

Attendre avec style et l'API Swing

par [Romain Guy](#)


Date de publication :

Dernière mise à jour :


Démonstration sur l'utilisation de composants Swing pour attendre lors de l'exécution de tâches longues.

- I - Démonstration Java Web Start
- II - Explications
- III - Téléchargements

I - Démonstration Java Web Start

 **Prérequis** : Pour essayer cette démonstration vous devez disposer de [Java Web Start](#) (compris avec le JRE et le JDK).

 **Lancez la démonstration**

 **Attention** : Seuls les fichiers `AnimatedPanel.java` et `InfiniteProgressPanel.java` sont librement utilisables. [Contactez moi](#) pour les autres sources.

II - Explications

Les UI (*User Interface*) peuvent être très agaçantes, tout spécialement quand elles semblent faire quelque chose, mais ne vous disent pas quoi. Pire, certaines ne vous disent même pas combien de temps leur prendra à réaliser leur tâche courante. En tant qu'utilisateur j'ai vécu ça trop souvent. En tant que développeur, je sais combien il est dur de déterminer la durée d'une tâche.

C'est pourquoi, certaines applications, comme l'installateur Mozilla, utilisent une barre de progression infinie. D'habitude, ce genre de composants ressemblent aux barres de progression normales, mais fonctionnent un peu différemment. Dans le cas de Mozilla, vous pouvez voir un rectangle rebondir entre les bords de la barre. J'ai aussi vu des barres de progression infinies agissant comme des barres normales, mais revenant en arrière une fois remplies et recommencent. Même si la barre de progression infinie est une bonne idée, ces implémentations "craignent". L'utilisateur n'a pas moyen de comprendre à première vue que la barre de progression est "infinie".

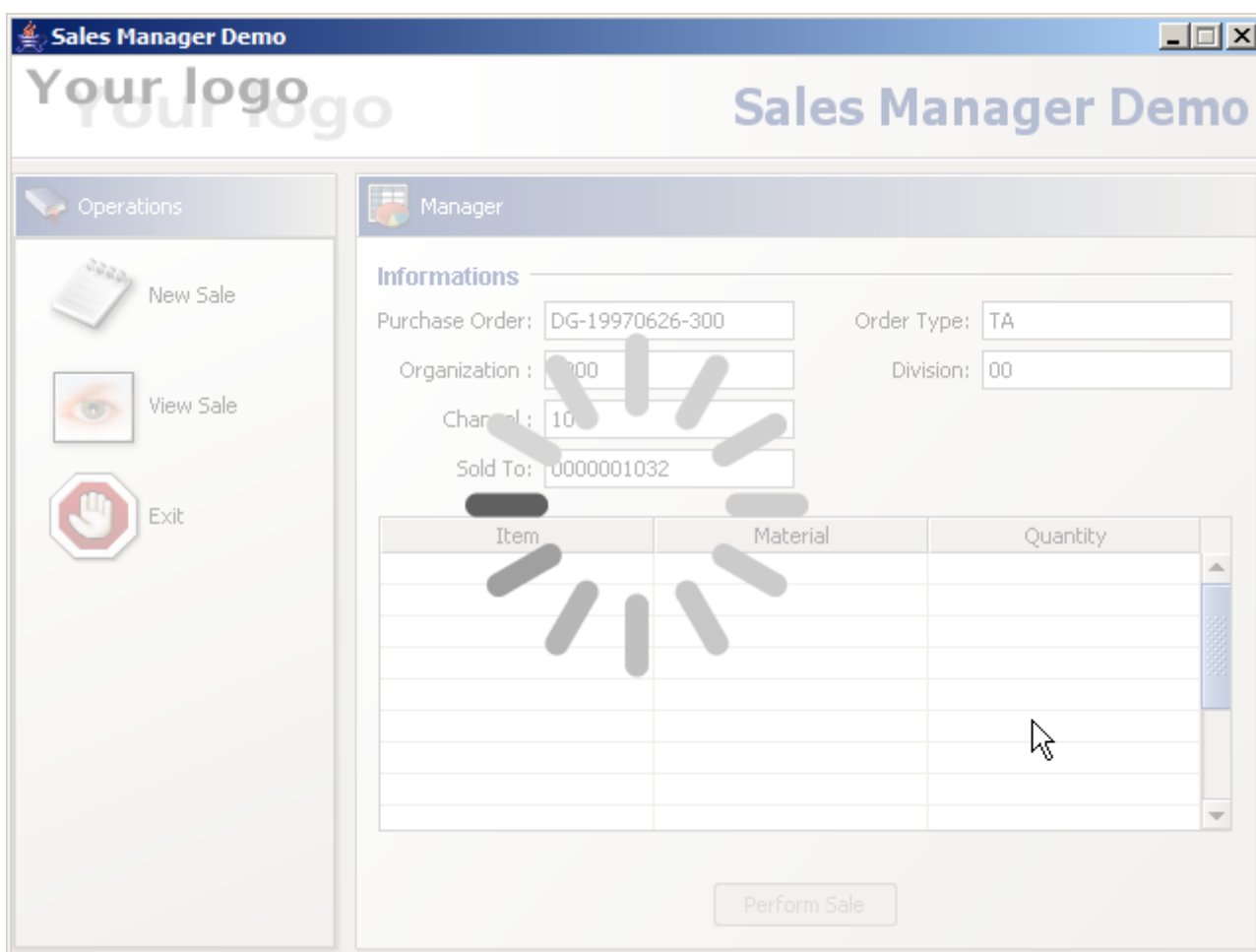
Heureusement, il existe un moyen simple de passer outre cette limitation. J'ai construits deux exemples pour vous montrer quels types de *composants de progression infinie* peuvent être construits avec Swing. Le premier est montre simplement une image et un label briller.



