IFT232

Laboratoire #6

Patrons de conception Template method, factory method et abstract factory

Ce laboratoire est à remettre 1 semaine après la séance de laboratoire.

Tous les numéros doivent être complétés.

Ce laboratoire comporte plusieurs exercices qui se basent sur un morceau de code déjà écrit. On assume que vous allez réaliser ces exercices dans l'environnement Eclipse. Le code de base du laboratoire se trouve sur le lecteur réseau public, c'est l'archive **labo6base.zip**.

Chargez le code de base du laboratoire dans Eclipse.

Exercice 0 : Reconnaissance dans le code

Tout d'abord, exécutez le code. Pour ce faire, choisissez la classe **Labo6Main** et appuyez sur le bouton play (ou choisissez l'option run as... dans le menu contextuel). Exécutez-le sous forme d'application java.

Le programme actuel affiche une fenêtre qui contient deux profils d'utilisateurs qui peuvent discuter avec les boîtes de texte. Le profil de gauche représente une personne (vous devriez écrire dans sa boîte de texte) qui a un genre et une nationalité. Le profil de droite représente une intelligence artificielle de conversation (ChatBot). Les réponses qu'il offrira ainsi que la photo de profil et son genre vont changer pour plusieurs

raisons à travers le laboratoire. Pour l'instant, il répond au hasard, choisit une photo de profil au hasard et possède un genre déterminé au hasard.

Trois boutons permettent de changer le comportement de l'intelligence artificielle. Le bouton SUIVANT crée une nouvelle intelligence artificielle dans la même session de discussion (on joue à la roulette des interlocuteurs). Le bouton GENRE change le genre de l'interlocuteur humain, et le bouton PAYS change son pays d'origine. Dans ces deux cas, une nouvelle session de discussion est démarrée.

Quand l'utilisateur humain ne change pas, la session reste ouverte, quand il change, la session se termine et une autre redémarre.

Description des classes

Labo6Main : Programme principal, utilisé pour démarrer le logiciel.

Très peu de changements à apporter ici. Seule la méthode **startTheRoulette** vous intéresse dans cette classe.

Package database :

Ce package contient deux classes représentant une base de donnée de photos de profil et une base de donnée de messages en texte. Les classes TextList et PictureList permettent de manipuler des listes d'images et de messages de texte. L'objectif de ces classes est de vous permettre de choisir des photos et des messages selon certains critères. Les critères et leurs valeurs sont passées aux méthodes keep() et remove() des classes PictureList et TextList.

Par exemple,

```
PictureList list = PictureDataBase.getAllPictures()
```

obtient toutes les photos.

```
list.keep(PictureKey.Gender,Gender.female);
```

Ne conserve que les photos de femmes (retire toutes les autres de cette liste).

ATTENTION! La méthode keep() et la méthode remove() modifient la liste de laquelle vous appelez la méthode keep ou remove. Ces méthodes ne retournent pas une nouvelle liste.

Vous pouvez faire des opérations ensemblistes facilement avec keep() et remove() et ainsi personnaliser les messages et les photos selon certains critères.

Session :

Cette classe représente la session de discussion avec un utilisateur humain.

Le comportement se trouvant dans la méthode start() est le focus de la première partie du labo (**Template Method**).

ChatBot :

Cette classe définit une intelligence artificielle de conversation. Les ajouts à cette classe ainsi que la création de sous-classes et de classes connexes est le sujet de la seconde partie du labo (**Factories**).

User :

Cette classe représente un utilisateur de l'application, que ce soit l'utilisateur humain (profil de gauche) ou l'utilisateur artificiel (profil de droite).

Package ui

Ce package contient des classes utiles à la génération de l'interface graphique.

Exercice 1 : Génération de réponses

Présentement, la réponse que le robot donne à chaque fois qu'un utilisateur modifie sa boîte de texte est toujours là même.

Afin d'avoir un comportement plus flexible, écrivez une méthode **generateAnswer**() qui a pour objectif de générer une réponse à la demande.

Pour l'instant, son implémentation devrait être d'obtenir n'importe quelle phrase au hasard dans la base de données de texte (`TextDatabase.getAllMessages().random().getMessage()`)

Exercice 2 : Deux types de réponses

Maintenant qu'on a encapsulé la génération d'une réponse dans une méthode, on peut en écrire plusieurs versions.

