

Toolkits (suite)

1. Copier-Coller
2. Glisser-déplacer (Drag'n Drop)

1. Copier-Coller

- Généralités
 - Mécanisme de communication inter-applications ou intra-application contrôlé par l'utilisateur
 - Mécanisme en 2 temps:
 1. Appli A
 - > Presse-Papier (Clipboard)
 2. Presse-Papier
 - > Appli B
- "Petits problèmes" posés par l'implémentation d'un tel mécanisme
 - Fonctionner quelque soit Appli A et Appli B
 - Minimiser les pertes d'informations dues au transfert
 - Fonctionner quelque soit le système de fenêtrage
 - tous les systèmes de fenêtrage ont des gestions du PP différentes

Cas simple: Copier-Coller de texte (Copier)

- Principe de fonctionnement
 1. Obtenir référence sur PP système
`Clipboard pp = Toolkit.getDefaultToolkit().getSystemClipboard();`
 2. Emballer le texte dans un objet Transferable
`StringSelection st = new StringSelection(texte_à_copier);`
 3. Transférer le texte vers le Presse-Papier (Copier)
`pp.setData(st, null)`

On peut passer ici un objet ClipboardOwner, propriétaire du pp. Inutile pour texte simple

Cas simple: Copier-Coller de texte (Coller)

4. Récupérer le texte du PP
`Transferable contenu = clipboard.getContents(obj)`
5. Traitement du contenu (Coller)
 - Cas du contenu vide
 - si PP est vide ou ne contient pas des données récupérables par Java
 - Cas du contenu non-vide
 - Vérifier que le contenu du PP peut être vu comme du texte
`contenu.isDataFlavorSupported(DataFlavor.StringFlavor)`
 - Récupérer le texte
`String str = (String) contenu.getTransferData(DataFlavor.StringFlavor)`

Type texte format unicode

NB: à mettre dans un bloc try...catch car getTransferData déclenche des exceptions quand le type (flavor) n'est pas supporté les données sont illisibles

Généralisation du copier-coller

- Objectifs
 - copier coller des données de tout type (tableaux, images, etc)
- Deux approches possibles

1. Tout transférer en texte

- principe du mail
- format MIME
- avantages
 - standard MIME qui facilite la communication inter-application
 - relativement léger à implémenter
- inconvénients
 - codage/décodage

2. Créer de nouveaux types de données transférables

- En utilisant le mécanisme des DataFlavors et des Transferable
- inconvénients
 - format d'objet "propriétaire" => pose problème de communication avec d'autres applications

5

Montautz Hascoët, Univ. Montpellier II

2. Glisser / déplacer

Une source de déplacement
DragSource

Une cible de lâcher
DropTarget

Des données qui transitent
Transferable

Mécanisme complexe d'émission et de gestion
d'événements

6

Montautz Hascoët, Univ. Montpellier II

Source de déplacement

- Création d'une source de déplacement
`DragSource ds = DragSource.getDefaultDragSource()`
- Création d'un gestionnaire d'évènements reconnus comme des "départs de glisser"

```
class preparePourGlisser implements DragGestureListener  
void DragGestureRecognized(DragGestureEvent event)
```

1. construire le Transferable qui sera récupéré lors du lâcher

/ c'est ici qu'il y a du travail pour emballer les données dans un transferable */*

2. lancer le suivi du déplacement

```
event.startDrag(cursor,transferable,dsl)
```

- Associer les deux

```
ds.createDefaultDragGestureRecognizer(widget,DnDConstants.ACTION_COPY_  
OR_MOVE, new PreparePourGlisser())
```

7

Montautz Hascoët, Univ. Montpellier II

Suivi du déplacement

- Pris en charge par un gestionnaire d'évènements `DragSourceListener`
- Construire son propre `DragSourceListener`
 - 5 méthodes à définir
 - Celles qui permettent de donner un feedback visuel
`DragEnter()`, `DragOver()`, `DragExit()`,
`DropActionChanged()`
 - Celle qui permet de mettre à jour les données quand le drop a été effectué
`DragDropEnd()`

8

Montautz Hascoët, Univ. Montpellier II

Cibles de déplacement

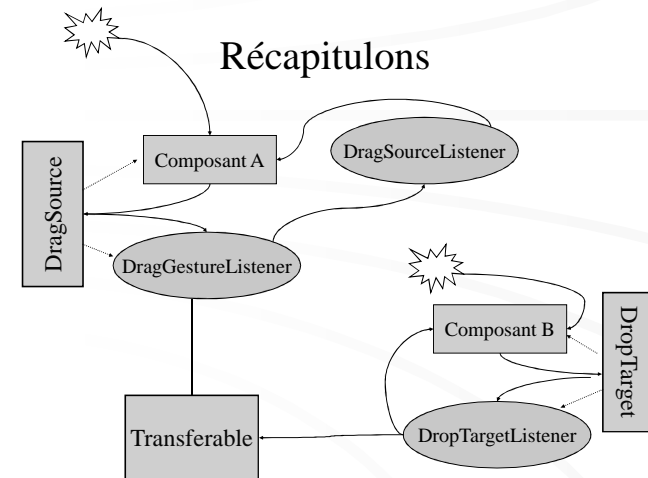
- Création d'un gestionnaire d'évènements de déplacement

class preparePourDéplacer implements DropTargetListener

// 5 méthodes à définir

- Celles qui permettent de donner un feedback visuel
dragEnter(), dragOver(), dragExit(), dropActionChanged()
- Celle qui permet de mettre à jour les données quand le drop a été effectué
drop ()
- Création d'une cible de déplacement et association avec le gestionnaire d'évènement ci-dessus
DropTarget dt = new DropTarget (composant, dtl)

Récapitulons



Remarques (1/2)

- Activer/désactiver une cible
 - dt.setActive(true/false)
- Différents types de déplacer
 - déplacer vraiment
 - déplacer (en copiant - ctrl ou shift)
 - déplacer (en créant un alias - ctrl et shift)
 - Il faut décider quels types de déplacer on souhaite accepter
 - setActionDefaults(constante)

Remarques (2/2)

- Tenir compte des particularités du glisser-déplacer
 - ☹ erreurs fréquentes
 - ☹ difficulté pour l'utilisateur à deviner :
 1. Quelles sont les objets sur lesquels le déplacer va avoir un effet
 2. Quel est l'effet du déplacer sur le-dit objet
 - ☺ rapide, "naturel", efficace
 - ☺ Conclusion:
 - ☺ A utiliser autant que possible mais savoir s'abstenir lorsque c'est nécessaire!!