Projet d' Architecture Orientée Objet

Sujet :

Réalisation d'un assistant personnel commandé par langage naturel. Ce programme a pour objectif de simplifier la vie d'un utilisateur. Pour cela on peut interagir avec lui via des commandes exprimées en **langage naturel**. L'assistant doit être ouvert à l'ajout de nouvelles fonctionnalités via un système de plugin.

Plugin :

Définition (Wikipedia) :

« En informatique, un plugin ou plug-in (aussi nommé module d'extension, greffon ou plugiciel au Québec) est un logiciel qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités. »

Fonctionnalités de base :

Gestion des contacts :

L'assistant doit être capable d'interroger une application de contact. Au choix, vous pourrez utiliser un système de gestion de contact (Google, Yahoo, Windows live… ou autre) ou un simple fichier comme base de contact.

L'assistant doit pouvoir :

Afficher les informations classiques d'un contact **: nom, prénom, adresse, mail (liste), téléphone (liste), date de naissance.** Vous êtes libre d'ajouter d'autres informations. Pour cela il devra répondre aux questions :

- « Connais-tu Michel Durant ? »
- « Qui est Michel Durant ? »
- « Où habite Michel Durant ? »
- « Où vit Michel Durant ? »
- « Quand est né Michel Durant ? »
- « Quelle est la date de naissance de Michel Durant ? »
- « Quel âge à Michel Durant ? »
- « Dis-moi tout ce que tu sais sur Michel Durant. »
- « Quel est le numéro de téléphone de Michel Durant ? »
- « Qui habite à Paris ? »
- « Qui a plus de 40 ans ? »
- Vous êtes libres (et encouragés) à multiplier les formulations possibles.

Gestion du calendrier :

L'assistant doit être capable d'interroger une application de gestion de rendez-vous. Au choix, vous pourrez soit développer une application de gestion des rendez-vous, soit utiliser un système existant (Google, Yahoo, Windows live… ou autre). Vous pouvez aussi utiliser un simple fichier comme base des rendez-vous. Un rendez-vous (ou évènement) à **une heure de début, heure de fin, titre, participants (liste), lieu.** Vous êtes libre d'ajouter d'autres informations. Pour cela il devra répondre aux questions :

- « Quel est le programme de ma journée ? »
- « Quel est le programme de la semaine ? »
- « Ai-je un rendez-vous aujourd'hui ? »
- « Quand est ce que je verrai Michel Durant ? »
- « Prend rendez-vous avec Michel Durant lundi à 18h à Paris. »
- « Quelle heure est-il ? »
- « Quand est mon prochain rendez-vous ? »
- Vous êtes libres (et encouragés) à multiplier les formulations possibles.

Gestion des plugins :

L'architecture du programme devra permettre l'ajout de nouvelle fonctionnalité sans avoir à modifier le code existant du programme. Seul le code de la classe « Main » aura le droit d'être modifié pour ajouter un plugin à l'application (on préfèrera cependant passer par un fichier de configuration, ou un chargement dynamique).

Principe de fonctionnement de la commande en langage naturel

La compréhension du langage naturel est très complexe à réaliser par un ordinateur, c'est pourquoi il n'existe pas à l'heure actuelle de système informatique capable de converser en langage naturel. Cependant, il est possible de l'approximer en identifiant des phrases/mots clés et en leur associant des actions.

Exemple :

- « Quand est né Michel Durand ? » => Quand est né X ? où X = Michel Durant => rechercher Michel Durant dans l'application de contact et renvoyer la date de son anniversaire.
- « Quelle est la date de naissance de Michel Durant ? » <=> Quand est né X ? où X = Michel Durant => rechercher Michel Durant dans l'application de contact et renvoyer la date de son anniversaire

On pourra utiliser les expressions régulières pour réaliser cette partie.

Dans le cas où une commande pourrait renvoyer plusieurs résultats, alors on retournera tous les résultats à l'utilisateur.

Exemple de plugin :

- Ouverture d'une application => « Lance Firefox !»
- Météo => « Est-ce qu'il va pleuvoir ce soir ? »

- Dictionnaire => « Que signifie anticonstitutionnellement ? »
- …

Attendu :

- L'application fonctionnelle. L'interface peut être réalisée de la manière de votre choix
- Le diagramme UML de classe de l'application.
- L'ensemble du code source de l'application développée avec Maven
- Un plugin et son code source.
- Expliquez en quelques paragraphes vos choix d'architecture de l'application.

Remarque :

Le but du projet est d'évaluer votre capacité à architecturer correctement votre application. Il sera donc plus important d'avoir un logiciel bien construit, commenté et facilement extensible plutôt qu'une application avec beaucoup de fonctionnalités mais mal architecturée.

Annexes :

Utilisation des expressions régulières

```
Pattern p = Pattern.compile("^QUE SIGNIFIE (.*)$",
Pattern.CASE_INSENSITIVE);
Matcher m = p.matcher("Que signifie voiture");
List<String> params = new ArrayList<String>();
if (m.find()) {
     for (int j = 0; j < m.groupCount(); j++) {
          params.add(m.group(j + 1));
     }
     //On retourne la liste des expressions trouvées
     return params;
}else {
     //On a rien trouvé
     return null;
}
```

Chargement dynamique d'une classe

```
String pluginPath = « …/plugin/«  ;
File f = new File(pluginPath);
File[] files = f.listFiles();
URL[] t = new URL[files.length];
for (int i = 0; i < files.length; i++) {
     t[i] = files[i].toURI().toURL();
}
ClassLoader loader = new
URLClassLoader(t,this.getClass().getClassLoader());
MaClasse o =(MaClasse)loader.loadClass("package.MaClasse")
                                          .newInstance();
```