Initiation au développement - SDA : TP1 (Deux séances)

DUT/INFO/R1_01
version 2021-2022 (PN BUT 2021)

Table des matières

1. Environnement de travail

Tout comme dans les TPs d’IAP, réaliser les actions suivantes pour configurer votre environnement de travail.

1. Lancer BlueJ.
2. Créer et nommer (nous vous suggérons TP1_NumeroDeGroupe_Nom_Prenom) un nouveau projet où vous souhaitez placer vos sources java.

2. Les enregistrements

Les types que l’on a vus jusqu'à présent sont pratiques, mais limités. Nous nous sommes d’abord principalement occupés des types de bases : le type entier `int`, le type flottant `double`, etc. Nous avons également pu manipuler des variables contenant des chaînes de caractères, c’est-à-dire de type `String`.

Nous avons également rencontré des types qui ne sont pas tout à fait des types de base, mais qui se construisent à partir d’eux : les tableaux de `int`, tableaux de `String`, etc. Une variable de type tableau de `int` est en quelque sorte une assemblée de variables de type `int`, puisque chacune des cases du tableau est de type `int` et peut se manipuler comme un `int`.

L’idée d’un type enregistrement est de généraliser le concept de type : définir un type enregistrement, c’est assembler des types déjà définis pour former un type plus complexe, c’est-à-dire décider de quels morceaux sera composée une variable de ce type enregistrement.

Type Enregistrement

Un *type Enregistrement* est un type, il correspond à une classe en java (par exemple **`Personne`** dans l’extrait de source suivant). Le nom de ce type est le nom de la classe. On pourra donc *déclarer* des variables de ce type de la même façon qu’on déclare une variable de type **`int`**. Une telle variable comporte plusieurs informations qui correspondent aux *champs* définis dans le *type Enregistrement* ici la *classe* qui décrit son type (modèle). Ces champs sont accessibles de tout programme en *préfixant* le nom du *champ* par le nom d’une *variable* de ce *type enregistrement* (on dira plus tard un objet ou une instance).

3. Définition et utilisation d’un type enregistrement

Créer les fichiers **`Mystere.java`** et **`ProgrammeMystere.java`**, et y placer le code ci-dessous.

```
public class Mystere {

  // toute variable de type Mystere contiendra un champ chaine et un champ entier
  String chaine ;
  int entier ;

  // les champs pourraient aussi bien représenter des nom et age des
  // identificateurs de champs plus explicites seraient alors utiles

}
```

```
public class ProgrammeMystere {


  public static void main(String arguments[]) {
        // création d'un nouvel enregistrement
        Mystere var;    /* ❶ */
        var = new Mystere();  /* ❷ */
        var.chaine = "des machines";
        var.entier = -16;
        // affichage du contenu de ses champs
        System.out.println("var.chaine = "+var.chaine);
        System.out.println("var.entier = "+var.entier);
  }
}
```

❶ la variable **`var`** est une variable de type **`Mystere`**

❷ **`new Mystere()`** crée en mémoire (rôle du new) une valeur de type **`Mystere`** et retourne un nouvel *objet*. Mystere() est l’appel d’un constructeur sans paramètre du type Mystere. Le rôle du constructeur est d’initialiser (ou construire) tout nouvel objet crée.

Compiler, exécuter et observer.

4. Définition et création de personnes

Créer une classe **`Personne`** qui contient la déclaration d’un type enregistrement **`Personne`**. On voudra garder les informations suivantes concernant une **`Personne`** :

- son prénom
- son année de naissance
- son âge actuel.

Créer également une classe **`ProgrammePersonne`**, dont le programme principal **`main(·)`** contient une variable de type **`Personne`** nommée Thelma, née en 1995.

Dans un deuxième temps, obtenez l’année actuelle, puis calculez et stockerez l'âge de cette personne.

note

Étudiez le programme suivant pour récupérer l’année actuelle.

```
import java.util.Calendar;
public class Year{
    public static void main(String [] a){

System.out.println(Calendar.getInstance().get(Calendar.YEAR));
    }
}
```

5. Les constructeurs

En **Java**, il est possible de définir des *sous-programmes* particuliers appelés *constructeurs* permettant d’initialiser les objets (variables) d’un type Enregistrement.

Un constructeur ressemble à un sous-programme et a le *même nom* que la *classe* qui le contient. Comme un sous-programme, il peut avoir des paramètres formels ou non. Tout *sous-programme* ayant le même nom que la classe qui la contient et qui n’a pas de valeur de retour déclarée est un constructeur.

