Nicolas Gonin
Karim Luy
Matthieu Bandelier

Design Patterns – Equipe I - Labo n°1 : Bridge pattern

Introduction

L'exemple que nous devons résoudre présente un problème où l'implémentation de différentes formes ce fait à l'aide de deux API. Pour chaque forme on a donc deux classes concrètes à implémenter. C'est ce qu'on appelle un couplage fort entre l'API et toutes les classes qui représentent des formes. Ça signifie que si l'on souhaite changer l'API, il faudra le faire dans toutes les classes qui représentent des formes, ce qui donne, avec une implémentation basique, un nombre de classes très conséquent.

Implémentation

Sans Bridge

Notre implémentation basique de la donnée, c'est-à-dire sans Design Pattern ni aucune réflexion orientée objet, est représenté sur le diagramme UML ci-dessous (Figure 1).

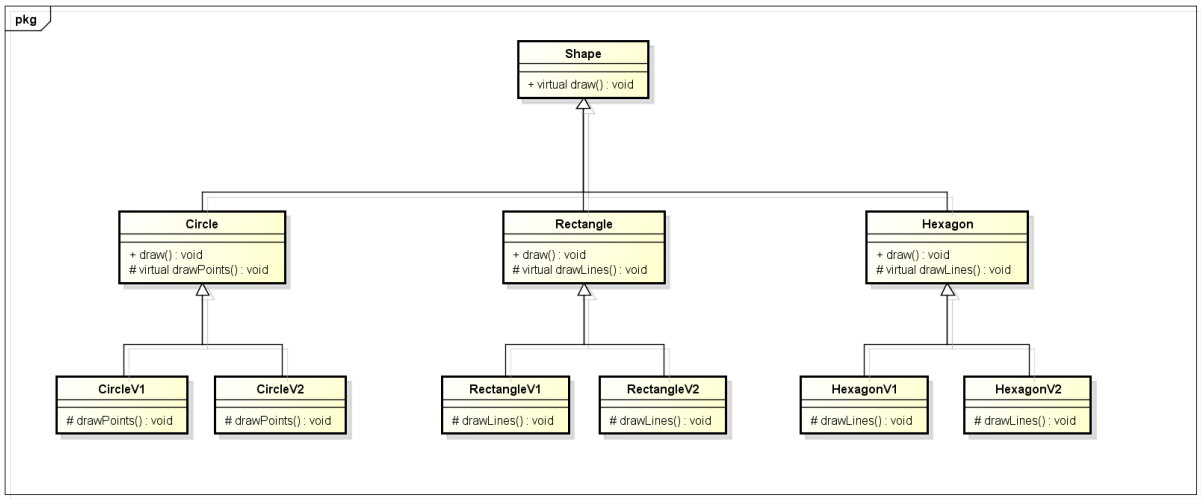

Figure 1

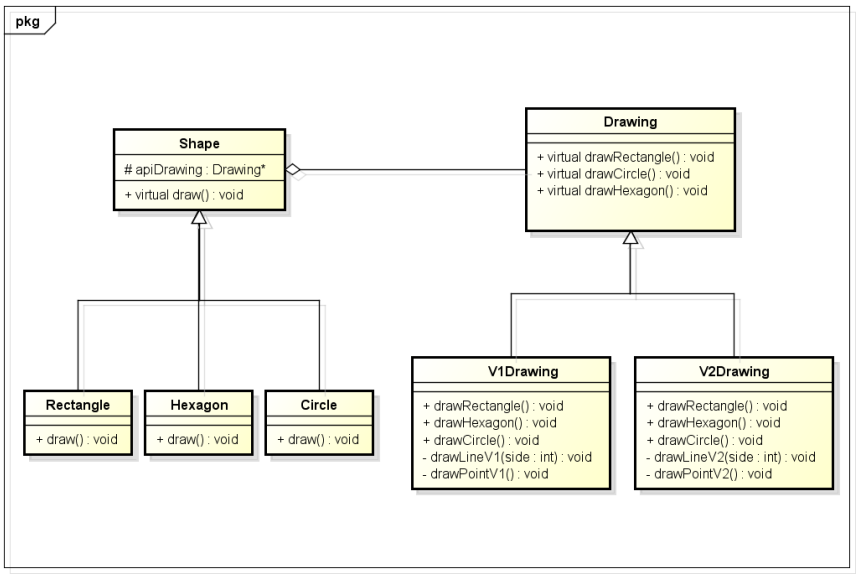

Figure 2

Nicolas Gonin
Karim Luy
Matthieu Bandelier

Avec Bridge

Notre implémentation avec le Design Pattern "Bridge" est représentée sur le diagramme UML ci-dessous (Figure 2).

Comparatif

	Bridge	Basique
Nombre de classes (sans main)	2 + nbFormes + nbAPI	1 + nbFormes + nbFormes *nbAPI
En fixant nbAPI à 2	nbFormes + 4	3* nbFormes + 1
En fixant nbAPI à 4	nbFormes + 6	5* nbFormes + 1

2 API

Quand on a beaucoup de "Formes" (Rectangle, Elipse, Cercle, etc...), on tend vers 3 fois plus de classes en codant bêtement comparé au Design Pattern 'Bridge'.

Exemple avec 30 "Formes" et 2 API

Bridge : 34 classes Basique: 91 classes

4 API

Quand on a beaucoup de " Formes " (Rectangle, Elipse, Cercle, etc...), on tend vers 5 fois plus de classes en codant bêtement comparé au Design Pattern 'Bridge'.

Exemple avec 30 "Formes" et 4 API

Bridge : 36 classes Basique: 151 classes

Conclusion

« Le ***pont*** est un patron de conception de la famille « Structuration », qui permet de découpler l'interface d'une classe et son implémentation. » citation tirée de Wikipédia.

Le pont ("Bridge" en anglais) résous ce problème en découplant l'API et les classes dérivées de formes les rendant ainsi plus indépendantes.

Dans le comparatif, nous constatons aisément que le nombre de classes avec Bridge est significativement inférieur au nombre de classes d'une implémentation basique de la donnée.

Cette différence est d'autant plus flagrante si l'on a un grand nombre de "Formes" et un grand nombre d'API.

Le Design Pattern "Bridge" nécessite donc un nombre de classes nettement inférieur à l'implémentation basique, pour les mêmes fonctionnalités, et toujours de manière "Orienté Objet"