
System.out.println("l'inverse de "+r1+" est "+r1.inverse());
System.out.println("multiplier l'inverse de "+r1+" par lui-même donne "
                  +Rationnel.produit(Rationnel.inverse(r1),r1));
System.out.println("multiplier "+r1+" par son inverse donne "
                  +Rationnel.produit(r1,Rationnel.inverse(r1)));}}
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.20

TestRationnel

Sortie

$r1 = -15/4$

$r2 = 7/6$

méthodes statiques (ou méthode de classe)

$-15/4 + 7/6 = -31/12$

$-15/4 * 7/6 = -35/8$

$1/(-15/4) = -4/15$

si $r = -15/4$ $(1/r)*r = 1$ et $r*(1/r) = 1$

méthodes d'instance

ajouter $7/6$ à $-15/4$ donne $-31/12$

multiplier $-15/4$ par $7/6$ donne $-35/8$

l'inverse de $-15/4$ est $-4/15$

multiplier l'inverse de $-15/4$ par lui-même donne 1

multiplier $-15/4$ par son inverse donne 1

Programmation Java 2003/2004 - D.Pastre

Chap.2.21

Définition de la classe Rationnel

```
public class Rationnel {
    private int num,den;
    public Rationnel(int n,int d){
        this.num=n;this.den=d;this.simplifier(); }
    public String toString(){
        if ((this.den==1)||((this.num==0))) return(""+this.num);
        else return(this.num+"/"+this.den);}
    private static int pgcd(int a,int b) {
        int r = a;
        while (r!=0) {r = a%b; a=b; b=r; }
        return(a);
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.22

```

private void simplifier(){
    if (this.num==0) this.den=1;
    else { int p = Rationnel.pgcd(this.num,this.den);
          this.num /= p;this.den /= p;
          if (this.den < 0) {this.num= -this.num; this.den=- this.den;}};}
public static Rationnel somme(Rationnel a,Rationnel b){
    return(new Rationnel(a.num*b.den+a.den*b.num,a.den*b.den)); }
public static Rationnel produit(Rationnel a,Rationnel b){
    return(new Rationnel(a.num*b.num,a.den*b.den)); }
public static Rationnel inverse(Rationnel a){
    return new Rationnel(a.den,a.num);
public Rationnel somme(Rationnel a){
    return(new Rationnel(a.num*this.den+a.den*this.num,a.den*this.den));
public Rationnel produit(Rationnel a){
    return(new Rationnel(a.num*this.num,a.den*this.den)); }
public Rationnel inverse(){return new Rationnel(this.den,this.num);}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.23

Documentation de la class EquationSta

```

package equation2nddegre;
/** Equation du second degré version statique */
public class EquationSta {
    /** renvoie une racine de l'équation 'a'x^2+ 'b'x+ 'c' */
    public static double racine1(double a, double b, double c) {}
    /** renvoie l'autre racine de l'équation 'a'x^2+ 'b'x+ 'c' */
    public static double racine2(double a, double b, double c) {}
    /** méthode privée, renvoie le discriminant de l'équation */
    private static double delta(double a, double b, double c) {}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.24

Test de la classe EquationSta

```
package equation2nddegre;
/** classe principale testant la classe EquationSta */
public class TestSta {
/** calcule les racines d'une équation dont les coefficients
 * sont donnés en paramètres */
public static void main(String[] args) {
    double a, b, c;
    double x1, x2;
    a = Double.parseDouble(args[0]);
    b = Double.valueOf(args[1]).doubleValue(); // pour varier, c'est pareil
    c = (Double.valueOf(args[2])).doubleValue();
    System.out.println("équation "+a+" x^2 + "+b+" x + "+c+" = 0");
    x1 = EquationSta.racine1(a,b,c); x2 = EquationSta.racine2(a,b,c);
    System.out.println("racines "+ x1 + " et " + x2);}}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.25

Définition de la classe EquationSta

```
package equation2nddegre;
public class EquationSta {
public static double racine1(double a, double b, double c) {
    double rac = (-b + Math.sqrt(delta(a,b,c)))/(2*a);
    return(rac);
}
public static double racine2(double a, double b, double c) {
    double rac = (-b - Math.sqrt(delta(a,b,c)))/(2*a);
    return(rac);
}
private static double delta(double a, double b, double c) {
    return b*b - 4*a*c;
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.26

Documentation de la classe **EquationObj**

