

SWT

Présenté par :
Maguelonne Héritier



IC
CRIM 20 ANS

Votre **accélérateur**
technologique

Introduction

- Présentation des différentes interfaces graphiques java
 - AWT
 - Swing
 - SWT
- Particularités de SWT
 - Structure d'une application SWT
 - JFace
- SWT vs Swing
 - Avantages
 - Inconvénients
- Extensions de SWT

Interface graphique en java

AWT (Abstract Window Toolkit)

- Première API qui a été développée(jdk 1.0)
- Propose des composants basiques, simples à utiliser
- Fournit des composants (Widgets) intégrés à la plateforme.
 - Interface complexe avec les composants natifs : Widgets affectés à un homologue système (peer) sous jacent (architecture très lourde)

Interface graphique en java

AWT (Abstract Window Toolkit)

- Composants offerts limités
 - N'offre pas l'accès à des composants de niveau supérieur.
 - Utilisation limitée aux composants qui sont disponibles sur toutes les plateformes. (« plus petit dénominateur commun » ou « LCD »)

Interface graphique en java

Swing (Java Foundation Classes)

- Développé par SUN
- Composants sont non-natifs : solution pure Java
- Fonctionnalités plus avancées (pas de problème LCD)
- Mauvaise performance (avant la version JDK 1.4)
 - Accapare la JVM
- Look différents de la plateforme.

Interface graphique en java

SWT(Standard Widget Toolkit)

- Bibliothèque de classes graphiques du projet Eclipse initiée par IBM
- Utilise les meilleurs des deux approches (SWT et swing)
- Utilise les composants graphiques de l'OS, lorsqu'ils sont disponibles
 - SWT définit une API transférable commune pour toutes les plateformes supportées .
 - Implémentation de l'API sur chaque plateforme, si possible à l'aide de composants natifs.

Interface graphique en java

SWT(Standard Widget Toolkit)

- Bonnes performances, sans sacrifier la portabilité
- Garde le look du système d'exploitation sous-jacent
- Standard Java non reconnu par le JPC

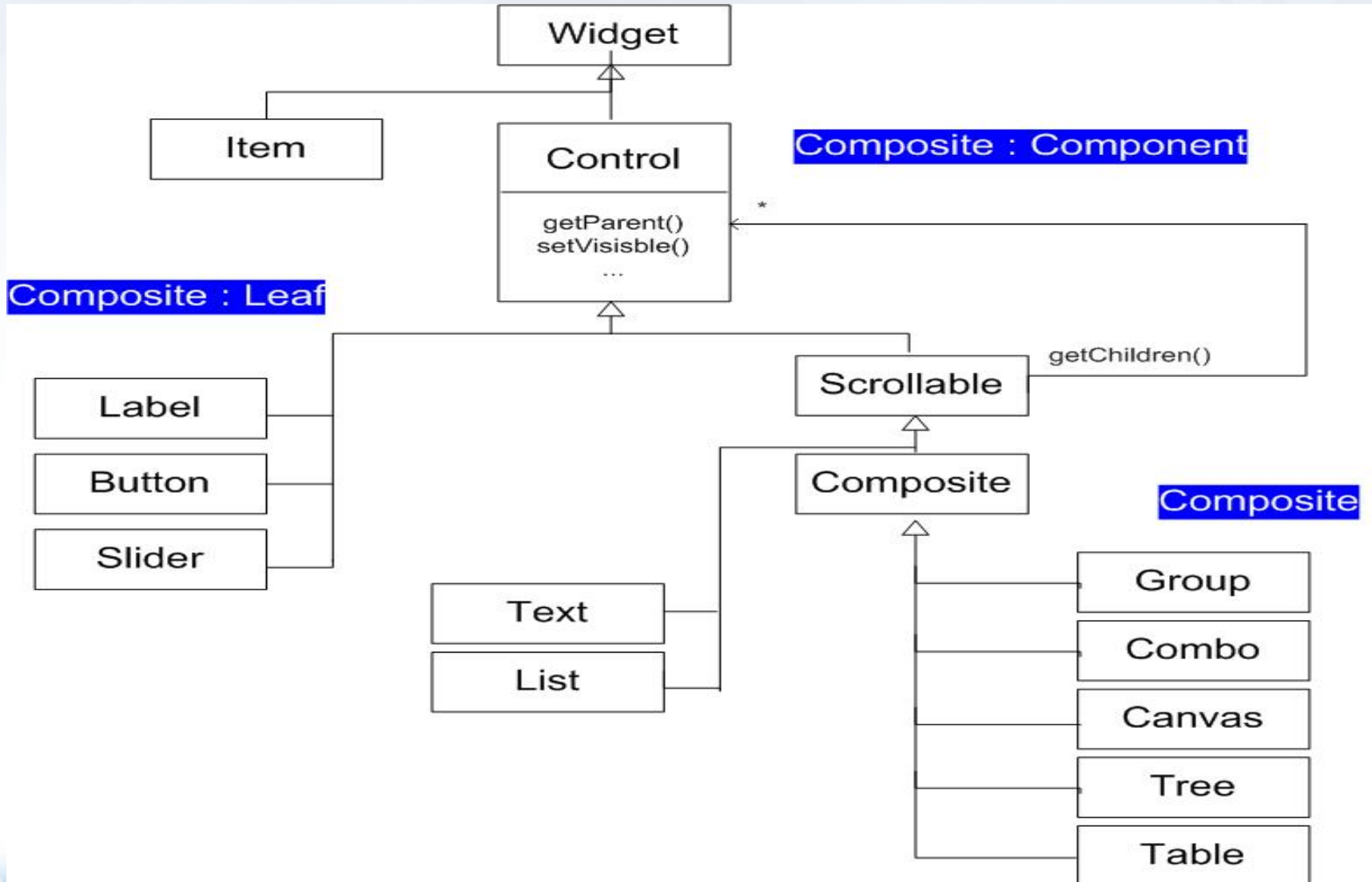
Problème du « plus petit dénominateur commun »

