Design Patterns - TD4

Code initial pour le TD4

Rappel du cours : 📱📱💻 http://bit.ly/jmb-cpoa

Informations générales

NOM

BRUEL

Prénom

Jean-Michel

Groupe #

- [x] Enseignants
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] Innopolis

Pré-requis

Il vous faut :

- [x] Un compte GitHub
- [ ] Un terminal de type Git Bash (si vous utilisez Window$)

Essayez la commande suivante dans votre terminal pour vérifier votre environnement `git` :

```
git config --global -l
```

Tâche initiale

- [x] Clickez sur le lien Github Classroom fourni par votre enseignant (en fait c'est déjà fait si vous lisez ces lignes).
- [ ] Clonez sur votre machine le projet Github généré pour vous par Github Classroom.

- ☐ Modifez le `README` pour modifier Nom, Prénom et Groupe.
- ☐ Commit & push:

commit/push

```
fix #0 Initial task done
```

Dans la suite de ce document, à chaque fois que vous trouverez un énoncé commençant par `fix #⋯` vous devez vérifier que vos scripts/fichiers modifiés sont bien dans votre dépôt local en vue de committer et de pusher les modifications sur votre dépôt distant en utilisant comme message de commit cet énoncé.

- Si vous voulez vérifier que vous êtes prêt pour le `fix #0`, utilisez la commande : `make check`.
- Si vous voulez avoir la liste des ToDos de ce TP/TP, exécutez `make todos`.

Les exercices de ce TD sont tirés de l'excellent livre "Tête la première : Design Pattern". Bert Bates, Eric Freeman, Elisabeth Freeman, Kathy Sierra. Editions O'Reilly. 2005.

1. Différences entre dépendance, association, composition, agrégation

Soit le diagramme de classe partiel suivant :

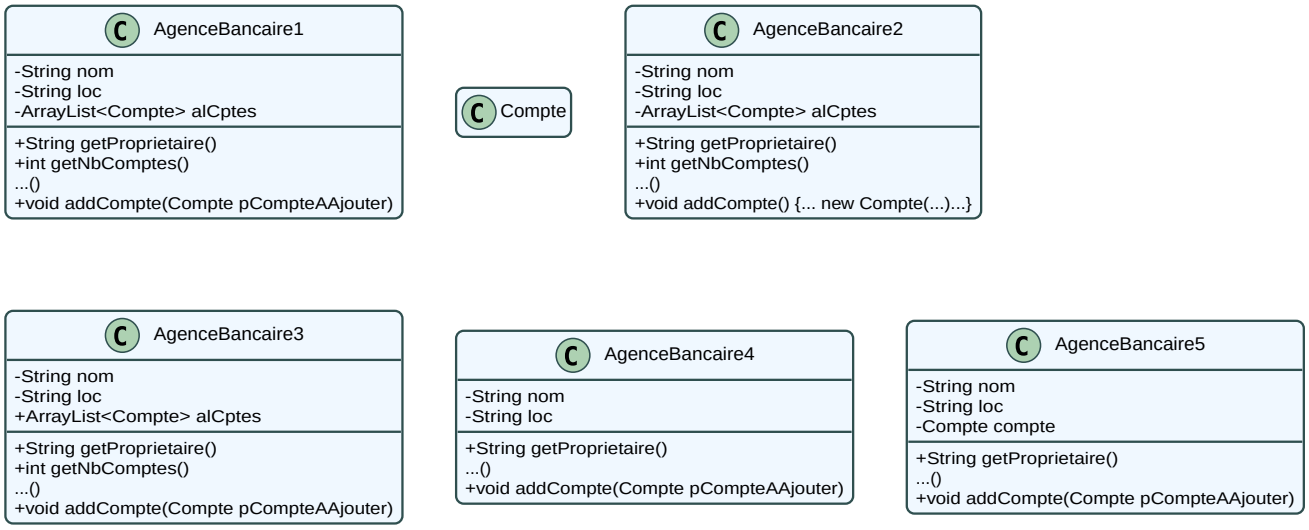

Figure 1. Diagramme de classe partiel

QUESTION

Complétez en ajoutant les relations (dépendance, association, composition, agrégation) entre les classes.

commit/push

```
fix #1.0 Completed class diagram
```

2. Patrons

QUESTION

Pour chacun des diagrammes de classe partiels suivants (représentant des patrons que vous connaissez), complétez :

- le nom du patron dans la légende,
- en ajoutant les relations.

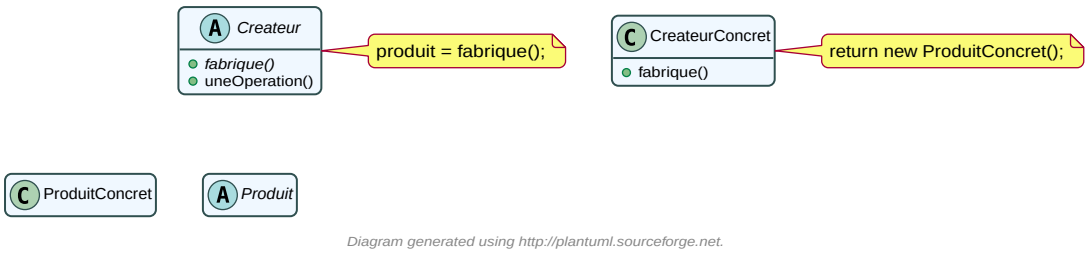

Figure 2. Patron ...