Je vous suggère vivement de lancer la [demo WebStart](#) pour le voir en action. Une fois l'application lancée, cliquez sur *View Sale* puis sur le bouton *Search*. Le code source de cet effet se trouve dans [org.progx.salesmanagers.ui.AnimatedPanel](#). Ce panel est fait d'un message et d'une image. Un thread d'animation change la transparence du message et la brillance de l'image. Le code source est plutôt facile à comprendre.

Cette première solution est jolie, mais pas aussi puissante qu'elle le pourrait. Dans cet exemple, l'UI reste active durant l'animation. Un programme complet devrait faire attention à bien invalider les composants "dangereux". Ma seconde solution utilise le *glass pane* de la fenêtre. Comme vous devez le savoir, le *glass pane* est un composant situé au dessus de tous les autres contenants de la fenêtre, tout spécialement le *content pane*. En interceptant tous les événements de la souris dans le *glass pane* vous pouvez bloquer toute l'application (les événements clavier doivent aussi être gérés pour éviter l'utilisation des accélérateurs).

J'ai aussi choisi une autre façon de représenter la barre de progression infinie. Inspiré de MacOS X, *org.progx.salesmanagers.ui.InfiniteProgressPanel* dessine un ensemble circulaire de barres en train de tourner. Et, devinez quoi ? Le cercle est la forme parfaite pour décrire le caractère infini du composant de progression. Pour finir, j'ai ajouté un voile pour montrer que l'UI ne peut plus être manipulée. Voilà à quoi cela ressemble :



Vous devriez vraiment essayer la démo online pour ce composant :) Une fois l'application lancée, cliquez sur "New Sale" puis sur le bouton "Perform Sale". Vous noterez l'estompement cyclique du panel. Les constructeurs par défaut vous permettent de choisir la transparence du voile et la durée de l'animation. Le code utilise quelques opérations de Java2D, mais reste facile à comprendre.

Une dernière remarque : *InfiniteProgressPanel* a été écrit très rapidement et le code est un peu brouillon, tout particulièrement le thread d'animation (qui aurait dû être splitté en trois threads). Le délais de sommeil du thread d'animation a tendance à consommer beaucoup de puissance processeur sur les machines pas très récentes (P4 1.5 Ghz). Cela demande à coup sur quelques optimisations :)

III - Téléchargements

[Version PDF \(mirroir HTTP\)](#)

[Version HTML/ZIP \(mirroir HTTP\)](#)

[Code source \(mirroir HTTP\)](#)

Attention : Seuls les fichiers *AnimatedPanel.java* et *InfiniteProgressPanel.java* sont librement utilisables. [Contactez moi](#) pour les autres sources.