Exemple :

```
//Ceci est un constructeur sans paramètre de la classe Mystere.
Mystere(){

  this.chaine  = ....;
  this.entier = ... ;

}
```

- Pas de **new** sans constructeur
- Un constructeur peut être vu comme un sous-programme mais ce n’en est pas un puisqu’il ne sert QUE dans un **new** !!
- **new** permet de créer un objet (variable) en mémoire
- Le constructeur permet d’initialiser l’objet créé, c’est-à-dire de donner une valeur à ces différents champs

5.1. Dans la classe Mystere

Voici 4 constructeurs permettant d’initialiser de 4 façons différentes un enregistrement de type **`Mystere`** :

```
  Mystere(){ /*❶*/
        this.chaine = "Nouveau" ;  // this.chaine représente le champ chaine du nouvel objet
        this.entier = 0 ;
  }

  Mystere(String pfValeurInitialeChaine){ /*❷*/
        this.chaine = pfValeurInitialeChaine ;
        this.entier = 1 ;
  }

  Mystere(int pfValeurInitialeEntier){ /*❸*/
        this.chaine = "Bla bla" ;
        this.entier = pfValeurInitialeEntier ;
  }

  Mystere(String pfValeurInitialeChaine, int pfValeurInitialeEntier){ /*❹*/
        this.chaine = pfValeurInitialeChaine ;
        this.entier = pfValeurInitialeEntier ;
  }
```

❶ ce constructeur n’a pas de paramètre et initialise les champs aux valeurs **`"Nouveau"`** et **`0`**

❷ ce constructeur initialise les champs aux valeurs **`pfValeurInitialeChaine`** et **`1`**

❸ ce constructeur initialise les champs aux valeurs **`"Bla bla"`** et **`pfValeurInitialeEntier`**

❹ ce constructeur initialise les champs aux valeurs **`pfValeurInitialeChaine`** et **`pfValeurInitialeEntier`**

Ajouter les 4 constructeurs précédents à la classe **Mystere**.

Compiler puis exécuter.

Ajouter le code suivant à la fin du **`main(·)`** de la classe **ProgrammeMystere** :

```
        // création d'un nouvel enregistrement
        Mystere var1;
        var1 = new Mystere(); //   nouvelle variable var1 prendra ici la valeur nommée this dans le constructeur

        // affiche la valeur des champs de var1
        // à vous

        // création d'un nouvel enregistrement
        Mystere var2;
        var2 = new Mystere("Claire");

        // affiche la valeur des champs de var2
        // à vous

        // création d'un nouvel enregistrement
        Mystere var3
        var3 = new Mystere(2);

        // affiche la valeur des champs de var3
        // à vous

        // création d'un nouvel enregistrement
        Mystere var4 ;
        var4 = new Mystere("Laure",36);

        // affiche la valeur des champs de var4
        // à vous
```

Compiler puis exécuter **ProgrammesMystere**.

Compléter le **`main(·)`** pour afficher les champs de tous les enregistrements créés avec les constructeurs disponibles.

5.2. Dans la classe Personne

Compléter la classe **`Personne`** en y ajoutant les constructeurs de **`Personne`** que vous jugez nécessaires.

Tester dans le main en créant de nouvelles variables de type **`Personne`** à l’aide des constructeurs fraîchement créés, puis utiliser le debugger afin de visualiser ces **`Personne`**.

6. Des étudiants et des adresses

Créer une classe **`Adresse`** destinée à définir un enregistrement **`Adresse`** et une classe **`Etudiant`** définissant un enregistrement **`Etudiant`**.

6.1. Adresses

Une **`Adresse`** comporte des champs pour le numéro de rue, le nom de la rue, un code postal et une ville :

1. Décrire l’enregistrement **`Adresse`** avec deux constructeurs : un constructeur sans paramètre (qui initialise les chaînes à `""` et l’entier à `1`) et un constructeur avec quatre paramètres, un pour chaque champ à initialiser.
2. Dans une nouvelle classe **ProgrammeEtudiant** ajouter un **`main`** afin de tester vos constructeurs.
3. Compiler puis exécuter.
4. Dans la classe **ProgrammeEtudiant**, ajouter une fonction permettant d’afficher une adresse.
5. Modifier le main afin d’appeler ce sous-programme.
6. Compiler puis exécuter.