Écrivez une nouvelle classe se nommant **SeductionSession**, dérivée de Session, qui redéfinit la méthode generateAnswer() pour ne retourner que des messages séduisants. Vous pouvez filtrer les messages à l'aide de la méthode `keep()` de la classe TextList, en utilisant la clé `TextKey.isSeductive`.

Pour pouvoir faire une session de ce type, modifiez la classe **Labo6Main** pour que ce type de session soit supporté. Ajoutez une constante `SEDUCTION_SESSION = "seduction"`, puis, dans la méthode startTheRoulette, ajoutez une condition qui détecte qu'on a demandé ce type de session et qui crée le bon type.

Finalement, changez les paramètres d'exécution du programme (menu Run, Run configurations... , onglet arguments) pour tester votre nouvelle classe.

Votre robot ne devrait répondre que par des messages séduisants si vous êtes dans ce type de session.

Exercice 3 : Type de session de plus.

Créez un nouveau type de session, CasualSession, qui génère des réponses qui évitent les messages séduisants. Ajoutez également le support pour ce type de session dans Labo6Main et testez-la. Votre robot ne devrait plus envoyer de messages séduisants si vous êtes dans ce type de session.

Exercice 4 : Mot de bienvenue

Présentement, le robot affiche un message de bienvenue qui est toujours le même. Ceci devrait dépendre du type de session, et potentiellement d'autre chose à l'avenir.

Remplacez le message de bienvenue par un appel à une méthode nommée **generateGreeting**(). Implantez des versions différentes de cette méthode : une qui trouve dans la base de données un message de bienvenue séduisant pour SeductionSession, et un autre en trouve un qui n'est pas séduisant pour CasualSession. L'implémentation par défaut est d'utiliser n'importe quel message de bienvenue (TextKey.isGeeting).

Exercice 5 : Profilage

Le type de session devrait également déterminer quelles photos sont utilisables pour le profil du robot.

Écrivez une méthode **getSuitablePictures**() qui produit une PictureList contenant seulement les photos séduisantes pour la SeductionSession, et seulement les photos non-séduisantes pour CasualSession. L'implémentation par défaut permet toutes les photos.

Écrivez une méthode getSuitableMessages qui produit une TextList contenant seulement les messages de texte séduisants pour la SeductionSession, et et seulement les messages non-séduisants pour CasualSession. L'implémentation par défaut permet tous les messages.

Passez cette liste de messages en paramètre à vos anciennes méthodes (generateGreeting et generateAnswer), puis simplifiez leur implémentation si possible. Selon votre implémentation, vous pouvez avoir besoin de passer une COPIE de la liste à ces méthodes, comme ceci : generateGreeting (liste.clone()).

Exercice 6 : Factory Method

Déplacez le code qui détermine quel type de session instancier qui est présentement dans la méthode startTheRoulette de Labo5Main vers la classe Session, dans une méthode de classe (static) se nommant : createSession(String type). Déplacez également les constantes représentant les types de session dans Session.

Exercice 7: Réveil conditionnel

Modifiez la méthode start() de la classe Session pour qu'au lieu de comparer simplement le texte de l'utilisateur avec l'ancien texte, une méthode boolean checkForWakeUp(String message) soit appelée.

Cette méthode permet le réveil du robot sous diverses conditions.

Le robot lui-même devrait décider s'il se réveille ou non, selon sa personnalité. Implantez donc cette méthode chez ChatBot, et écrivez

plusieurs sous-classes de ChatBot (PatientChatBot, ImpatientChatBot, SlowmoChatBot). Le robot patient ne se réveille que si la dernière ligne contient le caractère « ? » (si l'humain lui pose une question), tandis que l'impatient se réveille si l'interlocuteur écrit quoi que ce soit, alors que finalement, le slowmo ne se réveille que si l'utilisateur répète 2 fois de suite la même chose (c'est un robot un peu dur de la feuille).

Pour savoir quel type de robot instancier au début de la méthode, vous allez devoir repousser la décision à une Factory Method : createChatBot().

Implantez une version par défaut de cette méthode qui crée un PatientChatBot, la version de la méthode pour SeductionSession crée un ImpatientChatBot, tandis que la CasualSession crée un SlowmoChatBot.

