Analyse et conception

Cours de 1e année ingénieur

fabien.romeo@fromeo.fr
http://www.fromeo.fr

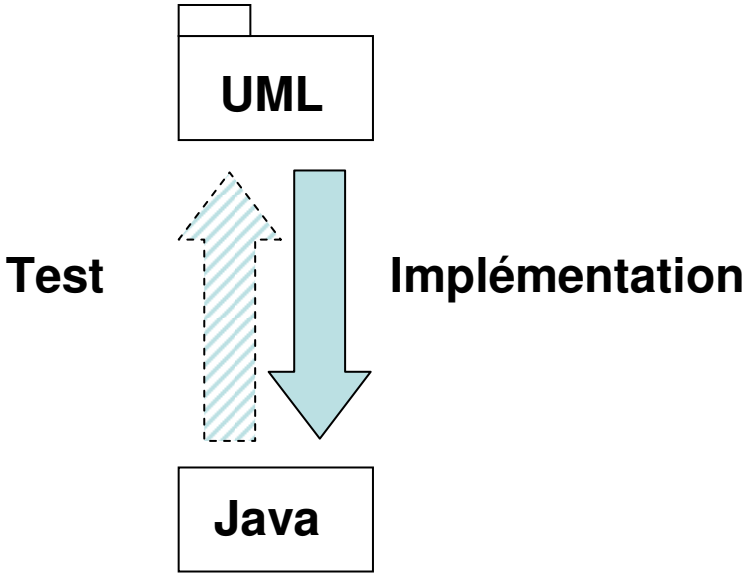
Mapping UML - JAVA

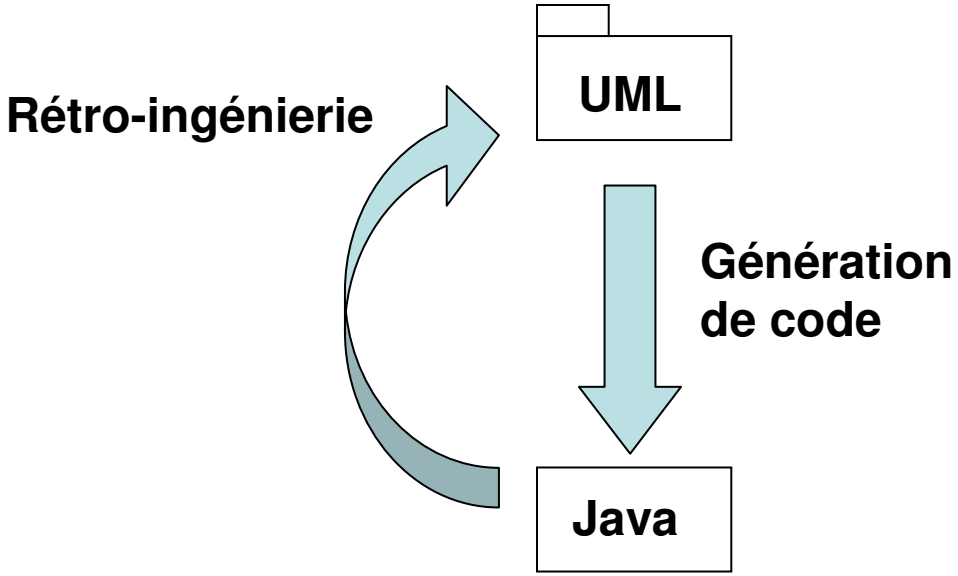
Plan

- Introduction
  - Mapping UML-Java
  - Génération de code vs. reverse engineering (outils)
- UML et Java : concepts OO
  - Classe, classe abstraite, interface, héritage, …
- Attributs et getter&setter
  - Mapping UML-Java : pas forcément uniforme dans les 2 sens
  - Setter -> Contraintes OCL
- Associations
  - Pas de concept d'association en Java
  - 1-1 ; 1-* ; qualifier ; classe d'association
- Composition et agrégation
- Diagramme de séquences
- Machine à états