```
package equation2nddegre;
/** Equation du second degré version objet */
public class EquationObj {
    /** crée une équation (instance de EquationObj) */
    public EquationObj(double a, double b, double c) {}
    /** renvoie une racine de cette équation */
    public double racine1() {}
    /** renvoie l'autre racine de cette équation */
    public double racine2() {}
    /** méthode privée, renvoie le discriminant de cette équation */
    private double delta() {}
    /** renvoie une représentation sous forme de chaîne de cette équation */
    public String toString() {}
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.27

Test de la classe **EquationObj**

```
package equation2nddegre;
/** classe principale testant la classe EquationObj */
public class TestObj {
    /** crée une équation dont les coefficients sont donnés en paramètres
     * et calcule ses racines*/
    public static void main(String[] args) {
        double a, b, c;
        double x1, x2;
        a = Double.parseDouble(args[0]);
        b = Double.valueOf(args[1]).doubleValue(); // pour varier, c'est pareil
        c = (Double.valueOf(args[2])).doubleValue();
        EquationObj equa = new EquationObj(a, b, c);
        System.out.println(equa.toString());
        x1 = equa.racine1(); x2 = equa.racine2();
        System.out.println("racines "+ x1 + " et " + x2); }}
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.28

Définition de la classe EquationObj

```
package equation2nddegre;
public class EquationObj {
    private double a, b, c;
    public EquationObj(double a, double b, double c) {
        this.a = a; this.b = b; this.c = c; }
    public double racine1() {
        double rac = (-this.b + Math.sqrt(this.delta()))/(2*this.a); return(rac);}
    public double racine2() {
        double rac = (-b - Math.sqrt(delta()))/(2*a); return(rac);}
    private double delta() {return this.b * this.b - 4 * this.a * this.c;}
    public String toString() {
        return("équation(version objet) "+
            this.a+" x^2 + "+this.b+" x + "+this.c+" = 0");}}}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.29

Documentation de la class EquationObj1

```
/** Equation du second degré version objet, deuxième version
 * le calcul des racines est mémorisé*/
public class EquationObj1 {
    /** crée une équation (instance de EquationObj1) */
    public EquationObj1(double a, double b, double c) {}
    /** résout cette équation et mémorise les racines */
    public void resolution() {}
    /** méthode privée, renvoie le discriminant de cette équation */
    private double delta() {}
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.30

```

/** renvoie une racine de cette équation */
public double getRacine1() {}
/** renvoie l'autre racine de cette équation */
public double getRacine2() {}
/** renvoie une représentation sous forme de chaîne
    * de cette équation
    * et de ses racines si elles ont été calculées */
public String toString() {}

```

Test de la classe EquationObj1

```

package equation2nddegre;
/** classe principale testant la classe EquationObj1 */
public class TestObj1 {
/** crée et résout une équation dont les coefficients
    * sont donnés en paramètres */
public static void main(String[] args) {
    System.out.println("exécution du main de TestObj1");
    double a, b, c;
    double x1, x2;
    a = Double.parseDouble(args[0]);
    b = Double.valueOf(args[1]).doubleValue(); // pour varier, c'est pareil
    c = (Double.valueOf(args[2])).doubleValue();
    EquationObj1 equa = new EquationObj1(a, b, c);

```



```

System.out.println(equa.toString());

equa.resolution();
System.out.println(equa.toString());

/* System.out.println("x1 = "+equa.racine1); non car privé
   déclarer racine1 public serait dangereux, encore plus que a,b,c */

System.out.println("\nappel de getRacine1() et getRacine2()");
x1 = equa.getRacine1();
x2 = equa.getRacine2();
System.out.println("x1 = " + x1 + " et x2 = " + x2);

}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.33

Définition de la classe EquationObj1

```

package equation2nddegre;
public class EquationObj1 {
    private double a, b, c;
    private double racine1 = Double.NaN;
    private double racine2 = Double.NaN;
    public EquationObj1(double a, double b, double c) {
        this.a = a; this.b = b; this.c = c; }
    public void resolution() {
        this.racine1 = (-this.b + Math.sqrt(this.delta()))/(2*this.a);
        this.racine2 = (-this.b - Math.sqrt(this.delta()))/(2*this.a);}
    private double delta() {return this.b * this.b - 4 * this.a * this.c;}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.34