- Émulation des fonctions généralement utiles et qui ne sont pas disponibles sur toutes les plateformes
- Omission des fonctions peu utiles.
- Les fonctions spécifiques à une plateforme ne sont fournies que sur la plateforme appropriée

Les composants SWT

- Tous les objets UI sont dérivés de la classe Widget
- Classe Widget
 - Classe Item : objet UI léger
 - Classe Control :
 - Composants complexes, couteux
 - Contient des objets composites (Patron Composite)
 - Contient des Objets Items

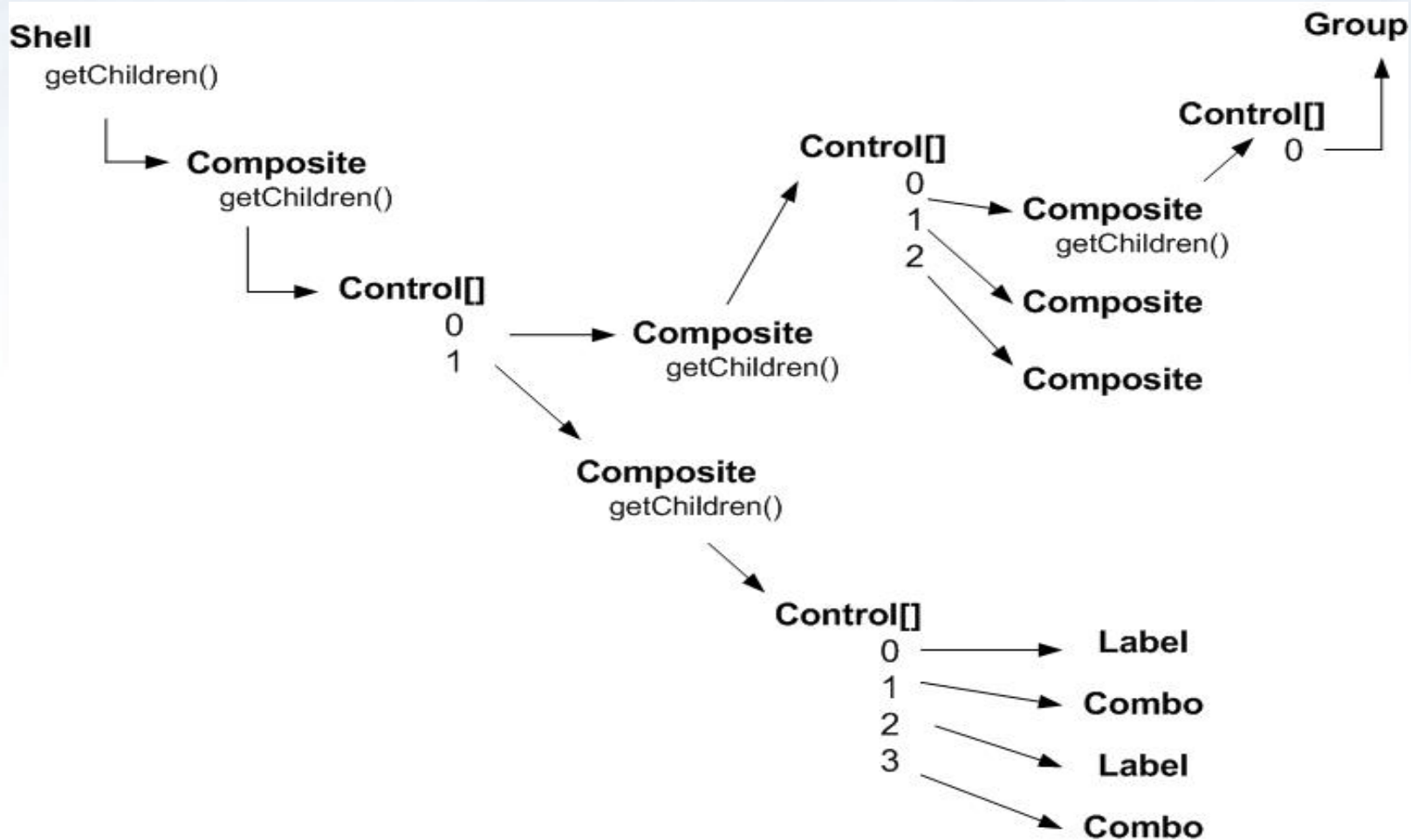
Les composants SWT



Structure d'une application SWT

- Création d'un affichage représentant une session SWT (objet *Display*).
 - représente le lien avec la plateforme sous-jacente
- Création d'un ou plusieurs shells (ils servent de fenêtre principale pour l'application)
- Création des composants requis dans le shell
- Initialisation des dimensions, des états pour les composants
- Enregistrement des écouteurs
- Ouverture de la fenêtre shell. (méthode *open*)
 - Exécution de la boucle de distribution des événements jusqu'à ce qu'une condition de sortie soit rencontrée (généralement lorsque la fenêtre shell principale est fermée par l'utilisateur).

Structure d'une application SWT



Structure d'une application SWT

- Les états sont stockés par le composant de la plateforme plutôt que par celui de SWT.
