Le patron *Façade*

Exercice 1 – *Une façade simple.*

Il est souvent utile, dans les processus de traitement ou d'analyse du texte, de disposer d'un indexe permettant d'utiliser un entier unique pour identifier un mot ou une séquence de texte (tokens). Par exemple, à partir d'un index vide, on obtient l'association suivante pour cette phrase :

Un	token	est	un	couple
0	1	2	0	3

On souhaite ici réaliser un tel indexe en utilisant la classe suivante :

```
1  import java.util.*;
2  import java.lang.*;
3
4  public class Token {
5    // La chaine de caracteres representant le token :
6    private String  _token;
7    // L'identificateur numerique du token :
8    private int _tokenId;
9    // Une annotation quelconque :
10   private String _annotation;
11
12   public Token(String token, int tokenId, String annotation) {
13     _token=token;
14     _tokenId=tokenId;
15     _annotation=annotation;
16   }
17
18   public String getTokenStr() { return _token; }
19   public int getTokenId() { return _tokenId; }
20   public String getAnnotation() { return _annotation; }
21 }
```

On souhaite avoir une relation (mapping) bi-directionnelle permettant de retrouver une association particulière soit en fonction de son identifiant numérique, soit en fonction du token (la chaîne de caractères). On utilisera pour le premier cas un vecteur, et pour le second cas une *Map* :

```
1  import java.util.*;
2  import java.lang.*;
3
4  public class IndexTest {
5    public static void main(String[] args) {
6      List<Token> idToToken=new ArrayList<Token>();
7      Map<String, Token> tokenToId=new HashMap<String, Token>();
8
9      ...
10   }
11 }
```

Cet indexe est donc réalisé en utilisant trois types d'objets différents qu'il faut maintenir à jour et synchronisés. La manipulation de ces différentes structures de données étant complexe et pouvant conduire facilement à des erreurs (incohérence structurelle), on souhaite implémenter une façade offrant les fonctions suivantes :

```
int addNewToken(String token, String annotation);
String getToken(int tokenId);
int getTokenId(String Token);
String getAnnotation(int tokenId);
Token getTokenMapping(int tokenId);
```

Question 1.1 : Dessinez le diagramme UML du système complet.
Question 1.2 : Pour chacune des fonctionalités, décrivez le traitement à réaliser.
Question 1.3 : Implémentez cette façade.
Question 1.4 : Devisez sur la qualité de l'interface qui est proposée pour la façade. Certaines méthodes sont-elles discutables ? (vis-à-vis, par exemple du principe d'isolation)

Exercice 2 – *Façades multiples d'un sous-système.*

Une entreprise de vente de véhicules souhaite permettre à ses clients d'accéder à la liste des véhicules en vente ainsi que de commander de la documentation papier, via un service web. Voici les éléments de base de ce système d'information. Les véhicules sont représentés par :

```
public interface Vehicule {
  /* returns the price of the vehicule */
  public int getPrice();
  /* returns the name of the vehicule */
  public String getName();
  /* gives an html description of the vehicule */
  public String getWebDescription();
  /* gives a text description for printing */
  public String getPrintableDocument();
}
```

et sont stockés dans une base de données représentée par un objet *DatabaseImpl* implémentant :

```
public interface Database {
  /* returns a vehicule by its name */
  public Vehicule getVehicule(String name);
  /* returns a list of vehicules by arange of prices */
  public List<Vehicule> getVehiculesByPrice(int low, int hi);
  /* add a vehicule into the database */
  public void addVehicule(Vehicule v);
}
```

Les services d'impression et de mailing sont représentés par les classes *PrinterImpl* et *MailingImpl* implémentant respectivement :

```
public interface Printer {
  /* print the documentation and return the job id */
  public int print(String doc);
}

public interface Mailing {
  /* send a documentation with a given print id to a given address and return
   * a confirmation message */
  public String send(int printid, String address);
}
```

L'imprimante par défaut s'obtient par

```
Printer p=PrinterImpl.getDefaultPrinter();
```

On souhaite disposer de deux types d'accès métiers différent :

- Un service web pour les clients permettant de rechercher un véhicule par son nom, par intervalle de prix ou de commander de la documentation,
- Une interface d'administration permettant la gestion du parc de vehicule.

Voici les deux interfaces de ces deux façades :

```
1  public interface FacadeWeb {
2    public String findVehicule(String name);
3    public String findVehiculeByPrice(int lo, int hi);
4    public String orderVehiculeDoc(String name, String address);
5  }
6
7  public interface FacadeAdmin {
8    public void addVehicule(String name, int price, String wdoc, String pdoc);
9  }
```

Questions :
Question 2.1 : Dessinez le diagramme UML du système complet,
Question 2.2 : implémentez les deux façades (récupérez les classes de base sur votre liste de diffusion),
Question 2.3 : testez les avec l'exemple de client suivant :

```
1  package tpfacade;
2
3  public class Client {
4    public static void main(String[] args) {
5      System.out.println("Debut client");
6
7      FacadeAdmin admin=new FacadeAdminImpl();
8      admin.addVehicule("Berline 5 portes", 6000,
9          "Compact 3 portes<br>Moteur diesel<br>Neuve<br>",
10         "Compact 3 portes\nMoteur diesel\nNeuve");
11     admin.addVehicule("Espace 5 portes", 8000,
12         "Espace 5 portes<br>Moteur essence<br>Neuve<br>",
13         "Espace 5 portes\nMoteur essance\nNeuve");
14     admin.addVehicule("Coupe 2 portes", 3000,
15         "Utilitaire 3 portes<br>Moteur diesel<br>Occasion<br>",
16         "Utilitaire 3 portes\nMoteur diesel\nOccasion");
17
18     FacadeWeb web=new FacadeWebImpl();
19     System.out.println("requete espace 5 portes :\n" +
20         web.findVehicule("Espace 5 portes"));
21     System.out.println("requete vehicule a moins de 7000 euros :\n" +
22         web.findVehiculeByPrice(0, 7000));
23
24     System.out.println("Demande de documentation :\n" +
25         web.orderVehiculeDoc("Berline 5 portes",
26         "Avenue de l'universite, 76800, Saint Etienne-du-Rouvray"));
27     System.out.println("Demande de documentation :\n" +
28         web.orderVehiculeDoc("Coupe 2 portes",
29         "Avenue de l'universite, 76800, Saint Etienne-du-Rouvray"));
30     System.out.println("Fin client");
31   }
32 }
```

Produisant la sortie suivante :

```
1  Debut client
2  requete espace 5 portes:
3  <html><body>
4  Espace 5 portes<br>Moteur essence<br>Neuve<br>
5  </body></html>
6
7  requete vehicule a moins de 7000 euros:
8  <html><body>
9  Compact 3 portes<br>Moteur diesel<br>Neuve<br>
10 Utilitaire 3 portes<br>Moteur diesel<br>Occasion<br>
11 </body></html>
12
13 Demande de documentation:
```

```
14 <html><body>
15 The documentation order no 1 will be sent to Avenue de l'universite, 76800, Saint Etienne-
       du-Rouvray
16 </body></html>
17
18 Demande de documentation :
19 <html><body>
20 The documentation order no 2 will be sent to Avenue de l'universite, 76800, Saint Etienne-
       du-Rouvray
21 </body></html>
22
23 Fin client
```