Interfaces graphiques en Java

Le problème :

- Le code exécutable Java est indépendant de la plate-forme
- Mais une interface graphique avec gestion d'événements implique le système sous-jacent

« Deux-trois » solutions :

- AWT (Abstract Window-ing- Toolkit)
- Swing
- … et SWT
- … et JavaFX

AWT (créé par Sun)

- Greffer des objets Java sur les composants du système sous-jacent
- Technique dite des composants *peers*
- Avantage : le « look and feel » de l'application reprend celui du système sous-jacent
- Inconvénients : consommation de ressources système, incompatibilités subtiles, et…
- L'inconvénient de l'avantage : le « look and feel » de l'application change selon le système sous-jacent
- Gros inconvénient : uniformisation par le bas (un composant AWT ***doit*** être disponible dans ***tous*** les systèmes)

Swing (créé par Sun)

- Gestion interne de ***tous*** les composants (sauf la fenêtre de base)
- Avantage : le « look and feel » de l'application ne dépend pas du système sous-jacent
- Avantages : consommation minimale de ressources système, aucun risque d'incompatibilité
- Avantage : un composant Swing est automatiquement disponible dans ***tous*** les systèmes
- Inconvénient : Swing doit tout gérer et a des performances plus limitées qu'AWT
- Avantage secondaire : la création de *plafs* : *Personalized Look And Feel*

SWT (créé pour Eclipse)

- Croisement open-source, extérieur à Sun, de AWT et Swing en terme de philosophie

JavaFX (créé par Sun/Oracle)

- Plus fort, plus beau mais plus compliqué à mettre en œuvre.