Algorithmique & Programmation
Semestre 2 ST
Les types agrégés

Problématique

- Fréquente association fonctionnelle des données manipulées dans le monde réel
  -> Association de ces données dans les programmes informatiques les modélisant

- **Exemples**
  - Une date: Un numéro d'année, un numéro de mois et un numéro de jour
  - Une heure: Un numéro d'heure, un numéro de minute et un numéro de seconde
  - Une position dans un espace à 2 dimensions: Une abscisse et une ordonnée

  - Une position dans un espace à 3 dimensions: 3 coordonnées usuellement nommées x, y et z
  - Un nombre complexe: Une partie réelle et une partie imaginaire
  - Une adresse: Un nom, un prénom, un numéro dans la rue, un nom de rue, un code postal, une ville et un pays

- Liaison de ces données en agrégats permettant de les manipuler en tant qu'une seule donnée (variable) composite
  -> Un bon moyen pour rendre plus aisé le développement:
  - Structuration du stockage de l'information
  - Manipulation en une seule variable de tout un lot d'informations

Solution: Les types agrégés

Types agrégés en langage algorithmique

- Type agrégé: Type de définition de variable regroupant en fait plusieurs "champs" (sous-variables, variables membres) élémentaires accessibles individuellement

- Dénomination usuelle: "structure"

- Syntaxe:

```
structure nomStructure
  champ1 : type
  champ2 : type
  ...
  champN : type
fin structure
```

  - nomStructure: Le nom donné au type agrégé
  - champ1, champ2, ..., champN: Les N noms des N champs définis au sein du type agrégé accompagnés individuellement d'un type spécifiant ce qu'ils sont à même de contenir (types éventuellement différents les uns des autres)

- Définition d'une variable selon un type agrégé selon la même syntaxe que celle des types "classiques":

```
Locales
  variable : nomStructure
```

- Syntaxe d'accès aux champs d'une variable de type agrégé:

```
nomVariable.champ1
```

- nomVariable: Une variable de type agrégé
- champ1: Un des champs de ce type agrégé

Choix d'un nom de structure

- Lettres non accentuées et _
- Chiffres (sauf premier caractère)
- **Conventions**
  - Choix des noms des structures de manière explicite vis à vis des informations stockées
  - Emploi exclusif de minuscules
  - Si utilisation d'un nom composé, initiale en majuscule pour chaque composante à l'exception de la première
- Exemples: `date`, `adressePostale`

Exemples de structures

```
structure date
  jour : entier
  mois : entier
  annee : entier
fin structure
```

```
structure heure
  h : entier
  mn : entier
  s : entier
fin structure
```

```
structure position2D
  abscisse : reel
  ordonnee : reel
fin structure
```

```
structure position3D
  x : reel
  y : reel
  z : reel
fin structure
```

```
structure complexe
  partieReelle : reel
  partieComplexe : reel
fin structure
```

```
structure adressePostale
  nom : chaine
  prenom : chaine
  numero : entier
  rue : chaine
  zip : entier
  ville : chaine
  pays : chaine
fin structure
```

Exemple d'algorithme avec structures

- Problème: Calculer et afficher la distance entre deux positions du plan

```
{ Calcul de la distance entre deux positions du plan }

{ Type agrege de stockage d'une position du plan     }
```

```
structure position2D
 abscisse : reel
 ordonnee : reel
fin structure

{ Programme principal                                    }

action principale()
   locales
    p1 : position2D
    p2 : position2D
    distance : réel
    dx : réel
    dy : réel
  Ecran.afficher("SVP, coordonnees point 1 ?")
  p1.abscisse <- Clavier.saisirReel()
  p1.ordonnee <- Clavier.saisirReel()
  Ecran.afficher("SVP, coordonnees point 2 ?")
  p2.abscisse <- Clavier.saisirReel()
  p2.ordonnee <- Clavier.saisirReel()
  dx <- p2.abscisse-p1.abscisse
  dy <- p2.ordonnee-p1.ordonnee
  distance <- sqrt(dx*dx+dy*dy)
  Ecran.afficher("Distance = ",distance)
fin action
```

