2 min – Modélisation objet 2

Cf. 2 minutes MOB1

Design pattern

Type	Nom	Description
Structure	Composite	Manipuler un groupe d'objet de la même façon que s'il s'agissait d'un seul objet
	Adapter	Permet de convertir l'interface d'une classe en une autre interface
	Bridge	Découple l'interface de l'implémentation -> modifier ou changer l'implémentation d'une classe sans devoir modifier le code client
	Décorateur	Permet d'attacher dynamiquement de nouvelles responsabilités à un objet.
	Façade	Cache une conception et une interface complexe difficile à comprendre
	Flyweight	
	Proxy	Se substitue à une autre classe pour ajouter une indirection à l'utilisation de la classe à substituer
Création	Abstract Factory	Encapsule un groupe de fabrique (endroit où l'on construit des objets) ayant une thématique connue
	Builder	Séparer la construction d'un objet complexe de la représentation afin que le même processus de construction puisse créer différentes représentations
	Factory method	Instancie des objets divers dont le type n'est pas prédéfini (ils seront crées dynamiquement en fonction des paramètres passés à la fabrique)
	Prototype	Copie d'une instance puis modification. Comporte une classe abstraite avec une méthode virtuelle pure clone()
	Singleton	Restreindre l'instanciation d'une classe à un seul objet.
Comportement	Chain of responsability	Permet à un nombre quelconque de classe d'essayer de répondre à une requête dans connaitre les possibilités des autres classes sur cette requête -> permet de séparer les différentes étapes d'un traitement
	Command	Encapsule la notion d'invocation. Séparer complétement le code initiateur de l'action du code de l'action elle-même.
	Mediator	Fournit une interface unifié d'interface d'un sous-système ->réduit les dépendances entre plusieurs classes
	Memento	Permet de restaurer un état précédent d'un objet dans violer le principe d'encapsulation. Deux objets : créateur (possède un état interne : état à sauvegarder) et gardien (qui permis de retrouver un étant antérieur)
	Observer	Envoie un signal à des modules qui jouent le rôle d'observateur -> action adéquate effectuée en fonction des informations qui parviennent depuis les modules qu'ils observent
	State	Permet de changer le comportement d'un objet sans pour autant en changer l'instante
	Strategy	Permet de selectionner des algo à la vole au cours du temps selon certaines conditions
	Template method	Définit un squelette d'un algo à l'aide d'opérations

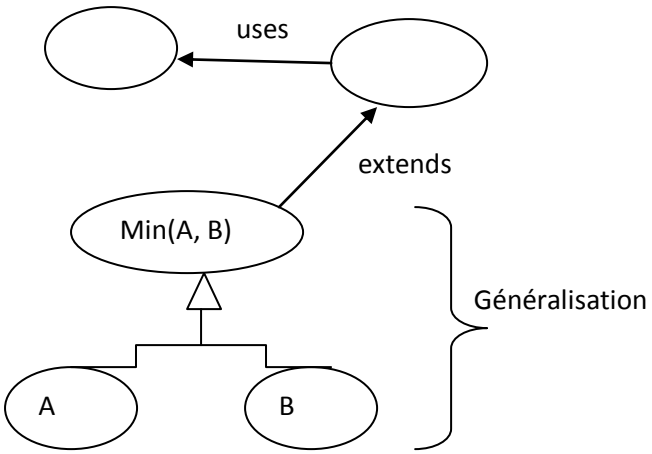
	abstraites dont le comportement concret se trouvera dans les sous-classes, qui implémenteront ces opérations
Visitor	Manière de séparer un algo d'une structure de données
Callback	Fonction passée en argument à une autre function -> permet l'usage de cet callback comme de n'importe quelle function, alors qu'elle ne la connaît pas par avance
Interpreter	Définit la grammaire d'un langage qui décrit des opérations, et utilise cette grammaire pour interpréter des états du langage
Iterator	Permet de parcourir tous les éléments contenus dans un autre objet

Définitions

3 axes : dynamique (à l'exécution), statique (à la compilation), fonctionnel (au sujet du logiciel, diagramme de cas d'utilisation)

Système : module fonctionnel, notion de sens, il existe différents grains cohérents et indépendants des autres modules

Acteur : utilisateur quelconque, extérieur au système, qui coopère avec le cas d'utilisation



Uses : dépendance forte – extends : bonus, extra

D'abord effectuer la partie avec la pointe de la flèche