 - Patron État Variable
 - Les composants SWT sont très petits
- Les composants SWT doivent être explicitement détruit. (méthode dispose())
 - Si un parent est détruit, tout ces enfants sont détruits

Structure d'une application SWT

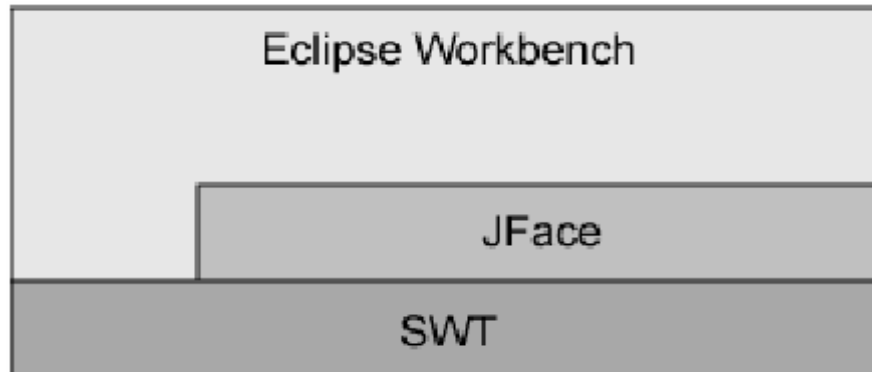
- Pas d'héritage de Widgets en SWT.
 - les objets sont des représentations du code natif d'un toolkit qui ne supporte pas forcément l'héritage (ex: gtk est codé en C)
 - Utilisation des Patron Stratégie et Commande

■ SWT

- fournit une couche légère et portable au dessus de la plateforme native de composants: fournit composants, layout, événements

■ JFace est une surcouche de SWT

- amène une couche plus objet, un niveau supérieur



- fonctionnalités supplémentaires :
 - Gestion des données dans les composants
 - Sémantique pour définitions des actions utilisateurs
 - Schémas communs pour le traitement des ressources de l'UI.
 - Structure pour la générations d'interactions complexes avec l'utilisateur

SWT vs Swing

■ Avantages de SWT

- **Natif:** s'adapte automatiquement au look and feel du système (donne un look très professionnel)
- **Petite api :** facile à prendre en main
- **Bonne Performance**
- **Faible consommation de mémoire.**
- **Bonne intégration** (notamment avec OLE, COM, ActiveX)
- **Opensource**