DistancePositions2D.lda

Types agrégés en langage Java

- Description avant ou après la fonction main dans la classe application qui la contient
- Pas de définition dans le main

- Dénomination en langage Java: classe ou "class"

- Syntaxe:

```
static class NomType {
  type variable1;
  type variable2;
  ...
  type variableN; };
```

  - NomType: Nom du type déclaré
  - variable1, variable2, ..., variableN: N sous-variables ("variable membre") définies selon les types spécifiés (éventuellement différents les uns des autres)

- Pour le langage Java, syntaxe particulière lors de la déclaration d'une variable d'un type agrégé:

```
NomType nomVariable = new NomType();
```

  - Utilisation obligatoire du new à la déclaration de toute variable de type agrégé

- Syntaxe d'accès aux champs d'une variable de type agrégé (identique à la syntaxe algorithmique):

```
nomVariable.variable1
```

  - nomVariable: Une variable de type agrégé
  - variable1: Une des variables membres de ce type agrégé

Choix d'un nom de classe

- Lettres non accentuées
- Utilisation éventuelle de chiffres (sauf premier caractère)
- **Conventions**
  - Choix des noms des classes de manière explicite vis à vis des informations stockées
  - Minuscules avec initiale en majuscule
  - Si utilisation d'un nom composé, initiale en majuscule pour chaque composante y compris la première
- Exemples : `Date`, `AdressePostale`

Exemples de classes

```
static class Date {
  int jour;
  int mois;
  int annee; };
```

```
static class Heure {
  int h;
  int mn;
  int s; };
```

```
static class Position2D {
  float abscisse;
  float ordonnee; };
```

```
static class Position3D {
  double x;
  double y;
  double z; };
```

```
static class Complexe {
  float partieReelle ;
  float partieComplexe ; };
```

```
static class AdressePostale {
  String nom;
  String prenom;
  int numero;
  String rue;
  int zip;
  String ville;
  String pays; };
```

Exemple de programme Java avec classes

- Problème: Calculer et afficher la distance entre deux positions du plan

```
/* Calcul de la distance entre 2 positions du plan */

public class DistancePositions2D {

/* Type agrege de stockage d'une position du plan  */

   static class Position2D {
     double abscisse;
     double ordonnee; };

/* Programme principal                              */

   public static void main(String [] args) {
     Position2D p1 = new Position2D();
     Position2D p2 = new Position2D();
     double distance;
     double dx;
     double dy;
    Ecran.afficherln("SVP, coordonnees point 1?");
    p1.abscisse = Clavier.saisirDouble();
    p1.ordonnee = Clavier.saisirDouble();
    Ecran.afficherln("SVP, coordonnees point 2?");
    p2.abscisse = Clavier.saisirDouble();
    p2.ordonnee = Clavier.saisirDouble();
    dx = p2.abscisse-p1.abscisse;
    dy = p2.ordonnee-p1.ordonnee;
    distance = Math.sqrt(dx*dx+dy*dy);
    Ecran.afficherln("Distance = ",distance);
  }
}
```

DistancePositions2D.java - Exemple d'exécution

Initialisation automatique des champs

- Intérêt supplémentaire des types agrégés: Réalisation d'une initialisation de leurs champs à la déclaration du type avec possibilité de choisir la valeur d'initialisation
  -> Initialisation implicitement réalisée donc certaine
    -> Moins de risque d'oubli
  -> Plus d'initialisation explicite à chaque déclaration de variable
    -> Economie de code

Syntaxe

Langage algorithmique	Langage Java

structure nomStructure
champ1 : type <- valeur1
champ2 : type <- valeur2
...
champN : type <- valeurN
fin structure
	static class NomType {
	type variable1 = valeur1;
	type variable2 = valeur2;
	...
	type variableN = valeurN; };