Introduction

- Mapping UML – Java
  - à la main (1e année) : passage de la conception à l'implémentation
  - utilisation d'outils (2e année MDA/IDM)



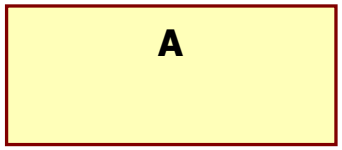


- Génération de code
  - Plusieurs choix de mapping possibles (plusieurs interprétations)
- Rétro-ingénierie
  - Le modèle retrouvé peut ne pas être identique au modèle de départ

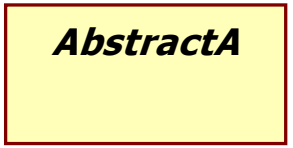
UML et Java

- UML = conception orientée objet
- Java = programmation orientée objet
- UML n'est pas lié à un langage de programmation en particulier et n'impose même pas d'utiliser un langage orienté objet
- Parce qu'ils sont orientés objets, UML et Java ont des concepts en commun qui facilitent le mapping
  - Classe, classe abstraite, interface, héritage, …
- Mais tous les concepts d'UML ne se retrouve pas forcément dans Java

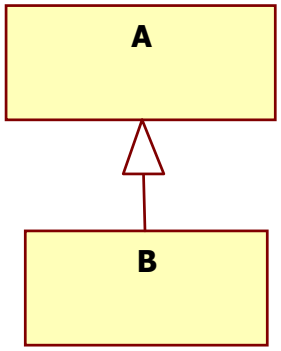
UML et Java (classes)




```
public class A {

}
```

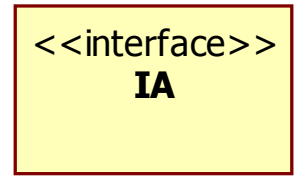



```
public abstract class AbstractA {

}
```

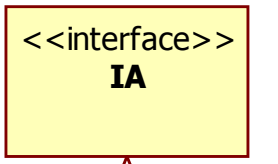



```
public class B extends A {

}
```

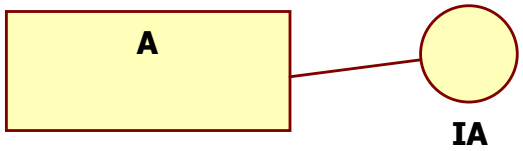
UML et Java (interfaces)



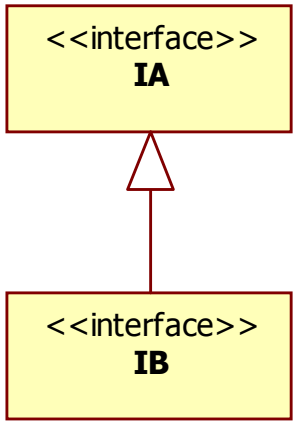

```
public interface IA {

}
```



ou

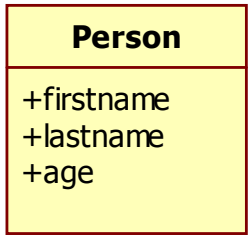



```
public class A implements IA {

}
```

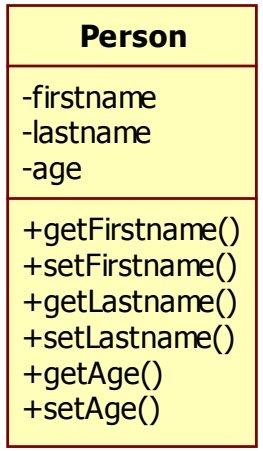



```
public interface IB extends IA {

}
```

