Java - 1ère année

Membres de Classes

Associations de Classes

Cours 3

1 - Membres de classe

- Membre d'instance
  - lié à un objet (appartient à un this)
  - données : un exemplaire du membre par objet créé de la classe
  - méthodes : la méthode connaît « this » et manœuvre les membres d'instance
- Membre de classe
  - lié à la classe
  - données : un seul exemplaire commun à toutes les instances
  - méthodes : la méthode est appelée hors objet. Elle ne connaît pas « this ».

1 - Membres de classe (2)

- Mot clef **`static`**
- Variables de classe = variables statiques
- Initialisation
  - Les variables statiques existent dès le chargement de la classe dans la machine virtuelle.
  - Initialisation statique : se fait à ce moment là.
- Persistance
  - Toute la durée de l'application

1 - Membres de classe (3)

- Membre de classe variable

  private static int nbPoints = 0;

  public Point(…){nbPoints++; …}

- Constantes uniques
  - Combinaison de **`final`** et **`static`** sur des données
  - Exemple : **`Math.PI`**
- Fonction de classe
  - Exemple : **`Math.abs(x)`**

2 – Initialisation membre de classe

– Initialisation statique de membre :

```
public static final double PI = 3.141592653589;
```

– Bloc statique :

```
public static final double PI;
static {
   double sommeEuler = 0.0;
   double termeEuler = 1.0;
   for (double i = 1.0; i<10000000.0; i++) {
      termeEuler *= i/(2.0*i+1.0);
      sommeEuler += termeEuler;
   }
   PI = 2*sommeEuler;
}
```

déconseillé

2 – Initialisation membre de classe

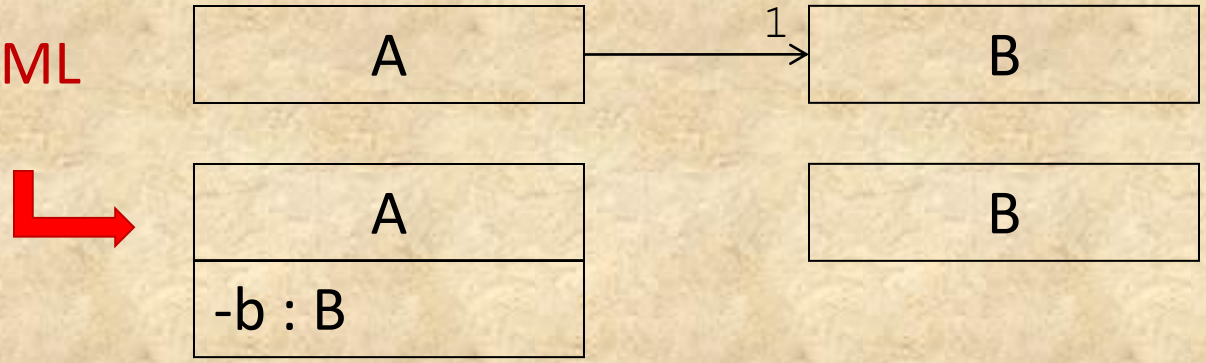
```
public static final double PI = calculePI();

private static double calculePI(){
   double sommeEuler = 0.0;
   double termeEuler = 1.0;
   for (double i = 1.0; i<10000000.0; i++) {
      termeEuler *= i/(2.0*i+1.0);
      sommeEuler += termeEuler;
   }
   return 2*sommeEuler;
}
```

better, isnt’t it ?

3 – Associations de classes

UML



Java

```
public class A {
    private B b;
}

public class B {
}
```

3 – Associations de classes (2)

UML

Java

```
public class A {
    private B role;
}

public class B {
}
```

3 – Associations de classes (3)

- association bi-directionnelle-

UML

Java

```
public class A {
    private B b;
}

public class B {
    private A a;
}
```

3 – Associations de classes (4) - initialisation (1) -

UML

Java

```
public class A {
    private B b;
    public A(B b) {
        this.b = b;
    }
    public B getB() {
        return b;
    }
}
```

```
public class A {
    private B b;
    public A() {
        b = new B();
    }
    public B getB() {
        return b;
    }
}
```

```
public class A {
    private B b;
    public void setB(B b) {
        this.b = b;
    }
    public B getB() {
        return b;
    }
}
```

3 – Associations de classes (5)

- initialisation bidirectionnelle (1) -

UML

Java

```
main() {
    A a;
    B b;
    a = new A(b);
    b = new B(a);
}
```

```
main() {
    A a;
    B b;
    a = new A();
    b = new B();
    a.setB(b);
    b.setA(a);
}
```

```
main() {
    A a;
    B b;
    a = new A();
    b = new B(a);
    a.setB(b);
}
```

3 – Associations de classes (6)

- Agrégation :

