Eléments de cours java

Chap1-Les classes

Syntaxe de déclaration de classes

public class *NomClasse*{

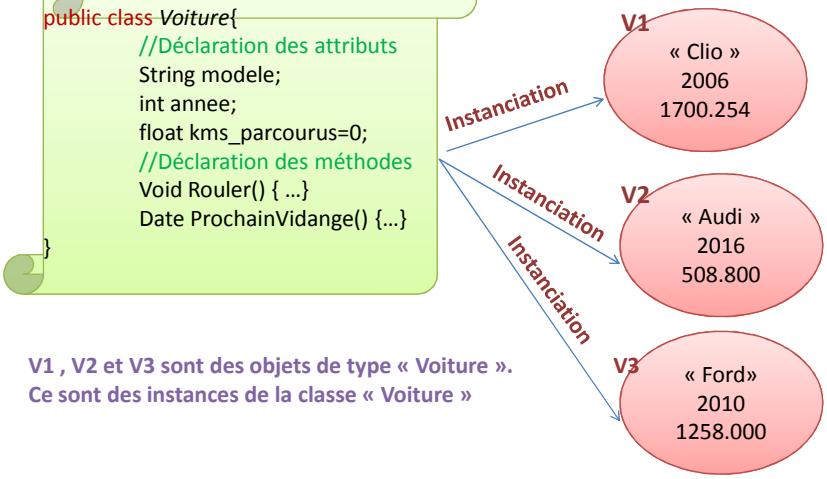
```
        //Déclaration des attributs
        static final int pi=3.14;
        protected final int taux=50;
        public int a=12;
      //Déclaration des méthodes
      static final int  F() { …}
      protected void G() {…}
      public int H() {…}
}
```

Public, static, protected, final et public sont des spécificateurs d'accès

1er principe de l'OO :l'instanciation

Une classe est un moule d'objets

```
public class Voiture{
        //Déclaration des attributs
        String modele;
        int annee;
        float kms_parcourus=0;
        //Déclaration des méthodes
        Void Rouler() { …}
        Date ProchainVidange() {…}
}
```



V1 , V2 et V3 sont des objets de type « Voiture ».
Ce sont des instances de la classe « Voiture »

L'instanciation (cntd)

Déclaration d'instances

```
Voiture v1; //(1) Déclaration d'une référence vers un objet Voiture
v1 = new Voiture(); /*(2) Création de l'espace mémoire nécessaire pour
    stocker un objet de type Voiture
                        (3) Invocation d'une méthode spécifique appelée
    «constructeur » utile pour l'initialisation de l'objet
int nb=10;
Voiture tab_v [] = new Voiture [nb]; //Création d'un tableau de 10 objets
                        de type «  Voiture »
Pour accéder aux membres des l'objets
v1.modele = « Mercedes »;
System.out.println(« Kms :» +v1. kms_parcourus);
Date aujourdhui = new Date() //Récupérer la date système Cf. la classe Date
Date  vidange = v1. ProchainVidange(); //Récupérer la date du prochain vidange de v1
if (vidange.after(aujourdhui ))
        //On peut rouler sans problème
        v1.Rouler();
else  System.out.println(« Vidange nécessaire! »);
```

L'instanciation (cntd)

Où instancier?

- Dans un main

```
public class Concessionnaire {
      ……
      public static void main( String args[]){
            Voiture v=new Voiture();
            //Faire le nécessaire ……
      }
}
```

- Dans une méthode d'une classe (de toute façon le main est une méthode!)

```
public class Concessionnaire {
      ……
      public  course(){
            Voiture v=new Voiture();
            v.Rouler();
      }
}
```

Exemple: la classe Date

java.util.Date

public class Date extends Object implements Serializable, Cloneable, Comparable<Date>

//Class constructors

Date()

This constructor allocates a Date object and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.

Date(long date)

This constructor allocates a Date object and initializes it to represent the specified number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.

//Class methods

after(Date when) This method tests if this date is after the specified date.

boolean before(Date when) This method tests if this date is before the specified date.

Object clone() This method return a copy of this object.

int compareTo(Date anotherDate) This method compares two Dates for ordering.

boolean equals(Object obj) This method compares two dates for equality.

long getTime() This method returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this Date object.

int hashCode() This method returns a hash code value for this object.

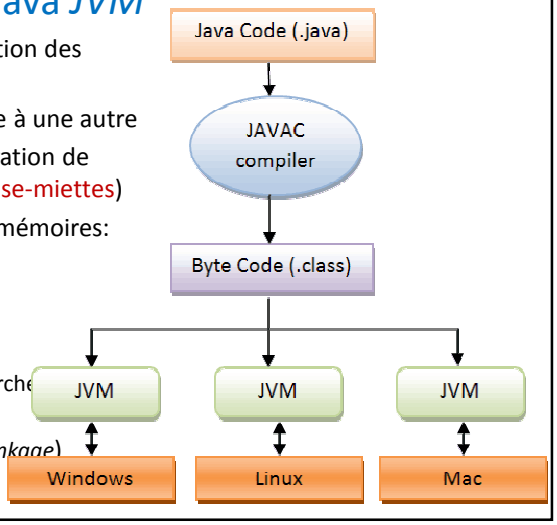
void setTime(long time) This method sets this Date object to represent a point in time that is time milliseconds after January 1, 1970 00:00:00 GMT.

String toString() This method converts this Date object to a String of the form.