Attributs et getter&setter

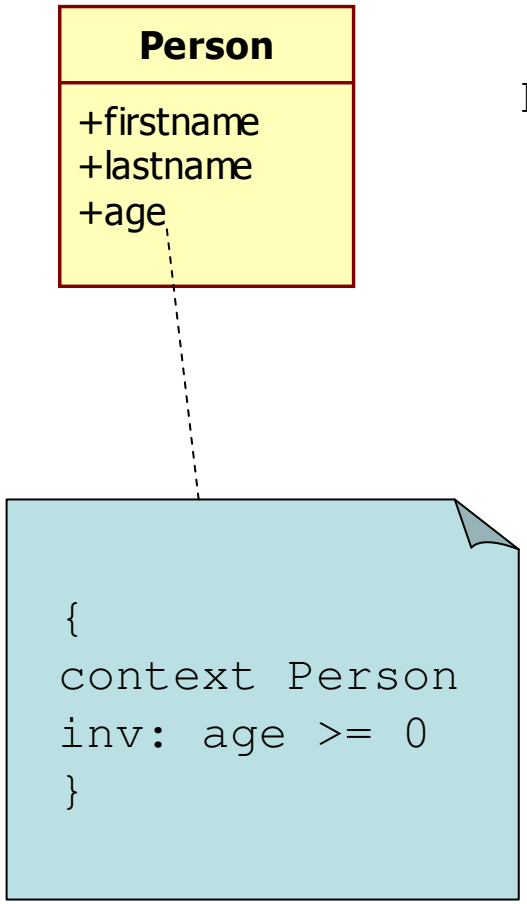



```
public class Person {
    public String firstname;
    public String lastname;
    public int age;
}
```




```
public class Person {
    private String firstname;
    private String lastname;
    private int age;

    public String getFirstname() {
        return this.firstname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    // ...
}
```

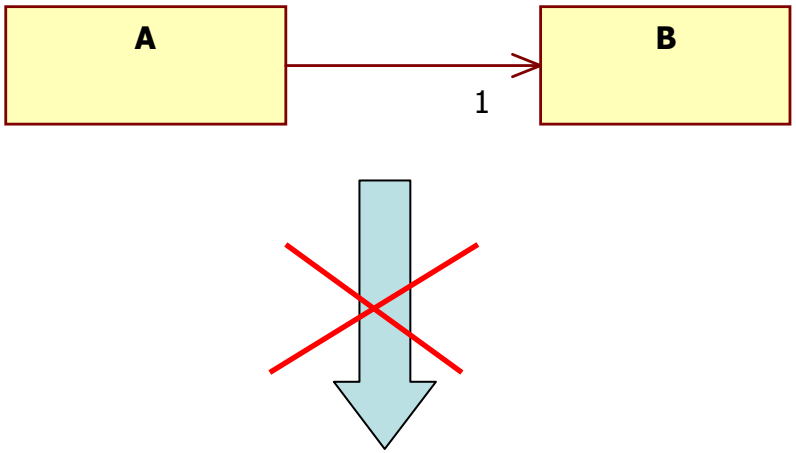
Attributs et getter&setter et OCL




```
public class Person {
    private String firstname;
    private String lastname;
    private int age;

    public int getAge() {
        return this.age;
    }
    public void setAge(int age) {
        if(age >= 0) {
            this.age = age;
        } else {
            throw new InvalidAgeException();
        }
    }
    // ...
}
```