```

public double getRacine1() { return(this.racine1); }
public double getRacine2() { return(this.racine2); }

public String toString() {
    String chaine = "\néquation (version objet avec attributs racines) "+
        this.a+" x^2 + "+this.b+" x + "+this.c+" = 0";
    if (Double.isNaN(this.racine1))
        chaine = chaine + "\n il n'y a pas (ou pas encore) de racines";
    else chaine += "\nracines "+ this.racine1+ " et "+this.racine2;
    return(chaine); }

```

La classe **Etudiant**

```

/** classe permettant de gérer les données d'un étudiant */
public class Etudiant {
    private String nom, prenom;
    private int age;
    private Vector lesNotes = new Vector();
    /** crée un étudiant connaissant ses noms, prénoms année de naissance */
    public Etudiant(String nom, String prenom, int annee) {
        this.nom = nom;
        this.prenom=prenom;
        this.age = 2003-annee;
    }
    /** ajoute une note à cet étudiant */
    public void ajouterNote(double note) {
        this.lesNotes.addElement(new Double(note));
    }
}

```

```

/** renvoie la moyenne des notes de cet étudiant */
public double moyenne() {
    int nbNotes = this.lesNotes.size();
    if (nbNotes == 0) return 0;
    double somme = 0;
    for (int i=0;i<nbNotes;i++)
        somme += ((Double)lesNotes.elementAt(i)).doubleValue();
    return somme/nbNotes;
}

/** renvoie une chaine de caractères représentant les données
 * de cet étudiant */
public String toString() {
    return this.prenom + " " + this.nom + "\nage : " + this.age
        + "\nnotes : " + this.lesNotes;
}
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.37

```

/** classe de test pour la classe Etudiant */
public class BonjourEtudiant {
    public static void main(String[] args) {
        String nom = Clavier.readString("tapez votre nom : ");
        String prenom = Clavier.readString("tapez votre prenom : ");
        int annee = Clavier.readInt("tapez votre année de naissance : ");
        Etudiant e = new Etudiant(nom, prenom, annee);
        System.out.println("\nbonjour " + e);
        System.out.println(
            "\ntapez les notes séparées par des blancs ou des \"Entrée\"");
        System.out.println("tapez un nombre négatif pour terminer la lecture");
        for (double note = Clavier.readDouble(); note>0;note=Clavier.readDouble())
            e.ajouterNote(note);
        System.out.println("\n" + e);
        System.out.println("moyenne : " + e.moyenne());
    }
}

```

Programmation Java 2003/2004 - D.Pastre

Chap.2.38

La classe **Etudiant1**

```
/** classe permettant de gérer les données d'un étudiant
 * (moyenne comprise) */
public class Etudiant1 {
    private ...
    private double moyenne = 0;
    /** crée un étudiant connaissant ses noms, prénoms année de naissance */
    public Etudiant1(String nom, String prenom, int annee) {...}
    /** ajoute une note à cet étudiant et met sa moyenne à jour */
    public void ajouterNote(double note) {
        this.lesNotes.addElement(new Double(note));
        this.moyenne = this.moyenne();
    }
    /** renvoie la moyenne des notes de cet étudiant */
    private double moyenne() {...}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.39

```
/** renvoie une chaine de caractères représentant les données
 * de cet étudiant (moyenne comprise) */
public String toString() {
    return this.prenom + " " + this.nom + "\nage : " + age
        + "\nnotes : " + this.lesNotes
        + "\nmoyenne : " + this.moyenne ;
}
```

```
/** classe de test pour la classe Etudiant1 */
public class BonjourEtudiant1 {
    public static void main(String[] args) {
        ...
        e.ajouterNote(note); // lamoyenne est calculée à chaque fois
        ...
        System.out.println("\n" + e); // la moyenne sera affichée
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.40

