

Interfaces Graphiques

Jean-Baptiste.Yunes@univ-paris-diderot.fr

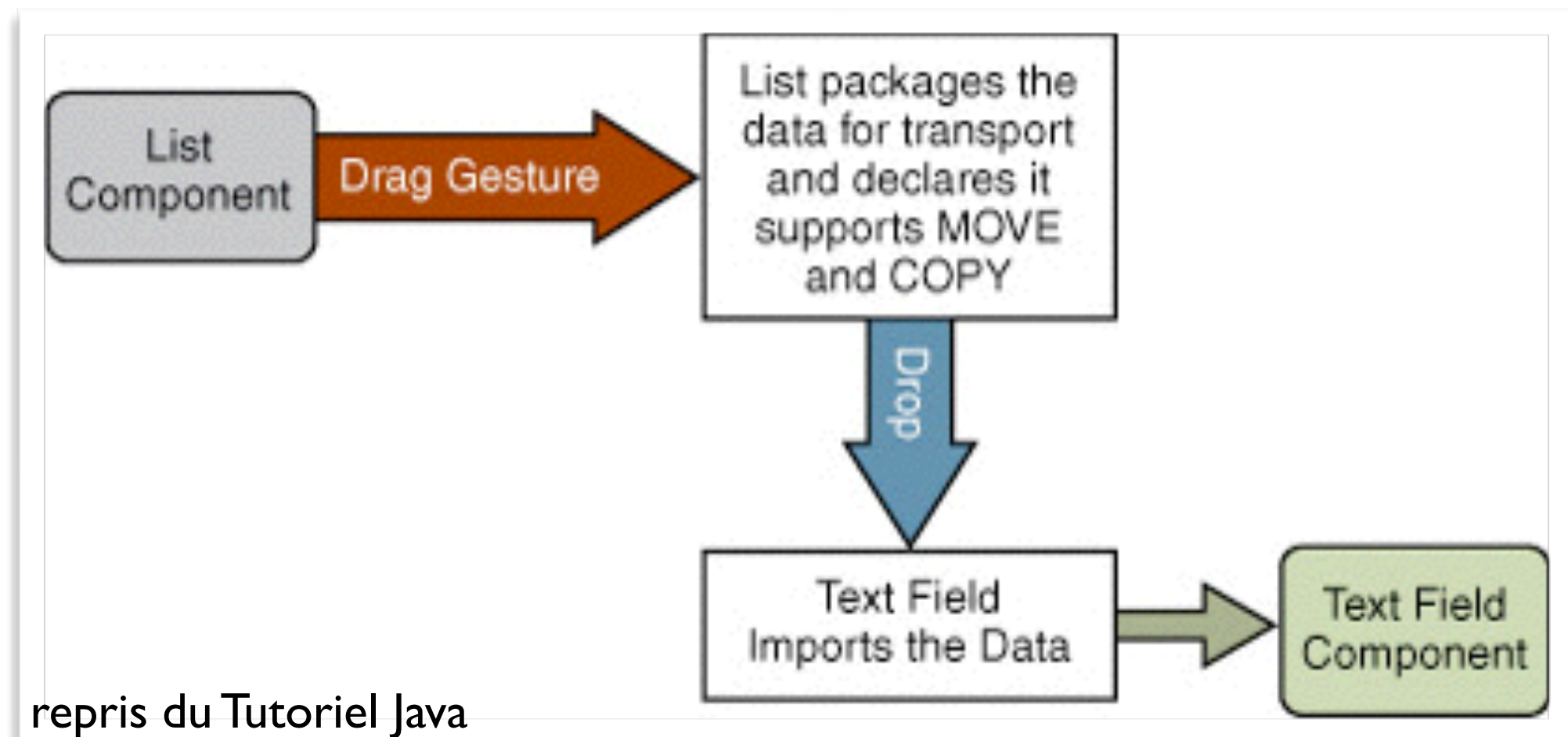
Université Paris Diderot

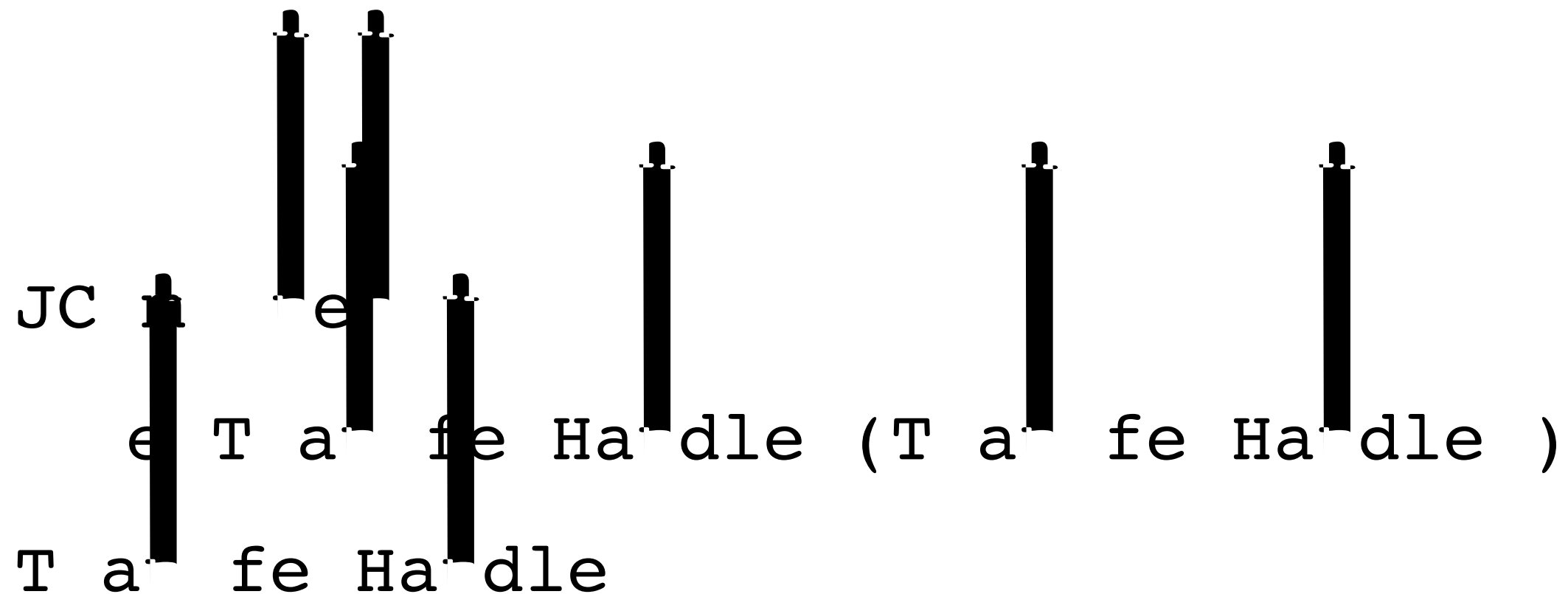
©2014

Glisser-Déposer *aka drag'n'drop*

- deux formes
 - le Glisser-Déposer
 - transfert direct par action à la souris
 - le Presse-Papier
 - transfert indirect par couper/copier - coller

- source
 - ButtonPress + Move (Drag Gesture)
 - préparation de l'export des données
- destination
 - *polling* (de la part du mécanisme de DnD)
 - import des données





- une classe encapsulant les primitives de gestion de transfert de données entre une source et une destination dans une opération de Glisser-Déposer

- Intéressons-nous au côté dépôt, deux choses importantes :
- recevoir les événements de *polling* et décider si l'on est prêt à recevoir les données
- recevoir les données si l'opération est réellement effectuée

- pour recevoir des données il suffit de

```
b lea ca Im (T a fe Handle .T a fe S )
```

- méthode régulièrement appelée pendant un Glisser-Déposer et qui doit renvoyer vrai si à cet instant le Déposer peut-être accepté

```
b lea im Da a(T a fe Handle .T a fe S )
```

- méthode appelée lorsque le Déposer est tenté

- la classe `TransferSupport` a une méthode `TransferSupport`
- encapsule les informations nécessaires afin de déterminer la faisabilité d'une opération de transfert

```
b lea i D ( )
```

- est-ce une opération de Déposer ?

```
b lea i Da aFla S ed(Da aFla )
```

- est-ce que ces données peuvent-être récupérées selon le format ?

Transferable < Transferable()

- permet de récupérer l'instance d'un Transferable (interface) permettant de récupérer les données selon différents formats

- Transferable (voir le détail plus loin)

Object < Transferable Data (DataFlavor)

- la class DataFlavor
- décrit une préférence de représentation pour des données transférables durant un Glisser-Déposer
 - essentiellement un type mime associé à une classe Java
- ex. : un fichier est glissé, lors du dépôt on peut vouloir, son contenu, son nom, etc.