Exercice 8: Attente complexe

Modifiez l'appel à robot.sleep() pour un comportement plus flexible, que vous allez encapsuler dans la méthode waitForUser().

Les trois robots existants auront trois comportements différents : Le robot Slowmo ne fait que sleep(2000), le robot Impatient attend 1000 millisecondes (donc sleep(1000)), puis écrit un message au hasard pour relancer la conversation, tandis que le robot Patient attend 3000 millisecondes puis pose une question.

Exercice 9: Comportements amovibles (patron Strategy)

Les comportements de vérification de l'utilisateur et d'attente ne devraient pas être fixés ensemble comme dans les étapes précédentes : on aimerait bien avoir un robot dur de la feuille qui pose des questions après une attente, par exemple, ce qui n'est pas possible présentement sans ajouter une nouvelle classe de robot. On voudrait également que ces comportements dépendent d'autre chose que du type de session.

Pour y arriver, encapsulez les comportements dans des classes :

La classe abstraite WaitBehavior et ses descendantes, WaitBehaviorNohing, WaitBehaviorAsk, WaitBehaviorSaySomething, représentant les trois comportements d'attente, ainsi que la classe CheckUserBehavior et ses descendantes représentant les comportements de vérification.

Éliminez les 3 sous-classes de ChatBot. Ajoutez une variable de type WaitBehavior et une variable de type CheckUserBehavior dans ChatBot.

Pour les initialiser, modifiez les implémentations de createChatBot dans les classes Session, CasualSession et SeductionSession. Ajoutez dans chacune des classes une méthode servant à créer le bon type de WaitBehavior et de CheckBehavior : createCheckBehavior et createWaitBehavior.

Exercice 10: Profil avancé (quasi Abstract Factory)

Afin de bien construire un robot correspondant à la situation, écrivez une nouvelle classe abstraite **Profiler**, vers laquelle vous allez déplacer les méthodes suivantes : generateGreeting, generateAnswer, getSuitablePictures, getSuitableMessages, createWaitBehavior, createCheckBehavior.

De plus, déplacez vers cette classe la méthode qui sert à construire le robot (createChatBot).

Ensuite, implantez trois classes descendantes : NormalProfile, CasualProfile, SeductiveProfile.

Modifiez ensuite la classe Session pour que les appels aux anciennes méthodes soient faites chez le Profiler.

Écrivez une factory method createProfiler qui sera appelée au début de la méthode start() pour initialiser le Profiler. Les implémentations de cette méthode dans les classes Session, CasualSession et SeductiveSession choisiront chacune le bon type de Profiler.

Finalement, modifiez la méthode createChatBot pour que le genre du robot corresponde toujours au genre de la photo de profil, puis ensuite, pour que le genre du robot créé par un SeductionProfile soit du genre opposé à l'interlocuteur (sauf si l'interlocuteur a un genre inconnu, auquel cas il sera également inconnu). Le genre affiché doit toujours correspondre à celui de la photo.

Exercice 11: Profil par région

Modifiez les implémentatons de getSuitableMessages pour que seuls les messages dans la bonne langue puissent être utilisés, selon le pays d'origine de l'interlocuteur : Français (Québec, France), Anglais (les autres).

De plus, modifiez le choix de photos (getSuitablePhotos) pour qu'un interlocuteur Japonais obtienne des robots avec des photos qui ne sont que des dessins animés (PictureKey.isComic).

Exercice 12: Enfin! Tout ce beau code pour...

Créez un nouveau type de session (TrollSession) et le Profiler correspondant (TrollProfiler) pour lequel le robot n'écrit jamais de messages séduisants. Lorsqu'il est en attente, il écrit des messages après 1000 millisecondes. Si l'interlocuteur lui écrit, il répond uniquement avec des messages offensifs (TextKey.isOffensive). S'il était en attente et a écrit plusieurs messages de suite, il passe de l'envoi de messages normaux à des messages passif-agressifs. Son mot de bienvenu est offensif. Finalement, il choisir la langue contraire à celle qui correspond au pays de l'interlocuteur.

Il choisit son genre au hasard, et s'il tombe sur le même genre que son interlocuteur, alors il n'écrit que des messages séduisants.