LE PATTERN ITERATEUR

Intention

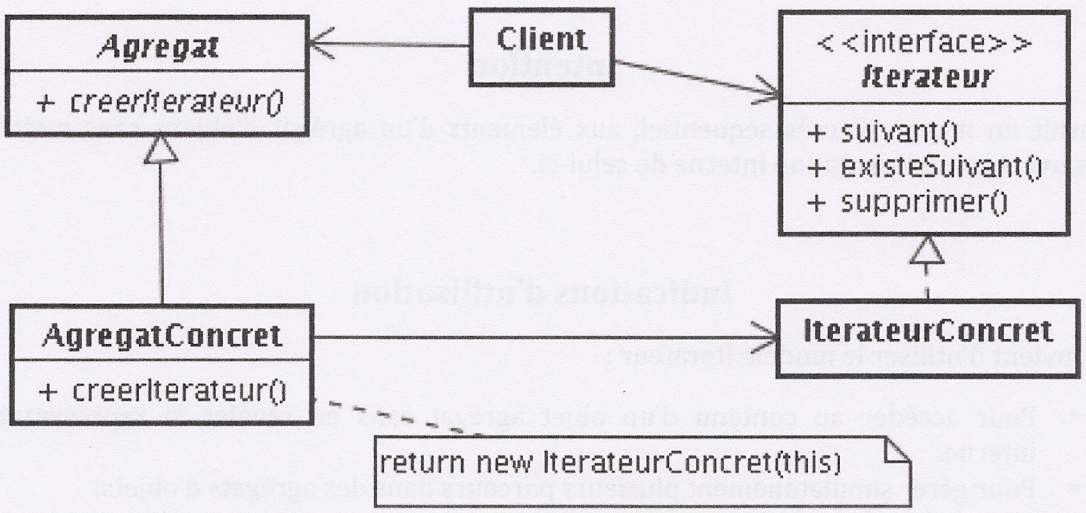
Fournit un moyen d'accès séquentiel, aux éléments d'un agrégat d'objets, sans mettre à découvert la représentation interne de celui-ci.

Indications d'utilisation

Il convient d'utiliser le modèle Iterateur :

- Pour accéder au contenu d'un objet agrégat sans en révéler la représentation interne.
- Pour gérer simulatanément plusieurs parcours dans des agrégats d'objets.
- Pour offrir une interface uniforme pour les parcours au travers de diverses structures agrégats (c'est-à-dire, pour permettre l'itération polymorphe).

Structure



Constituants

Iterateur

- Définit une interface pour accéder aux éléments et les parcourir.

IterateurConcret

- Implémente l'interface Iterateur.
- Assure le suivi de l'élément courant lors de la traversée d'un agrégat.

Agregat

- Définit une interface pour la création d'un objet Iterateur.

AgregatConcret

- Implémente l'interface de création de Iterateur, afin de retourner l'instance adéquate de IterateurConcret.

Collaborations

Un IterateurConcret assure le suivi de l'objet courant de l'agregat, et peut déterminer l'objet suivant dans le parcours.

```
=====================================================
                      PLAT
=====================================================

public class Plat {
    private String nom;
    private String description;
    private double prix;

    public Plat(String nom, String description, double prix) {
        super();
        this.nom = nom;
        this.description = description;
        this.prix = prix;
    }

    public String getNom() {
        return nom;
    }


    public String getDescription() {
        return description;
    }


    public double getPrix() {
        return prix;
    }

}

=====================================================
                   MENU CREPERIE
=====================================================

public class MenuCreperie {
    private ArrayList<Plat> plats;

    public MenuCreperie(){
        plats = new ArrayList<Plat>();

        ajouterPlat(new Plat("Crêpe à l'oeuf", "Crêpe avec oeuf au plat ou
brouillé", 2.99));
        ajouterPlat(new Plat("Crêpe complète", "Crêpe avec oeuf au plat et
```



```
jambon", 2.99));
        ajouterPlat(new Plat("Crêpe forestière", "Myrtilles fraîches et sirop
de myrtilles", 4.49));
    }

    public void ajouterPlat(Plat plat){
        plats.add(plat);
    }

    public ArrayList<Plat> getPlats(){
        return plats;
    }
}

=====================================================
                   MENU CAFET
=====================================================

public class MenuCafet {
    private static final int MAX_PLATS = 6;
    private int nbPlats = 0;
    Plat[] plats;

    public MenuCafet(){
        plats = new Plat[MAX_PLATS];

        ajouterPlat(new Plat("Salade printanière", "Salade verte, tomates,
concombre, olives, pommes de terre", 2.99));
        ajouterPlat(new Plat("Salade parisienne", "Salade verte, tomates,
poulet, emmental", 2.99));
        ajouterPlat(new Plat("Soupe du jour", "Soupe du jour et croûtons
grillés", 3.29));

    }

    public void ajouterPlat(Plat plat){
        if(nbPlats < MAX_PLATS)
            plats[nbPlats++] = plat;
    }

    public Plat[] getPlats(){
        return plats;
    }

    public int getNbPlats(){
```

```
        return nbPlats;
    }

}
```

```
=======================================================
                    SERVEUSE
=======================================================
```

```
public class Serveuse {

    private MenuCreperie menuCreperie = null;
    private MenuCafet menuCafet = null;

    public Serveuse(MenuCreperie menuCreperie, MenuCafet menuCafet) {
        super(); //pas nécessaire
        this.menuCreperie = menuCreperie;
        this.menuCafet = menuCafet;
    }

    public void afficheMenus(){
        ArrayList<Plat> platsCreperie = menuCreperie.getPlats();
        Plat[] platsCafet = menuCafet.getPlats();

        for(int i=0;i<platsCreperie.size();i++){
            Plat plat = platsCreperie.get(i);
            System.out.println(plat.getNom() + " " + plat.getPrix() + " -- "
+ plat.getDescription());
        }

        for(int i=0;i<menuCafet.getNbPlats();i++){
            Plat plat = platsCafet[i];
            System.out.println(plat.getNom() + " " + plat.getPrix() + " -- "
+ plat.getDescription());
        }

    }
}
```

```
=======================================================
                    MAIN
=======================================================
```

```
public class Test {
```



```
    public static void main(String[] args) {
        MenuCreperie menuCreperie = new MenuCreperie();
        MenuCafet menuCafet = new MenuCafet();
        Serveuse serveuse = new Serveuse(menuCreperie, menuCafet);

        serveuse.afficheMenus();
    }

}
```

interface Iterateur<E> { bool encore(); E suivant(); }

int position = 0; encore position < plats.size().

suivant