Association

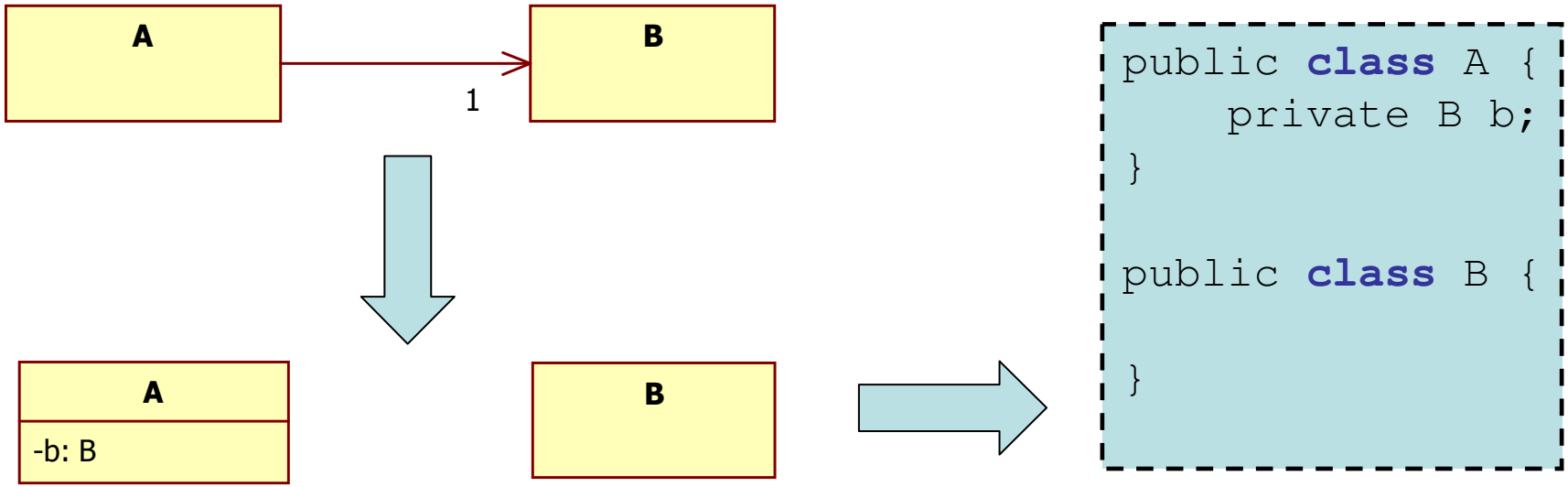
- Le concept d'association d'UML n'existe pas en Java




```
public class A association B { //???

}
```

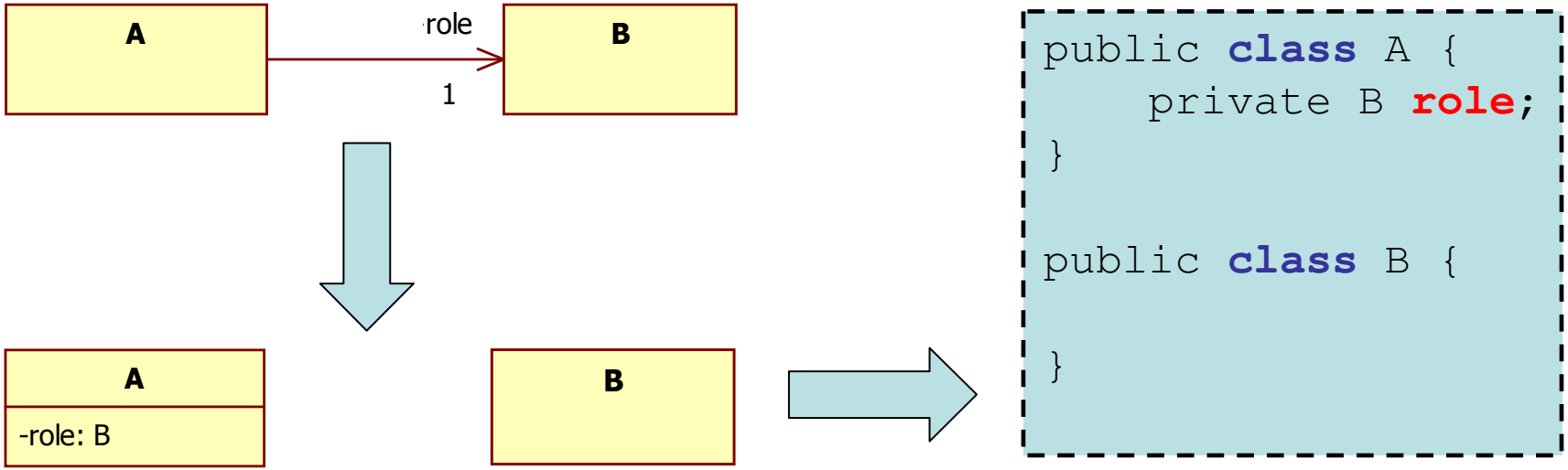
UML sans association

- Un modèle UML utilisant des associations peut se traduire en un modèle sans association
- Le mapping vers Java peut alors s'effectuer