Chap2-Spécificités du langage java

La machine virtuelle java *JVM*

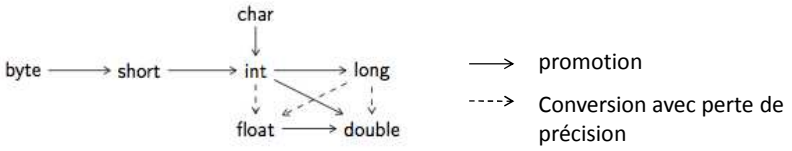
- Est un environnement d'exécution des applications java
- Est différente d'une plateforme à une autre
- Se charge de l'allocation / libération de l'espace pour les objets (ramasse-miettes)
- Dispose de plusieurs types de mémoires:
  - Les registres
  - Les tas (heap)
  - Une pu plusieurs piles (stacks)
  - Une zone de méthodes
- Chargement des classes (recherche les fichiers .class et du bytecode)
- Procède à l'édition des liens (*linkage*)
- Charge la police de sécurité



Les types primitifs

- Les conversions
- Les constantes
- Les types enumérés

To Be Done



Les tableaux

java.lang.reflect.Array

Tableaux unidimensionnels

```
type  tab[];
tab=new type[nb];
type  tab[]={E1,E2,E3};
```

Parcours ➔
```
for (type compteur  : tab)
    System.out.println(type); //en lecture seule
```

➔
```
for (int i=0;i<tab.length; i++)
    System.out.println(tab[i]);
```

Méthodes de la classe System

- **public static void** arraycopy(Object src, **int** srcPos, Object dest, **int** destPos, **int** l)
- **static** Object get(Object t, **int** index)
- **static** xxx getXxx(Object t, **int** index)
- **static void** set(Object t, **int** index, Object valeur)
- **static void** setXxx(Object t, **int** index, xxx z)

Les tableaux (Cntd)

Tableaux multidimensionnels

```
type matrice [][];
type matrice [][]={{E1,E2},{E3,E4,E5},{E6,E7,E8,E9}};
//ce sera une matrice de 3 lignes avec un nombre de colonnes variables
type matrice [][]= new type[4][]; //ce sera une matrice de 4 lignes avec un nombre de colonnes variables
Matrice[0] = new type [8]; //la première ligne contiendra un tableau de 8 cases
Matrice[i] = new type [i]; //la ième ligne contiendra un tableau de i cases
```

Exemple Une méthode qui retourne un booléen indiquant si une matrice est symétrique?

Les chaînes de caractères

La classe String

Construction

- String(**byte**[] bytes)
- String(**byte**[] bytes, **int** offset, **int** length)
- String(**char**[] value, **int** offset, **int** count)
- String(String value)

Comparaison

- **int** compareTo(String anotherString)
- **boolean** equals(Object anObject)

Sous-chaines

- **char** charAt(**int** i)
- String substring(**int** d)
- **boolean** startsWith(String prefix)
- **boolean** startsWith(String prefix, **int** i)

Les chaînes de caractères (Cntd)

Conversion

- String toLowerCase()
- String trim()
- String replace(**char** ac, **char** nc)
- **byte** []getBytes()
- **static** String valueOf(**int** i)

Recherche

- **int** indexOf(**int** ch)
- **int** indexOf(String str, **int** fromIndex)
- **int** lastIndexOf(**int** ch)
- … Cf classe String

Surcharge de fonctions

➔ Plusieurs fonctions qui portent le même nom avec des signatures différentes

```
public class Cercle{
        float  centre_x, centre_y;
        char couleur;
      public void Dessiner() {
                    //Dessiner le cercle
                    }
    public void Dessiner(char coul) {
                    couleur = coul;
                    //Dessiner le cercle selon la couleur en paramètre
                    }
    public void Dessiner(String s) {
                    //Dessiner le cercle avec la légende s
                    }}
```

- La résolution de l'appel se fait au moment de la compilation
- Le compilateur choisit parmi les fonctions candidates celle la moins couteuse en conversions (évaluation de la distance)
- S'il existe 2 fonctions candidates à la même distance le compilateur soulève alors une ambiguïté.

L'opérateur Ellipse

➔ Permet de spécifier un nombre variable d'arguments (obligatoirement le dernier de la liste)

```
public class Mathematique{
   public int Somme(int …a) { //a est considéré comme un tableau d'entier
                    int s=0;
                    for  (int i :a)
                                s+=i;
                    return s;
                    }
    public void Quelconque() {
                    System.out.println(Somme (1,2,3);
                    int tab[] = new int [10]; …
                    if (Somme (tab) >100)
                                System.out.println(« C'est gagné »);
                    }
     }
```

Robot

Attributs :

- ***nom*** : chaîne de caractères
- position : ***x*** et ***y*** : entiers
- ***direction*** : caractère (‘E’ = Est,’W’ = Ouest, ‘N’=Nord et ‘S’= Sud)

Méthodes :

- *Creer_Robot (nom, position, direction).* Le nom est obligatoire mais on peut ne pas spécifier la position et la direction, qui sont définis par défaut à **(0,0)** et **"Est"**.
- *Avance( )* avance le robot d’un seul pas en avant.
- *Avance **(…)*** qui avance le robot de ***n*** pas en avant, le nombre de pas est passé en paramètre.
- *Droite ( )* retourne le robot de 90° vers la droite. Les robots ne peuvent pas tourner à gauche.
- *Afficher( )* qui affiche l’état d’un robot en détail.