Exemples

```
structure position3D
  x : reel <- 0.0
  y : reel <- 0.0
  z : reel <- 0.0
fin structure
```

```
static class Position3D {
  double x = 0.0;
  double y = 0.0;
  double z = 0.0; };
```

```
structure date
  jour : entier <- 1
  mois : entier <- 1
  annee : entier <- 1901
fin structure
```

```
static class Date {
  int jour = 1;
  int mois = 1;
  int annee = 1901; };
```

```
structure adresse
  nom : chaine <- ""
  prenom : chaine <- ""
  numero : entier <- 0
  rue : chaine <- ""
  zip : entier <- 0
  ville : chaine <- ""
fin structure
```

```
static class Adresse {
  String nom = "";
  String prenom = "";
  int numero = 0;
  String rue = "";
  int zip = 0;
  String ville = "";
  String pays = ""; };
```

Recommandation

- Toujours faire une initialisation des champs des types agrégés

Exemple de programme Java avec classe (type agrégé) avec initialisation des champs

```
/* Test de classes sans et avec initialisation */

public class TestInitialisationClasse {

/* Stockage d'une date sans initialisation     */

   static class DateSansInit {
```

```
      int jour;
      int mois;
      int annee; };

/* Stockage d'une date avec initialisation     */

   static class DateAvecInit {
      int jour = 1;
      int mois = 1;
      int annee = 1901; };

/* Programme principal                         */

   public static void main(String [] args) {
      DateSansInit d1 = new DateSansInit();
      DateAvecInit d2 = new DateAvecInit();
     Ecran.afficher("Date 1 : ");
     Ecran.afficherln(d1.jour,":",
                      d1.mois,":",
                      d1.annee);
     Ecran.afficher("Date 2 : ");
     Ecran.afficherln(d2.jour,":",
                      d2.mois,":",
                      d2.annee);
   }
}
```

TestInitialisationClasse.java - Exemple d'exécution

Type agrégé dans un type agrégé

- Utilisation possible d'un type agrégé déjà déclaré, comme n'importe quel autre type, lors de la déclaration d'un **autre** type agrégé
- ***Attention***: En langage java, dans la déclaration de type, initialisation par new obligatoire pour chaque variable membre de type agrégé

Exemples

```
structure position2D
  x : reel <- 0.0
  y : reel <- 0.0
fin structure

structure cercle
  centre : position2D
  rayon : reel <- 1.0
fin structure
```

```
static class Position2D {
  double x = 0.0;
  double y = 0.0; };

static class Cercle {
  Position2D centre = new Position2D();
  double rayon = 1.0; };
```

Exemple: Programme Java de manipulation d'un temps avec année, mois, jour, heure, minute et seconde

```
/* Type agrege constitue de types agreges    */

public class TypeAgregeDeTypesAgreges {

/* Stockage d'une date avec initialisation   */

   static class Date {
     int jour = 1;
     int mois = 1;
     int annee = 1901; };

/* Stockage d'une heure avec initialisation  */

   static class Heure {
     int seconde = 0;
     int minute = 0;
     int heure = 0; };

/* Stockage d'un temps compose               */
/* d'une Date et d'une Heure                 */
/* Initialisation obligatoire                */
/* des deux variables membres agregees       */

   static class Temps {
     Heure heure = new Heure();
     Date date = new Date(); };

/* Programme principal                       */

   public static void main(String [] args) {
     Temps temps = new Temps();
    Ecran.afficherln(temps.date.annee,":",
                     temps.date.mois,":",
                     temps.date.jour,":",
                     temps.heure.heure,":",
                     temps.heure.minute,":",
                     temps.heure.seconde);
  }
}
```

TypeAgregeDeTypesAgreges.java - Exemple d'exécution

```
{ Stockage d'une date avec initialisation   }

structure typeDate
  jour : entier <- 1
```

```
  mois : entier <- 1
  annee : entier <- 1901
fin structure

{ Stockage d'une heure avec initialisation  }

structure typeHeure
  seconde : entier <- 0
  minute : entier <- 0
  heure : entier <- 0
fin structure

{ Stockage d'un temps compose               }
{ d'un typeDate et d'un typeHeure           }
{ Initialisation obligatoire                }
{ des deux variables membres agregees       }

structure typeTemps
  heure : typeHeure
  date : typeDate
fin structure
```