Association avec rôle

- Même convention qu'en OCL,
  - quand il n'y a pas de rôle : nom de la classe avec première lettre en minuscule
  - quand il y a un rôle : nom du rôle



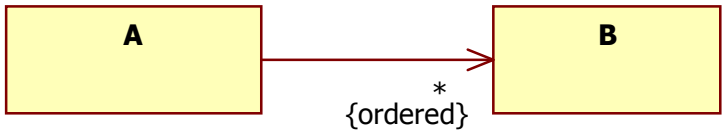
Associations et cardinalités

OCL `self.b : B`

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

OCL `self.b : Set(B)`

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



OCL `self.b : OrderedSet(B)`

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

Associations [1]

OCL `self.b : B`

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

Associations [*]

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

    public A() {
        b = new HashSet<B>();
// b = new HashSet<B>();
// b = new CopyOnWriteArraySet<B>();
// b = new … implements Set<>
    }

    public boolean add(B b) {
        return this.b.add(b);
    }
    public boolean addAll(B... b) {
        return this.b.addAll(Arrays.asList(b));
    }
    public boolean addAll(Collection<B> b) {
        return this.b.addAll(b);
    }
}
```

OCL `self.b : Set(B)`

Associations [*]

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

    public A(Set<B> b) {
        this.b = b;
    }

    public A(B... b) {
        this.b = new HashSet<B>(Arrays.asList(b));
    }

}
```

OCL `self.b : Set(B)`

Associations [*]

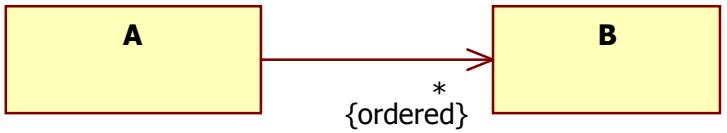
```
public class A {
    private Set<B> b;

    public A() {
        b = new HashSet<B>();
// b = new HashSet<B>();
// b = new CopyOnWriteArraySet<B>();
// b = new … implements Set<>
    }

    public Set<B> getB() {
        return this.b;
    }
}
```

OCL `self.b : Set(B)`

```
main() {
  A a = new A();
  a.getB().add(new B());
  a.getB().addAll(...);
}
```

Associations [*] {ordered}

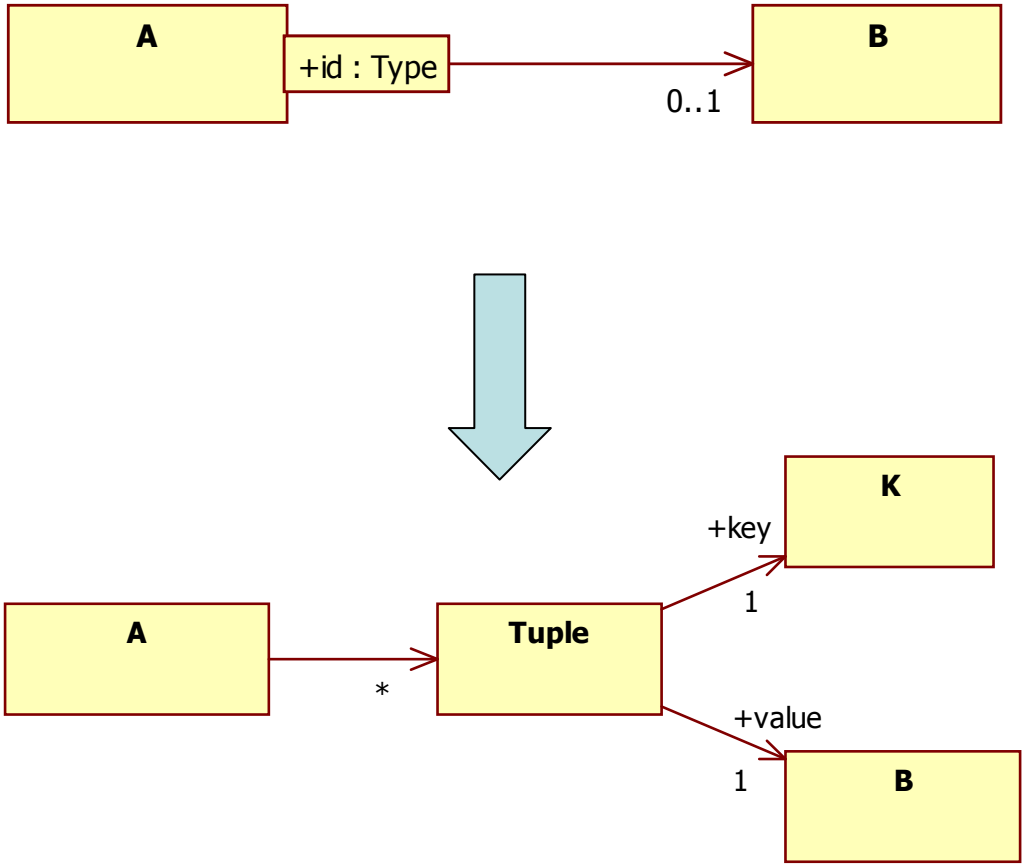
```
public class A {
    private List<B> b;

    public A() {
        b = new ArrayList<B>();
// b = new LinkedList<B>();
// b = new CopyOnWriteArrayList<B>();
// b = new … implements List<>
    }

    public List<B> getB() {
        return this.b;
    }
}
```