6.2. Etudiant·e·s

Un **`Etudiant`** est représenté par son nom (champs nom), son prénom (champs prenom), son numéro étudiant, son adresse (champs adr), sa promotion (c’est un caractère `'1'`, `'2'` ou `'L'` pour, respectivement, première année, deuxième année et licence):

1. Décrire l’enregistrement **`Etudiant`** avec deux constructeurs : un constructeur sans paramètre (qui initialise les chaînes à `""` et l’adresse avec le constructeur sans paramètre d' **`Adresse`**) et un constructeur avec cinq paramètres, un pour chaque champ à initialiser.
2. Compiler.
3. Ajouter le code suivant dans un **`main()`** de la classe **ProgrammeEtudiant** :

```
//  ProgrammeEtudiant.java
public class ProgrammeEtudiant {
  public static void main(String arguments []) {
    Etudiant tabInfo[];
    tabInfo = new Etudiant[8];//tous les inscrits en info
    Adresse adresse1 = new Adresse(13, "rue des Noyers", "31000", "Toulouse");
    Adresse adresse2 = new Adresse(12, "rue Alfred Kastler", "17000", "La Rochelle");
    Adresse adresse3 = new Adresse(1, "rue des Rossignols", "31700", "Blagnac");
    Adresse adresse4 = new Adresse(20, "place des cerisiers", "11000", "Carcasonne");
    Adresse adresse5 = new Adresse(66, "avenue Michelet", "47000", "Agen");
    Adresse adresse6 = new Adresse(13, "rue des Erables", "31700", "Cahors");
    Adresse adresse7 = new Adresse(7, "rue du Cagire", "31100", "Toulouse");
    tabInfo[0] = new Etudiant ("Peninou", "Andre", "y33", 'L', adresse1);
    tabInfo[1] = new Etudiant ("Canut", "Marie-Françoise", "y23", '1', adresse2);
    tabInfo[2] = new Etudiant ("Demay", "Laurent", "z23", '2', adresse3);
    tabInfo[3] = new Etudiant ("de Michiel", "Marianne", "l87", '1', adresse4);
    tabInfo[4] = new Etudiant ("Sotin", "Pascal", "l67", '1', adresse5);
    tabInfo[5] = new Etudiant ("Stolf", "Patricia", "v12", '1', adresse6);
    tabInfo[6] = new Etudiant ("Nonne", "Laurent", "p56", 'L', adresse7);
    int nbEtudiants = 7;

    for(int i = 0; i < nbEtudiants; i++) {
      System.out.println("Nom : "+tabInfo[i].nom + " prénom " +tabInfo[i].prenom+ " adresse ");
      afficheAdresse(tabInfo[i].adr);
    }

    for(int i = 0; i < nbEtudiants; i++) {
      System.out.println("Nom : "+tabInfo[i].nom + " prénom " +tabInfo[i].prenom+ " adresse " + tabInfo[i].adr.rue);
    }

    for(int i = 0; i < nbEtudiants; i++) {
      System.out.println("Nom : "+tabInfo[i].nom + " prénom " +tabInfo[i].prenom+ " adresse " + tabInfo[i].adr);
    }

  }
}
```

4. Compiler puis exécuter **ProgrammeEtudiant**
5. Que constatez-vous ?
6. Essayer de comprendre avec le dessin mémoire suivant :

instruction	pile			tas	instruction
	tabInfo	ref1	ref1	ref11	TabInfo[0]
TabInfo[0].adresse	adresse1	ref2		ref12	TabInfo[1]
	adresse2	ref3		ref13	TabInfo[2]
	adresse3	ref4		ref14	TabInfo[3]
	adresse4	ref5		ref15	TabInfo[4]
	adresse5	ref6		ref16	TabInfo[5]
	adresse6	ref7		ref17	TabInfo[6]
	adresse7	ref8		null	TabInfo[7]
	nbEtudiants	7	Ref2 = Adresse@3d9a8c41	(13, "rue des Noyers", "31000", "Toulouse")	
			ref3	(12, "rue Alfred Kastler", "17000", "La Rochelle")	
			ref4	(1, "rue des Rossignols", "31700", "Blagnac");	
			ref5	(20, "place des cerisiers", "11000", "Carcasonne")	
			ref6	(66, "avenue Michelet", "47000", "Agen")	
			ref7	(13, "rue des Erables", "31700", "Cahors")	
			ref8	(7, "rue du Cagire", "31100", "Toulouse")	
			ref11	("Peninou", "Andre", "y33", 'L', ref2)	
			ref12	("Canut", "Marie-Françoise", "y23", '1', ref3)	
			ref13	("Demay", "Laurent", "z23", '2', ref4)	
			ref14	("de Michiel", "Marianne", "l87", '1', ref5)	
			ref15	("Sotin", "Pascal", "l67", '1', ref6)	
			ref16	("Stolf", "Patricia", "v12", '1', ref7)	
			ref17	("Nonne", "Laurent", "p56", 'L', ref8)	

Figure 1. Dessin mémoire