TypeAgregeDeTypesAgreges.lda

Affectation et test d'égalité entre variables de type agrégé

Affectation

- Opération d'affectation utilisable sur les variables de type agrégé
- ATTENTION, comportements différents en langage algorithmique et en langage Java

- En langage algorithmique (utilisation du signe <-)
  Affectation automatique de chaque champ de la variable cible avec la valeur du champ correspondant de la variable source
  -> Fonctionnement "normal"

- En langage Java
  Confusion totale et irréversible entre les deux variables après l'affectation
  -> Variable unique désignée par deux noms
  -> Toute modification apportée à l'une est apportée à l'autre.
  -> Dangereux

  Affectation sans confusion à tout jamais
  -> Programmation explicite de l'affectation champ par champ

```
/* Affectation entre variable agregees        */

public class AffectationVariablesAgregees {

/* Stockage d'une position en 3 dimensions    */

   static class Position3D {
     double x = 0.0;
     double y = 0.0;
     double z = 0.0; };

/* Programme principal                        */

   public static void main(String [] args) {
     Position3D p1 = new Position3D();
     Position3D p2 = new Position3D();
    Ecran.afficherln("Variables après déclaration");
    Ecran.afficherln("P1 : ",p1.x," ",p1.y," ",p1.z);
    Ecran.afficherln("P2 : ",p2.x," ",p2.y," ",p2.z);
    Ecran.afficherln();
    p2.x = 1.0;
    p2.y = 1.0;
    p2.z = 1.0;
    Ecran.afficher("Après affectations sur ");
    Ecran.afficherln("les variables membres de P2");
    Ecran.afficherln("P1 : ",p1.x," ",p1.y," ",p1.z);
    Ecran.afficherln("P2 : ",p2.x," ",p2.y," ",p2.z);
    Ecran.afficherln();
    p1 = p2;
    Ecran.afficherln("Après affectation de P2 à P1");
    Ecran.afficherln("P1 : ",p1.x," ",p1.y," ",p1.z);
    Ecran.afficherln("P2 : ",p2.x," ",p2.y," ",p2.z);
    Ecran.afficherln();
    p1.x = 4.0;
    p2.y = 2.0;
    Ecran.afficher("Après affectations sur les ");
    Ecran.afficherln("variables membres de P1 & P2");
    Ecran.afficherln("P1 : ",p1.x," ",p1.y," ",p1.z);
    Ecran.afficherln("P2 : ",p2.x," ",p2.y," ",p2.z);
    Ecran.afficher("-> Il n'y a plus qu'une seule ");
    Ecran.afficherln("variable avec deux noms");
  }
}
```

AffectationVariablesAgregees.java - Exemple d'exécution

Test d'égalité

- Langage algorithmique: Pas de problème (utilisation du signe =)

- Langage Java: ATTENTION, ne marche pas
  Solution: Implanter un test champ par champ

```
/* Tests d'egalite entre 2 Position2D            */

public class TestEgalitePosition2D {

/* Type agrege de stockage des informations      */
/* relatives a une position en deux dimensions   */

   static class Position2D {
     double x = 0.0;
     double y = 0.0; };

/* Programme principal                           */

   public static void main(String [] args) {
    Position2D p1 = new Position2D();
    Position2D p2 = new Position2D();
     boolean egal;
    egal = (p1 == p2);
    Ecran.afficherln(egal);
    egal = ( ( p1.x == p2.x ) && ( p1.y == p2.y ) );
    Ecran.afficherln(egal);
  }
}
```

TestEgalitePosition2D.java - Exemple d'exécution

Auteur: Nicolas JANEY
UFR Sciences et Techniques
Université de Besançon
16 Route de Gray, 25030 Besançon
nicolas.janey@univ-fcomte.fr