Projet Programmation Orientée Objet

2018 - 2019

L'objectif de ce projet est de réaliser un chat permettant de discuter avec des chatbots en ligne de commande. Ce chat proposera une collection de chatbots qui vont permettre à l'utilisateur de récupérer diverses informations disponibles notamment sur internet (à travers des APIs) directement depuis son terminal à travers un chat.

Pour faciliter le travail des groupes, qui seront des binômes, le projet est découpé en quatre itérations successives. Chaque itération fera l'objet d'une démo et d'une rapide revue de code durant le début de chaque séance de suivi de projet (sauf la première, dédiée à la clarification du sujet et au lancement du projet).

Étape 1 : mentionner un chatbot

L'objectif de cette étape sera d'implémenter une session de chat ou on pourra préciser quel pseudo nous souhaitons utiliser, ainsi qu'un chatbot basique qui ne fera que répondre bonjour ainsi que la date et l'heure lorsqu'on le mentionne, comme sur l'exemple suivant.

```
shell$ java -jar chatbot.jar -p toto

[toto] @hello
[hello] Salut toto! Nous sommes Jeudi 25 octobre et il est 14h30.
[toto] foo
[toto] ++

shell$
```

Il vous faudra donc :

- Récupérer la valeur de l'argument "-p".
- Lancer une session de chat ou l'application attendra que l'utilisateur saisisse son message, on considère que l'utilisateur a terminé lorsqu'il appuiera sur la touche "Entrer" (astuce: regarder du côté de *System.in*)
- En cas d'un message égal à @hello, le chatbot répond comme sur l'exemple ci-dessus, en cas d'autre message, le chatbot ne répond pas.
- Pour quitter la session (et donc terminer l'exécution de l'application), l'utilisateur devra saisir le message "++".

Étape 2 : un système de chargement de bots avec traitement des erreurs

Avant d'aller plus loin dans le projet, il vous est demandé de vous poser un instant et de réfléchir à l'architecture de votre application.
Jusqu'à présent, nous avons eu à utiliser qu'un seul chatbot, cependant, dans la suite de ce projet, il nous sera possible de discuter une infinité de chatbots.
Il va donc falloir pouvoir charger autant de chatbots que nécessaire, de s'assurer que chaque chatbot a bien un nom unique, de lui transmettre les messages qui lui sont adressés ainsi que les arguments dont il a besoin, de gérer les situations ou le chatbot adressé n'existe pas comme présenté dans l'exemple qui suit, etc...

Note: Le design doit permettre de pouvoir ajouter pleins de bots de la manière la plus légère possible.

```
shell$ java -jar chatbot.jar -p toto

[toto] @hello
[hello] Salut toto!
[toto] @time
[hello] Nous sommes Jeudi 25 octobre et il est 14h30.
[toto] @foo
[system] Je ne connais pas le chatbot @foo!
```

Étape 3 : un chatbot consommant une API REST et permettant de fournir des paramètres

Dans cette seconde étape, vous allez devoir concevoir un chatbot qui devra envoyer une requête HTTP vers un serveur distant afin de récupérer l'information nécessaire à l'utilisateur. Les différentes requêtes HTTP que l'on peut envoyer au serveur forment ce que l'on appelle une API.
Pour notre premier chatbot connecté nous allons utiliser l'API http://www.icndb.com/api/ qui permet de récupérer les célèbres Chuck Norris Facts.
La réponse à la requête se fait dans le format JSON. Depuis Java il faudra donc faire une requête HTTP et analyser sa réponse JSON. Pour ce faire, il vous faudra utiliser la classe *java.net.HttpUrlConnection* de la librairie standard de Java pour effectuer la requête et utiliser un parseur JSON pour analyser la réponse. Nous vous conseillons le parser org.json qui est très simple d'utilisation et bien documenté. Vous pouvez retrouver sa documentation sur le site suivant: https://stleary.github.io/JSON-java. Il vous faudra trouver comment ajouter ce parser au classpath de votre programme.

```
shell$ java -jar chatbot.jar -p toto

[toto] @icndb
[icndb] Chuck Norris can win a game of Connect Four in only three moves.
```

Dans un deuxième temps, vous devez ajoutez la capacité pour le chatbot de recevoir en paramètre le nombre de Chuck Norris Facts à afficher. Prenez soin de mettre à jour votre système de traitement des erreurs pour pouvoir prévenir les utilisateurs quand les paramètres envoyés au bot sont incorrects.
N'oubliez pas également de prendre en charge les situations spécifiques au chatbot tel que le fait que la connexion à internet ne soit pas disponible, ou que le serveur ne répond pas.

```
shell$ java -jar chatbot.jar -p toto

[toto] @icndb 2
[icndb] Chuck Norris can win a game of Connect Four in only three moves.
[icndb] There are no steroids in baseball. Just players Chuck Norris has
breathed on.
[toto] @icndb foo
[icndb] Je ne sais pas quoi faire avec foo?
```

Étape 3 : un chatbot conversationnel

Dans cette nouvelle étape, nous allons implémenter un chatbot plus complexe avec qui ont peut avoir une conversation. Le principe de ce bot est de faire un jeu de type quiz avec gestion du score comme sur l'exemple suivant.

```
shell$ java -jar chatbot.jar -p toto

[toto] @quiz 2
[quiz] Question 1 : From which country does the piano originate?
  1- Germany
  2- Austria
  3- France
  4- Italy
[toto] 3
[quiz] Mauvaise réponse! La bonne réponse était: Italy!
[quiz] Question 2 : The LS2 engine is how many cubic inches?
  1 - 364
  2 - 346
  3 - 376
  4 - 402
```

```
[toto] 1
[quiz] Bonne réponse!
[quiz] Score final: 50% de bonnes réponses (1/2)
```

Bien entendu, il doit être possible à tout moment d'interrompre la conversation avec le bot quizz en utilisant le mot-clé @*stop*.

Pour le quizz, nous allons utiliser une nouvelle API disponible sur l'adresse suivante: https://opentdb.com/api.php?amount=10. Comme vous l'aurez remarqué, il est possible de passer un paramètre à travers l'URL, ce paramètre représente le nombre de questions qui seront retournées par le serveur. Ce nombre sera choisi par l'utilisateur lors de sa première mention du chatbot.

Etape 4 : des bots et encore des bots

Durant cette étape, nous vous laissons carte blanche pour implémenter un ou plusieurs bots de votre choix. Nous vous demandons simplement dans le README de décrire comment utiliser les bots, et aussi de nous résumer les changements éventuels qui vous avez du faire dans le système de gestion des bots.

Contraintes sur le code

Vous devez absolument RESPECTER ces contraintes. Une pénalité de 2 points sera appliquée sur chaque projet qui ne les respectent pas.

- Votre projet contient plusieurs paquetages.
- Les noms des paquetages sont intégralement en minuscules.
- Les noms des classes et interfaces commencent par une majuscule.
- Les noms des méthodes, des attributs et des variables commencent par une minuscule.
- Utilisez le *camel case* pour les noms des classes, interfaces, méthodes et attributs.
- Les attributs ont une visibilité `private` ou `protected`. Des accesseurs permettent si nécessaire d'accéder en lecture et/ou écriture aux attributs.
- La visibilité des méthodes dépend de leur utilisation.
- Les constructeurs sont définis uniquement s'ils sont nécessaires.
- Le code des méthodes compliquées doit être commenté.