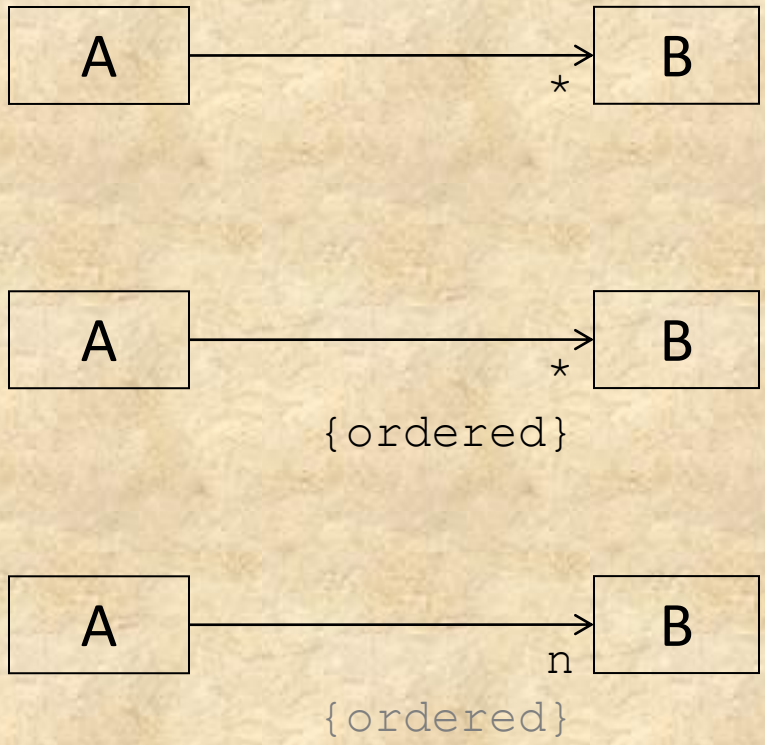
=> traduction Java identique à une association simple
(technique ≠ sémantique)

- Composition :

⇒ éviter les partages de références avec le monde extérieur
(ou procéder par copies)

3 – Associations de classes (7)
- cardinalité multiple -

UML



Java

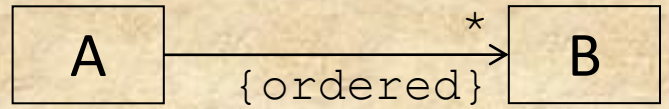
```
public class A {
    private Set<B> b;
}
```

```
public class A {
    private List<B> b;
}
```

```
public class A {
    private B[] b;
}
```

Les collections du package java.util feront l'objet d'un cours ultérieur.

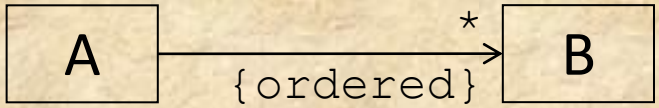
3 – Associations de classes (8)
- initialisation (*) -




```
public class A {
    private List<B> b;
    public A() {
        b = new LinkedList<B>(); // or Vector|ArrayList|…
    }
    public void addB(B b) {
        this.b.add(b);
    }
    public void addB(B… b) {
        this.b.addAll(Arrays.asList(b));
    }
    public void addB(Collection<B> b) {
        this.b.addAll(b);
    }
}
```

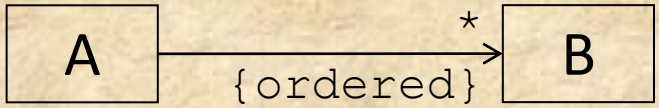
3 – Associations de classes (9)
- initialisation (*) -

```
public class A {
    private List<B> b;
    public A(List<B> b) {
        this.b = b;
    }
    public A(B… b) {
        this.b = new LinkedList(Arrays.asList(b));
    }
}
```



3 – Associations de classes (10)
- accès (*) -

```
public class A {
    private List<B> b;
    public A() {
        b = new LinkedList<B>();
    }
    public B getB(index i) {
        return b.get(i);
    }
    public List<B> getB() {
        return b;
    }
    public List<B> getB(T criteria) {
    // look for b item matching criteria
    }
}
```

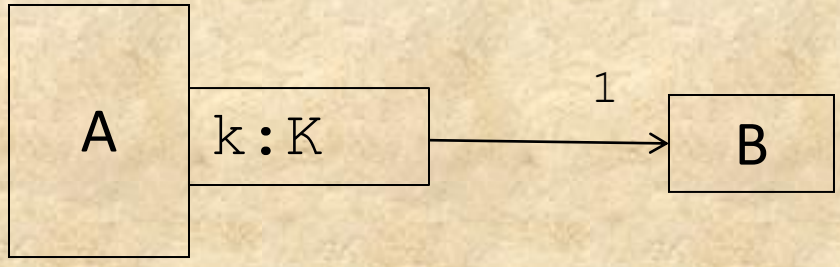



```
public static void main(String[] args) {
    A a = new A();
    a.getB().add(new B());
    a.getB().addAll(…);
}
```

!?