Utilisation de classes définies pour ce cours

1. La classe Clavier (statique)
2. La classe Terminal
3. La classe FenetreGroupe
4. La classe DessinsDeTetes

Programmation Java 2003/2004 - D.Pastre

Chap.2.41

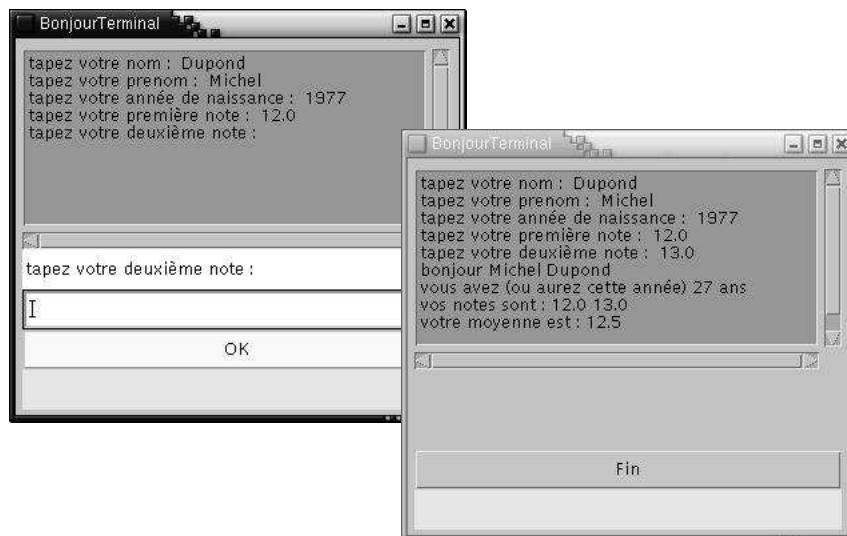
Utilisation de la classe **Terminal**

*Une instance de la classe **Terminal** est une Fenetre dans laquelle il est possible de faire des lectures et des affichages*

```
// classe écrite par Yannick Parchemal
import up5.mi.pary.term.Terminal ;
public class BonjourTerminal {
    public static void main(String[] args) {
        Terminal term = new Terminal("BonjourTerminal",600,300);
        String nom = term.readString("tapez votre nom : ");
        ...
        int annee = term.readInt("tapez votre année de naissance : ");
        ...
        double note1 = term.readDouble("tapez votre première note : ");
        double note2 = term.readDouble("tapez votre deuxième note : ");
        term.println("bonjour "+ prenom+" "+nom);
        term.println("vous avez (ou aurez cette année) "+age+" ans "+ age);
        ...
        term.end();
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.42



Programmation Java 2003/2004 - D.Pastre

Chap.2.43

Utilisation d'une fenêtre de saisie des données

```
/** classe permettant la gestion d'un groupe d'étudiants
 * - les matières sont données le programme principal
 * - les données relatives aux étudiants sont gérées à partir d'une
 * fenetre */
```

```
import bibliDP.fenetreDeSaisie.*;
public class Gestion {
    public static void main(String[] args) {
        Groupe gr = new Groupe("licence");
        Matiere[] tabMatiere = new Matiere[4];
        tabMatiere[0] = new Matiere("physique");
        tabMatiere[1] = new Matiere("math");
        tabMatiere[2] = new Matiere("chimie");
        tabMatiere[3] = new Matiere("anglais");
        FenetreGroupe f = new FenetreGroupe(gr,tabMatiere);
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.44

licence

nom prenom année de naissance

physique math chimie anglais

Nom : Dupond Prenom : Jean age : 19 ans

relevé de notes :

physique : 14
math : 16
chimie : 7
anglais : 11

note finale (moyenne) : 12

données mémorisées et relevé de notes enregistré dans le fichier Dupond_Jean.

===== notes finales du groupe licence =====

Martin Paul : 12,5
Dupond Jean : 12
moyenne des notes finales des 2 étudiants du groupe licence : 12,25

===== moyennes des 2 notes de chaque matière =====

physique : 14,5
math : 16,5
chimie : 7,5
anglais : 10,5

Programmation Java 2003/2004 - D.Pastre

Chap.2.45

Utilisation de la classe **DessinsDeTetes**

```
import java.awt.Frame;
import dessinsDeTetes.*;
public class QuelquesTetes {
    public static void main(String[] args) {
        Frame f = new Frame("quelques têtes"); // classe prédéfinie
        f.setSize(500,500);
        f.addWindowListener(new Fermeture()); // pour pouvoir arrêter le
            // programme en cliquant sur la case de fermeture
        Dessin d = new Dessin();
        f.add(d); f.show();
        d.ajouterObjet(new Tete()); // tête standard, en haut à gauche
        d.ajouterObjet(new Tete("Jean",150,120,75,150,-1));
        d.ajouterObjet(new Tete("Nicole",300,150,80,50,1));
        d.ajouterObjet(new Tete("Jacques",400,150,80,50,-1));
        d.ajouterObjet(new Tete("Marie",130,360,150,150,0));
        d.ajouterObjet(new Tete("Camille",375,350,200,200,1));
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.2.46

