Contrôle M2104–BCOO

Mercredi 15 juin 2016 – Une seule feuille A4 manuscrite autorisée
Durée : 1h30

Description

Votre rôle est de spécifier un nouveau système informatique à embarquer dans toutes les voitures connectées : MyWaze [1] !

Vous avez vu un GPS ? Vous avez utilisé GoogleMap ? Vous connaissez Waze ? Au volant, pas toujours facile à utiliser... Vous avez déjà Twitté ? Vous avez parfois envoyé un email, un SMS ? Envoyé une photo sur Instagram ? Jamais en conduisant évidemment !! Et bien nous allons construire un boitier embarqué dans votre voiture qui vous permettra de faire tout cela en conduisant [2].

MyWase doit permettre à un "conducteur" de s'identifier, visualiser la carte, spécifier un itinéraire, d'envoyer un message, de recevoir un message. Un conducteur n'a pas besoin de s'identifier pour demander à visualiser une carte, mais dans tous les autres cas c'est indispensable. Il peut s'identifier soit par un mot de passe, soit en utilisant des périphériques extérieurs comme un lecteur d'empreintes digitales. Le conducteur doit pouvoir commander le boitier à la voix ou au toucher. Un "gentilAdministrateur" peut configurer le boitier pour l'associer à un conducteur en précisant les informations sur ses réseaux sociaux (par exemple, le compte "chut" sur Twitter, "NSA" sur google+, etc.). Il doit également pouvoir créer de nouveaux modèles de messages, dits "créatifs", en modifier ou en détruire. Un modèle de messages peut être soit prédéfini et dans ce cas, il s'agit des modèles de messages fournis par Waze (accident, bouchon, ...) , soit "créatif" et dans ce cas nous nous nous

1. Merci à Mireille Blay-Fornarino, du département informatique de l'IUT de l'Université de Nice Sophia Antipolis pour ce sujet.

2. Bien sûr tout au long du projet, nous vérifierons que ce boitier n'est pas une cause de danger, mais cela n'est pas à prendre en compte dans le cadre de ce contrôle.

limitons aux modèle de messages sur Twitter ou par Email. Dans tous les cas de modèles créatifs de messages, un intitulé est associé (par exemple, Retard), un identifiant pour la reconnaissance vocale (par exemple, Late), une icône graphique (par exemple, un fichier `.png` correspondant à une montre cassée), un contenu (par exemple, "je suis en retard") et l'ensemble des informations qui devront être associées au message parmi un ensemble prédéfini : itinéraire, position courante, destination, heure d'arrivée prévue, ... Par exemple, à un message correspondant à un Retard, la position courante et l'heure d'arrivée prévue devront être associées. La durée de tentative pour envoyer le message est aussi associée au modèle de messages (par exemple pour un Retard, 5mn, car après c'est inutile). Dans le cas d'un modèle de messages correspondant à des emails, l'adresse à laquelle envoyer les emails (par exemple "monPote@iut.fr") est également définie. Dans le cas d'un modèle de messages correspondant à des tweet, le compte Twitter à partir duquel les messages doivent partir doit être précisé (par exemple, @surLaRoute). Si le compte n'a pas encore été paramètré dans MyWaze, il peut l'être en cours de création du modèle de messages. Sur la route, lorsque le conducteur sélectionne le modèle de messages ainsi créé (par exemple Retard), le système envoie automatiquement le message contenant les informations associées (par exemple, dans le cas d'un Retard, un email : A : monPote@iut.fr ; sujet : retard ; contenu : "je suis en retard. Je suis à (Latitude : 43.58, Longitude : 7.11). Je dois arriver vers 19h20., date envoi : 19h05"). Cinq minutes plus tard le conducteur peut renvoyer un message et s'il a pu rouler, le contenu du message sera alors "je suis en retard. Je suis à (Latitude : 43.65, Longitude : 7.11). Je dois arriver vers 19h27., date envoi : 19h10". Si le système ne parvient pas à envoyer le message, le message passe alors dans un état EnAttente. Le système re-tente l'envoi pendant la durée prévue par le modèle du messages, en l'occurrence dans le cas d'un retard, 5mn. Lorsque l'envoi est réussi, le message est détruit. Lorsque l'envoi échoue, le message passe dans l'état Echec. Nous envisageons que les modèles de messages fassent l'objet de jeux dans différentes communautés : #voitureJauneCroisée, #NiceMonacoRecordDeLenteur, etc. Pour envoyer un message, le conducteur déjà identifié dispose d'une interface dédiée qui lui présente les différents modèles de messages possibles. Scénario : Le conducteur sélectionne le modèle de messages parmi les modèles connus de son boitier [3]. Le système construit le message,