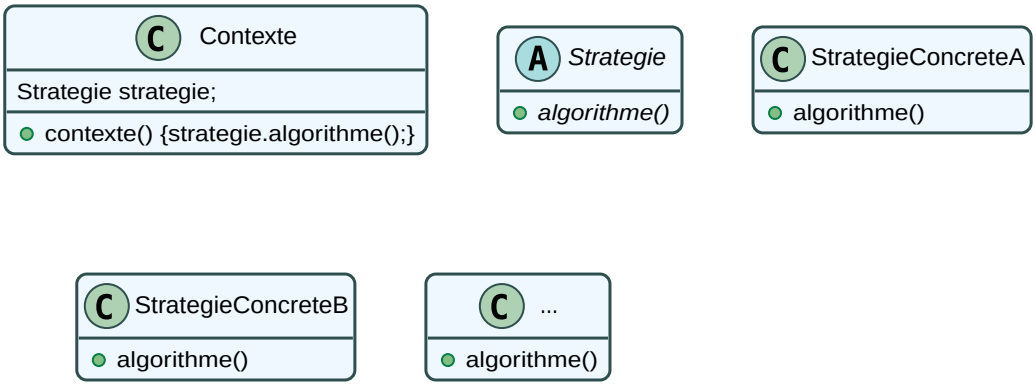

Figure 3. Patron ...

3. Machines à état

QUESTION

1. Dessinez le diagramme d'états correspondant aux feux tricolores (en France) classiques. Ajoutez la prise en compte de la panne dans un 2ème temps.
2. Dessinez le diagramme d'états correspondant au déroulement d'une partie d'échecs.

QUESTION

1. Commit&Push when everything is ready

   commit/push

   ```
   fix #2: State Machines...
   ```

4. Diagrammes de séquences

Vous devez documenter, à partir des extraits de codes Java suivants, l'application `ApplicationBanque`, développée en S2.

Vous refactorerez cette application en TP, l'objectif n'est donc pas pour l'instant de remédier aux problèmes de conception mais plutôt de les identifier.

Méthode statique `comptesDUnPropretaire` *(*`ApplicationAgenceBancaire.java`*)*

```
public static void comptesDUnPropretaire (AgenceBancaire ag, String nomProprietaire) {
    Compte []  t;

    t = ag.getComptesDe(nomProprietaire);
    if (t.length == 0) {
        System.out.println("pas de compte à ce nom ...");
    } else {
        System.out.println("  " + t.length + " comptes pour " + nomProprietaire);
        for (int i=0; i<t.length; i++)
            t[i].afficher();
    }
}
```

QUESTION

Réalisez un diagramme de séquence illustrant le fonctionnement de cette méthode.

Extrait de `AccesAgenceBancaire.java`

```
public class ApplicationAgenceBancaire {

    public static void main(String argv[]) {

        String choix;

        boolean continuer ;
        Scanner lect;
        AgenceBancaire monAg ;

        String nom, numero;
        Compte c;
        double montant;

        lect = new Scanner ( System.in );
        lect.useLocale(Locale.US);

        monAg = AccesAgenceBancaire.getAgenceBancaire();
```

```
        continuer = true;
        while (continuer) {
            ...
            choix = lect.next();
            choix = choix.toLowerCase();
            switch (choix) {
                case "q" :
                System.out.println("ByeBye");
                continuer = false;
                break;
                case "l" :
                monAg.afficher();
                break;
                case "v" :
                System.out.print("Num compte -> ");
                numero = lect.next();
                c = monAg.getCompte(numero);
                if (c==null) {
                    System.out.println("Compte inexistant ...");
                } else {
                    c.afficher();
                }
                break;
                case "p" :
                System.out.print("Propriétaire -> ");
                nom = lect.next();
                ApplicationAgenceBancaire.comptesDUnPropretaire (monAg, nom);
                break;
                case "d" :
                ...
                break;
                case "r" :
                ...
                break;
                default :
                ...
                break;
            }
        }
    }

    public static void comptesDUnPropretaire (AgenceBancaire ag,
        String nomProprietaire) {...}

    public static void deposerSurUnCompte (AgenceBancaire ag,
        String numeroCompte, double montant) {...}

    public static void retirerSurUnCompte (AgenceBancaire ag,
        String numeroCompte, double montant) {...}
}
```

Extrait de `AccesAgenceBancaire`

```
public class AccesAgenceBancaire {

    private AccesAgenceBancaire () {}
    public static AgenceBancaire getAgenceBancaire () {

        AgenceBancaire ag = new AgenceBancaire("CAISSE ECUREUIL", "PIBRAC");
        ...
    }
...
}
```

QUESTION

1. Commit&Push when everything is ready

   commit/push

   ```
   fix #3: Sequence diagram...
   ```

Appendix A: Pour Aller plus loin...

QUESTION

1. Réalisez le diagramme de classe de l'application
2. Que vous rappelle la classe `AccesAgenceBancaire`?
3. Réalisez un diagramme de séquence illustrant le fonctionnement de cette application (`main`). On utilisera des blocs "ref" pour les appels aux méthodes statiques, et on ne s'occupera pas des scanners.
4. Peut-on, dans un code Java, faire la différence entre agrégation `1 <>-> *` et association `1 -> *`?
5. Commit&Push when everything is ready

   commit/push

   ```
   fix #Bonus: Here is additional material...
   ```

Contributeurs

- Jean-Michel Bruel

À propos...

Baked with Asciidoctor (version 2.0.11) from 'Dan Allen', based on AsciiDoc. 'Licence Creative Commons'.  licence Creative Commons Paternité - Partage à l'Identique 3.0 non transposé.