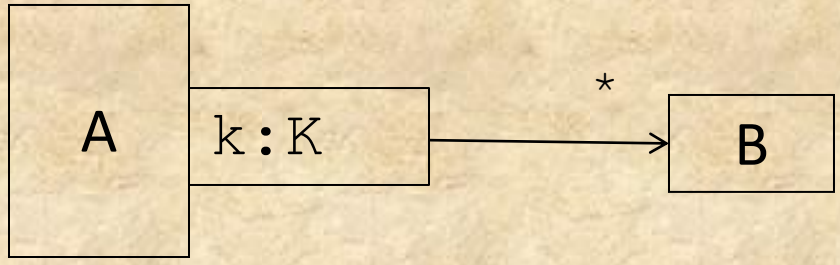
3 – Associations de classes (11)
- association qualifiée -

UML



Java

```
public class A {
    private Map<K,B> b;
}
```




```
public class A {
    private Map<K,Set<B>> b;
}
```

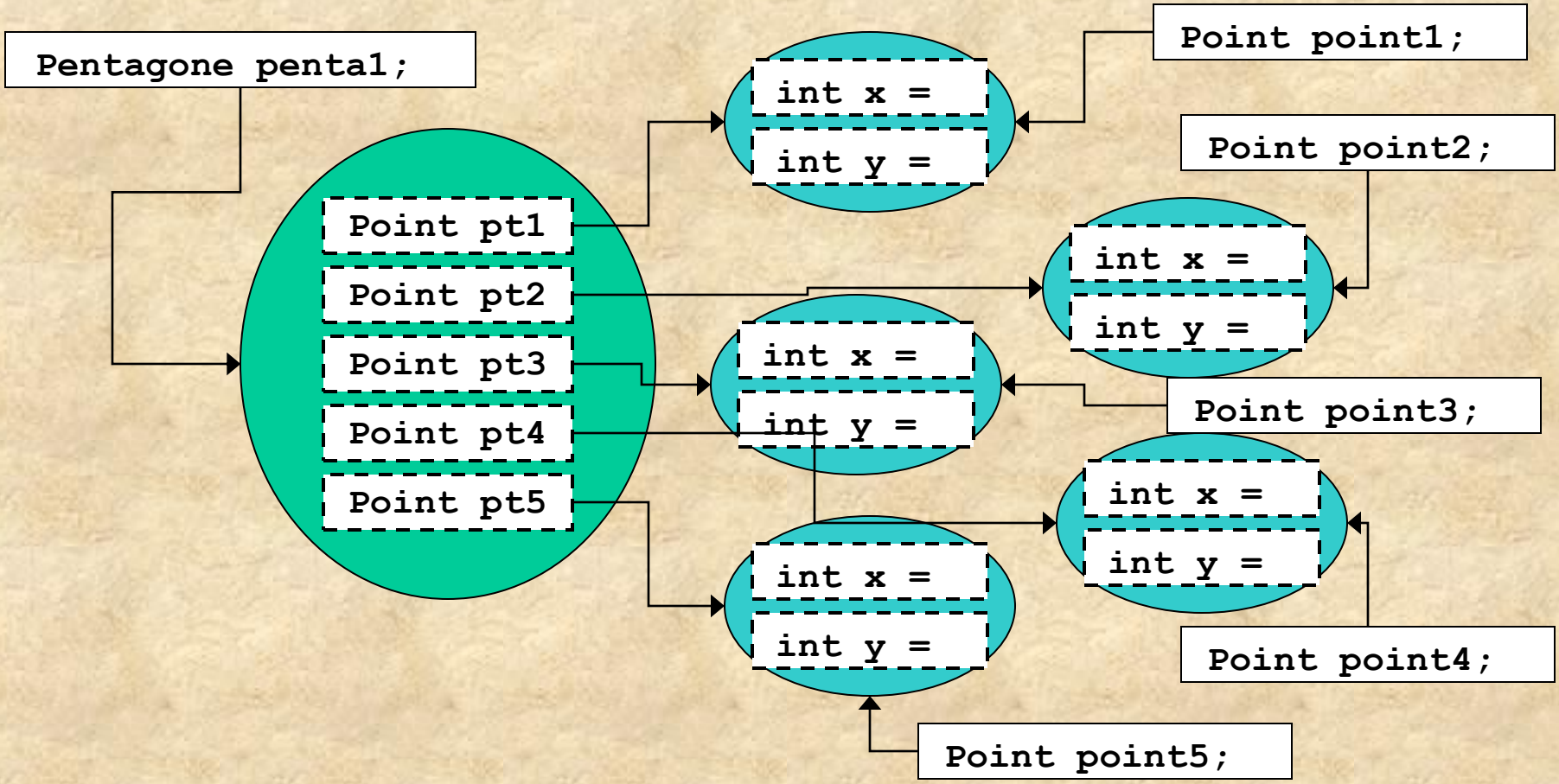
4 - Créer des objets (1)

- Créer des objets complexes
  - On crée les objets intérieurs (équipements)
  - On crée l'objet englobant

```
Point point1 = new Point(0,0);
Point point2 = new Point(2,3);
Point point3 = new Point(4,1);
Point point4 = new Point(-2,4);
Point point5 = new Point(-5,-5);
Pentagon penta1 = new Pentagon(point1, point2,
  point3, point4, point5);
```

4 - Création d'objets (2)

- Objets complexes en mémoire



FIN DU COURS