3. Il peut s'agir de toucher l'écran ou d'énoncer le modèle de messages "Police" ou "Late" par exemple. On considère que la même "interface" (au sens Java) gère les deux

puis le lit. Le conducteur doit alors valider l'envoi. Le message est alors envoyé par le système. Si le système ne parvient pas à l'envoyer, il annonce le problème, puis re-essaie jusqu'à réussir ou que le délai associé à ce modèle de message soit dépassé. Dans ce cas, il avertit le conducteur que le message n'est pas parti par un signal sonore. Inversement dès que le message est envoyé, le conducteur est averti.

1 Questions

Les questions suivantes n'ont pas à être traitées dans l'ordre. Dessins au crayon à papier autorisé.

1.1 Diagramme des Cas d'Utilisation

Représentez acteurs et cas d'utilisation sur un diagramme.

1.2 Diagramme de Séquence

Représenter le diagramme de séquence correspondant au scénario.

1.3 Diagramme de Classes

Construire un diagramme de classes qui représente le système en intégrant les informations présentes dans l'ensemble des besoins exprimés précédemment.

1.4 Codage d'un diagramme de classe

En considérant le diagramme de classes de la figure 1 :

1. Justifiez la cardinalité 1 à partir des éléments présents dans le diagramme lui-même.
2. Donner la définition de la structure des classes correspondantes en java.

---

types de communication.

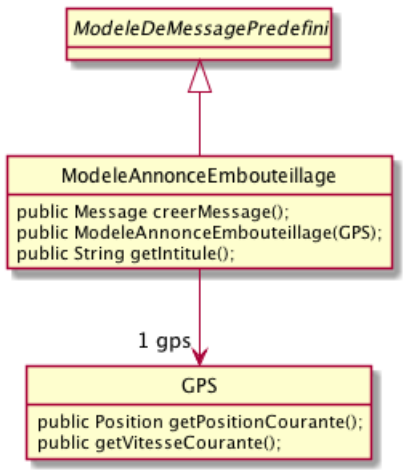

FIGURE 1 – Un diagramme de classe

1.5 Rétro-ingénierie d'un code Java

Compléter votre diagramme de classe (section 1.3) pour prendre en compte le code suivant [4].

```
public class FabriqueDeMessages extends Fabrique {
        public HashMap<String , ModeleDeMessage> modeles =
                                        new hashMap<String , ModeleDeMessage >()
        public Message creerMessage(String unModeleIntitule) {
                ModeleDeMessage mdm =  modeles.get(unModeleIntitule);
                if (mdm == null)
                        return null;
                else
                        return mdm.creerMessage();}
        public boolean envoyer(Message unMessage) {
                return unMessage.envoyer();}
        public boolean ajouterModeleDeMessage(ModeleDeMessage unModele){
                if (modeles.containsKey(unModele.getIntitule()) )
                                return false;
                else {      modeles.put(unModele.getIntitule(), unModele);
                            return true;} }}
```

4. N'inventez pas. Si vous n'avez pas d'information, modélisez uniquement les informations dont vous disposez.