Faculté des Sciences
Aix*Marseille Université

Année universitaire 2017/2018

Site : ☐ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☒ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☐ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve : 1h30
Examen de : L3 Nom du diplôme : Licence d'Informatique
Code du module : SIN5U2TL Libellé du module : Programmation et conception orientées objet
Calculatrices autorisées : NON Documents autorisés : OUI, notes de Cours/TD/TP

Exercice 1

La classe suivante permet de stocker un certain nombre de mots et de déterminer rapidement si un mot appartient au dictionnaire et dans le cas échéant de donner sa définition (vous remarquerez une structure d'arbre préfixe).

```
public class Dictionnary {
        Node root = new Node()

        private class Node{
                private ArrayList<Node> arrayList = new ArrayList<Node>();
                private String word;
                private String definition;
                private Character lastChar;
                private boolean isEndChar = false;

                Node(Character lastChar){this.lastChar = lastChar;}

                public Node next(char nextChar){
                        for(Node child : arrayList)
                                if (child.lastChar == nextChar) return child;
                        return null;
        }

        public void put(String word, String definition){
                Node current= root;
                for(char nextChar: word.toCharArray()){
                        if (current.next(nextChar) == null)
                                current.arrayList.add(new Node(nextChar));
                        current = current.next(nextChar);
                        }
                current.isEndChar = true;
                current.word = word;
                current.definition = definition;
        }

        public String get(String word){
                Node current = root;
                for(char nextChar: word.toCarArray()){
                        current = current.next(nextChar);
                        if current == null return null;
                if current.isEndChar return current.definition;
                else return null;
                }
}
```

Attention : Chaque question suppose que le code a été modifié selon les instructions des questions précédentes. Les paramètres des classes ne sont pas précisés dans l'énoncé, mais elle peuvent en avoir.

1. Ajouter une méthode `print` qui affiche assez simplement dans le terminal l'ensemble des mots avec leur définition associée dans le terminal, vous pouvez ajouter d'autres méthodes.
2. Nous souhaiterions afficher ce dictionnaire sur différents formats, HTML/latex par exemple.

```
public static void main(String[] args){
        Dictionnary dictionnary = new Dictionnary()
        dictionnary.put("Chat","Ronronne quand on le caresse");
        dictionnary.put("Chien","Ramène la balle");
        dictionnary.print(new LatexFormat());
        dictionnary.print(new HTMLFormat());
        }
```

   Lors de son exécution ce code doit afficher dans le terminal :

```
\begin{itemize}
\item[Chat] Ronronne quand on le caresse
\item[Chien] Ramène la balle
\end{itemize}
<ul>
<li> Chat : Ronronne quand on le caresse <\li>
<li> Chien : Ramène la balle <\li>
<\ul>
```

   Créer les classes nécessaires pour une bonne organisation de cet affichage et une extension facile du type d'affichage.

3. L'attribut `definition` est désormais une `ArrayList<String>`. Pour l'affichage celle-ci doit être mise en forme en fonction du format. Les définitions de `chat` et `chien` ont été mise a jour pour contenir deux éléments. Ajouter les méthodes nécessaires pour obtenir cet affichage (pas besoin de tenir compte des tabulations) :

```
\begin{itemize}
\item[Chat]
        \begin{itemize}
        \item Ronronne quand on le caresse
        \item Chasse les souris
        \end{itemize}
\item[Chien]
        \begin{itemize}
        \item Ramène la balle
        \item Agite la queue
        \end{itemize}
\end{itemize}
<ul>
<li> Chat :
        <ul>
        <li> Ronronne quand on le caresse <\li>
        <li> Chasse les souris <\li>
        <\ul>
<\li>
<li> Chien :
        <ul>
        <li> Ramène la balle <\li>
        <li> Agite la queue <\i>
        <\ul>
<\li>
<\ul>
```

4. Changer le code pour que l'attribut définition soit un Printable à la place du `String` ou `ArrayList<String>`.

```
public interface Printable{
        public void print(PrintFormat format);
        }
```

5. Écrire le code d'une classe implémente Printable et qui contient une liste de `String` de manière à ce que cela s'affiche comme précédemment selon le format voulu. Écrire également le code d'une classe qui implémente Printable et qui contient juste une `String`. Cela va vous permettre d'avoir un dictionnaire composite dont les définitions seront soit un paragraphe, soit une liste de paragraphe.
6. Voici l'interface `Map` (réduite pour l'examen).

```
public interface Map<K,V>{
        public void put(K key, V value);/* ajoute value associée à key dans la Map*/
        public V get(K key); /* renvoie la valeur associée à la clé si la clé existe
                                                        et null sinon*/
        }
```

   La classe `HashMap<K,V>` implémente cette interface.

   Remplacer l'attribut `arraylist` de `Node` par une `HashMap`, faire les modifications pour que l'exécution du code reste identique.
7. Changer l'attribut `word` qui est actuellement un `String` en `Iterable` générique. Faire les modifications nécessaires uniquement dans la classe `Dictionnary`. Vous pouvez remarquer qu'en rendant la classe générique elle ne peut plus avoir des `String` comme clés, car les `String` ne sont pas `Iterable`, le code du `main` ne peut plus s'exécuter correctement.
8. Implémenter une classe `DictionnaryAdapter` qui prend des clés `String` et utilise un objet de la classe `Dictionnary` (avec des clés `Iterable`) de manière à fournir les mêmes services que précédemment : le `main` initial doit pouvoir s'exécuter en remplaçant `new Dictionnary()` par `new DictionnaryAdapter()`.
9. Expliquer pourquoi `Dictionnary` ne peut pas implémenter `Map<K,V>`. Quelle interface proche la classe `Dictionnary` peut implémenter.