OCL `self.b : OrderedSet(B)`

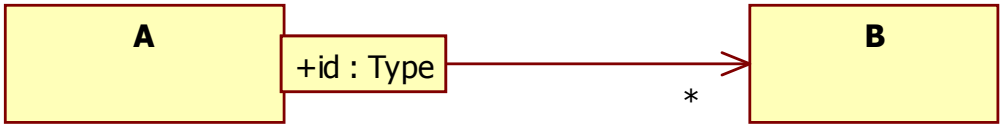
```
main() {
  A a = new A();
  a.getB().add(new B());
  a.getB().addAll(...);
}
```

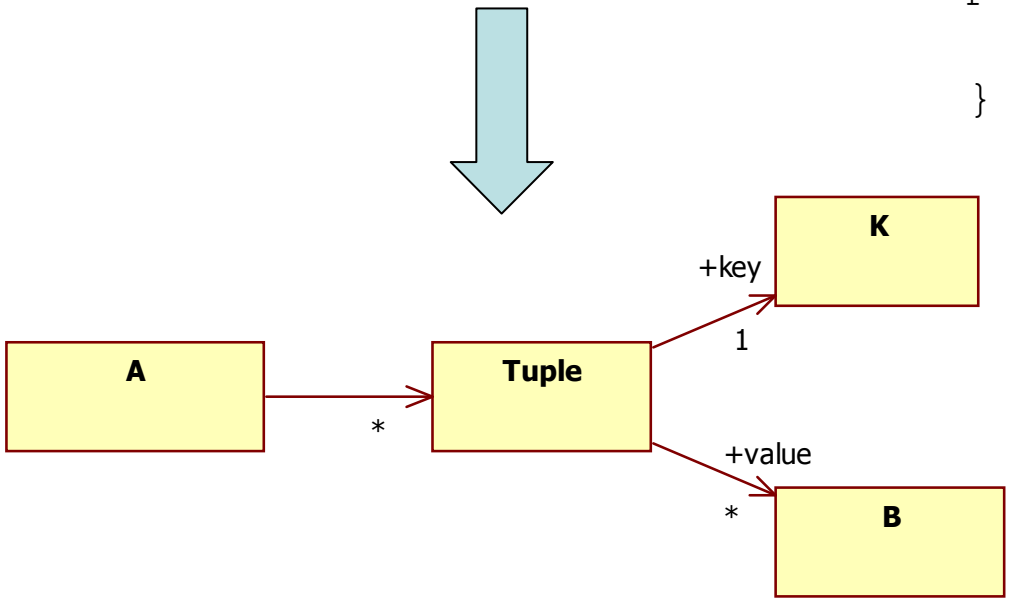
Associations qualifiées [0..1]