- L'exemple DNDE am le.ja a
- permet de Glisser-Déposer la couleur d'un JC 1 Ch e sur un JPa e1 et donc d'en changer immédiatement la couleur de fond
- le Da aFla de la couleur d'un JC 1 Ch e est :

```
Da aFla .ja aJVML calObjec MimeT e +  
";cla =" +C 1 .cla .ge Name( )
```
- Attention le JC 1 Ch e doit déclarer accepter d'être l'origine d'un DnD : e D agE abled(e)

- Intéressons-nous au début d'une telle opération, deux choses importantes
- détecter l'action utilisateur démarrant l'opération de Glisser
- déclarer les données

- pour émettre des données il faut

```
imageSourceAcquire(JC, m, e)
```

- pour déclarer les actions de transfert supportées (combinaison de COPY, MOVE, LINK) depuis le composant donné

```
TransferableComponent(JC, m, e)
```

- pour déclarer créer l'objet de transfert

```
id = Data(JC, m, e, Transferable, i)
```

- pour libérer les données lorsque l'opération est terminée ou annulée

- une interface avec trois méthodes

```
Object T a f e D a a (D a a F l a )
```

- pour obtenir les données selon le format

```
D a a F l a [ ] g e D a a F l a ( )
```

- pour obtenir l'ensemble des formats supportés

```
b l e a i D a a F l a S e d (D a a F l a )
```

- pour déterminer si un format particulier est supporté

- pour initier un Glisser-Déposer il faut :

- détecter un geste utilisateur

- exporter un Glisser

e A D ag (JC m e , I E e , i)

- ou exporter un copier dans le presse-papier

e T Cli b a d (JC m e , Cli B a d , i)

- L'exemple DND2.ja a
 - permet de faire Glisser la couleur de fond d'un composant vers un autre capable de l'accepter
 - on exportera une couleur au même format que ceux des JC 1 Ch e

- le Glisser-Déposer à grain fin permet de contrôler finement les événements qui concernent la destination

`id e D Ta ge (D Ta ge)`

- la classe `D Ta ge` permet d'établir dynamiquement une relation entre

- le composant sur lequel le glissé s'effectue

- les événements du Glisser-Déposer

`D Ta ge (C m e , D Ta ge Li e e)`

- interface DropTargetListener
 - id d E e (D Target DragE e)
 - id d E i (D Target E e)
 - id d O e (D Target DropE e)
 - id d (D Target E e)
- note : d C h l e e (b l e a) pour spécifier que le dépôt s'est réalisé avec succès ou non
 - id d A c i Ch a g e d (D Target DragE e)



- L'exemple DNDFile am le.ja a
- permet de décorer le composant destinataire lorsqu'un Glisser opère par-dessus

- le Glisser-Déposer à grain fin permet de contrôler finement les événements qui concernent la source

- la détection d'un mouvement de Glisser-Déposer n'est pas évidente
- la classe `DragGestureRecognizer` permet d'obtenir la détection d'un mouvement initial de Glisser-Déposer
- cette détection appelle la méthode `dragGestureRecognized()` d'un `DragGestureRecognizer`

- un `DragGestureRecognizer` ne peut-être obtenu qu'à partir d'un `DragGestureRecognizer`
- la création d'un `DragGestureRecognizer` peut-être obtenue de plusieurs manières

```

    DragGestureRecognizer();
    DragGestureRecognizer().get_Default_DragGestureRecognizer();
    Create_Default_DragGestureRecognizer(Cm
    , i , DragGestureRecognizerLi e e )

```

```
id  
dragGestureListener = new DragGestureListener(id, dragGestureListener, dragGestureListener);
```

- appelée lorsqu'un geste de début de Glisser est détecté
- le DragGestureListener peut alors décider d'amorcer le Glisser-Déposer

```
startDrag()
```

- qui permet de paramétrer l'opération

- Content, Image, Parcelable, Targetable, DragSourceContext

- On peut vouloir obtenir un *scrolling* automatisé durant une opération de dépôt
- il suffit que le composant implémente l'interface

$$A \quad c \quad ll$$

```

id a      c ll (P i
;

```

 - appelé durant l'*autoscrolling*

$$I \quad e \quad ge \quad A \quad c \quad ll \quad I \quad e \quad (\quad) \quad ;$$
 - appelée pour déterminer les zones d'*autoscrolling*

- L'exemple DNDASE am le.ja a
- illustre l'ensemble des mécanismes de Glisser-Déposer à grain fin