SWT vs Swing

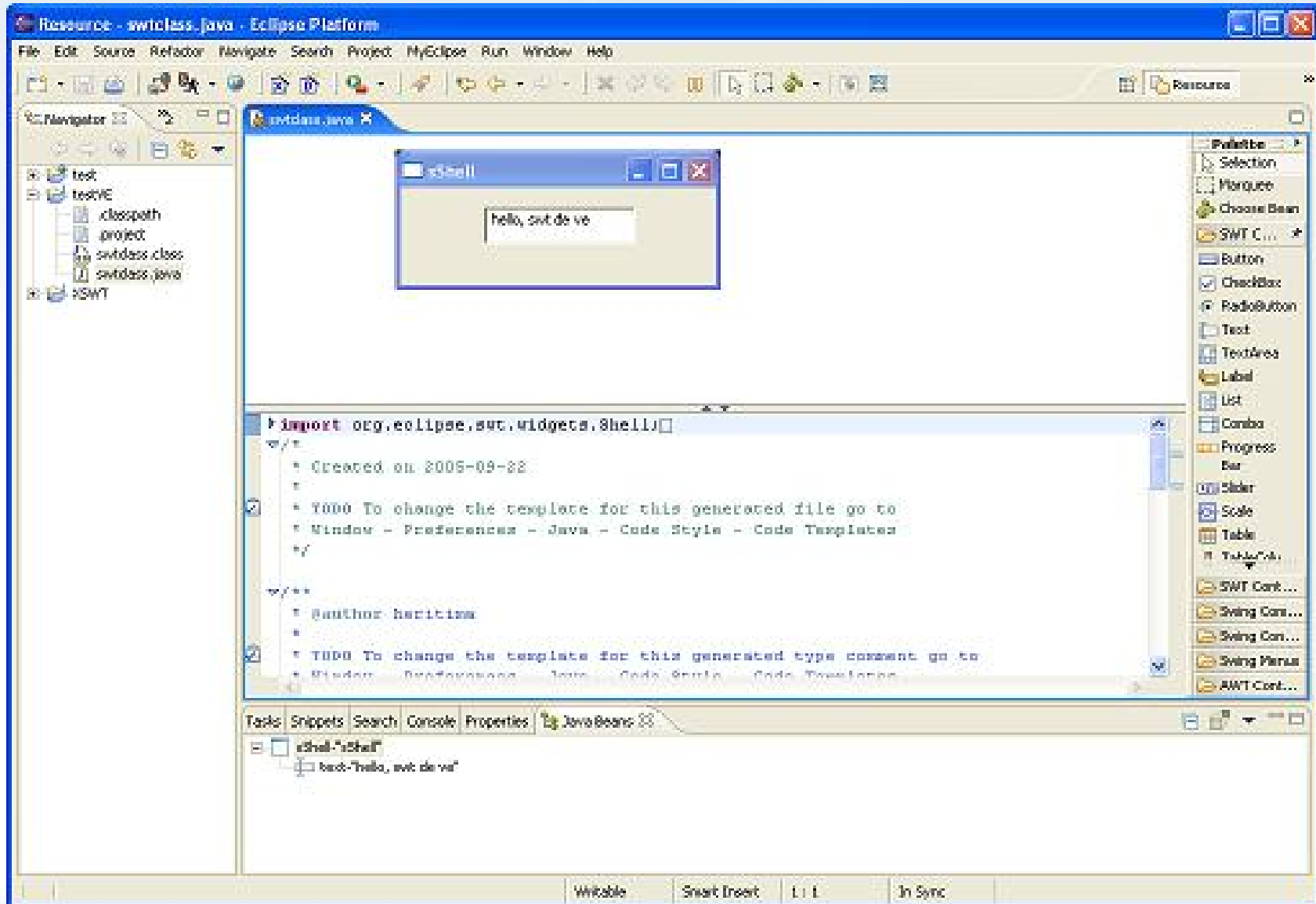
- Inconvénients de SWT
 - Non standard
 - Natif : limitation de la plateforme
 - Portabilité : limitée aux plateformes supportées
 - Gestion mémoire non automatique (il est nécessaire de faire un `.dispose()` sur tous les éléments graphiques pour libérer la mémoire du composant natif qu'il représente)
 - Moins de fonctionnalités que Swing
 - Pas d'applet possible

- Intégration de composantes Swing dans SWT
- Description des interfaces par XML : (XSWT, CookSwt)
- Développement de bibliothèques graphiques (Manipulation de formes, intégration OpenGL)

Extension

- Connexion des beans java aux interfaces graphiques SWT (support de JGoodies Data Binding framework)
- Widgets personnalisés utilisés par le PDE d'Eclipse (Plug-in Development Environment) fournis comme API public

Pugin Eclipse: Visual Editor



Conclusion

- SWT intéressant
 - Performance
 - Look
- Pas intégrer dans le JDK
- L'option Swing (à partir de la version 1.4) est toujours à envisager