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

Associations qualifiées [*]






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

Association, agrégation, composition

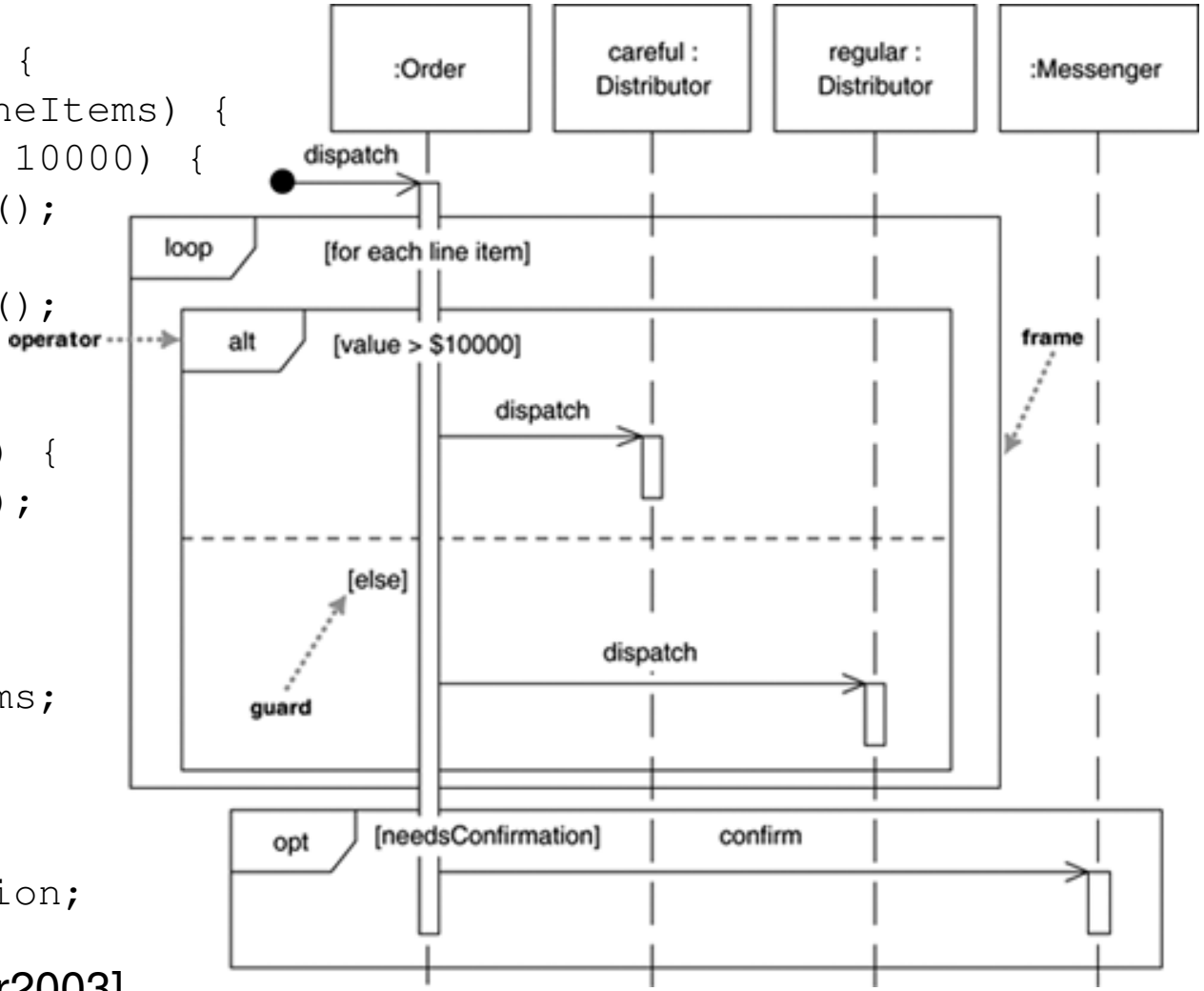
- ToDo en TD

Classes d’association

- ToDo en TD

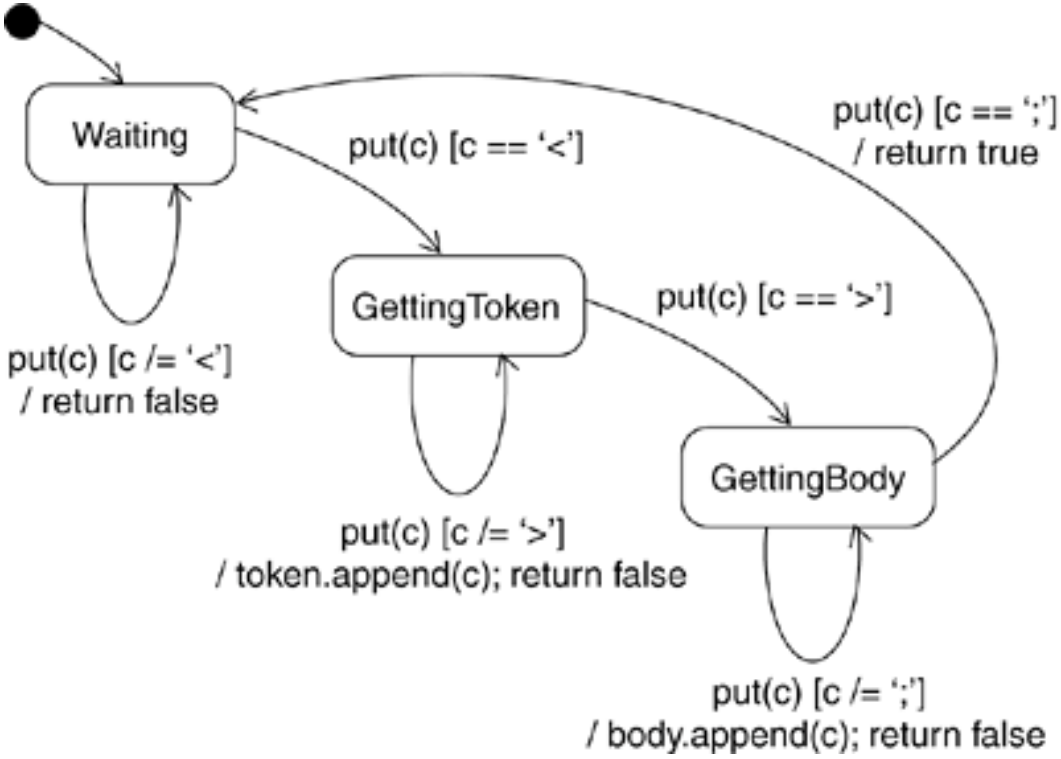
Mapping Sequence Diagram - Java

```
public class Order {

  public void dispatch() {
    for(LineItem li : lineItems) {
      if(product.value > 10000) {
        careful.dispatch();
      } else {
        regular.dispatch();
      }
    }
    if(needsConfirmation) {
      messenger.confirm();
    }
  }

  List<LineItem> lineItems;
  Distributor careful;
  Distributor regular;
  Messenger messenger;
  boolean needsConfirmation;
}
```



[Fowler2003]

Mapping State Machines - Java



[BRJ2005]

```
public class MessageParser {

 public boolean put(char c) {
  switch (state) {
   case Waiting:
    if (c == '<') {
     state = GettingToken;
     token = new StringBuffer();
     body = new StringBuffer();
    }
    break;
   case GettingToken :
    if (c == '>') state = GettingBody;
    else token.append(c);
    break;
   case GettingBody :
    if (c == ';') state = Waiting;
    else body.append(c);
    return true;
  }
  return false;
 }
 public StringBuffer getToken() { return token; }
 public StringBuffer getBody() {  return body; }

 private final static int Waiting = 0;
 private final static int GettingToken = 1;
 private final static int GettingBody = 2;
 private int state = Waiting;
 private StringBuffer token, body;
}
```