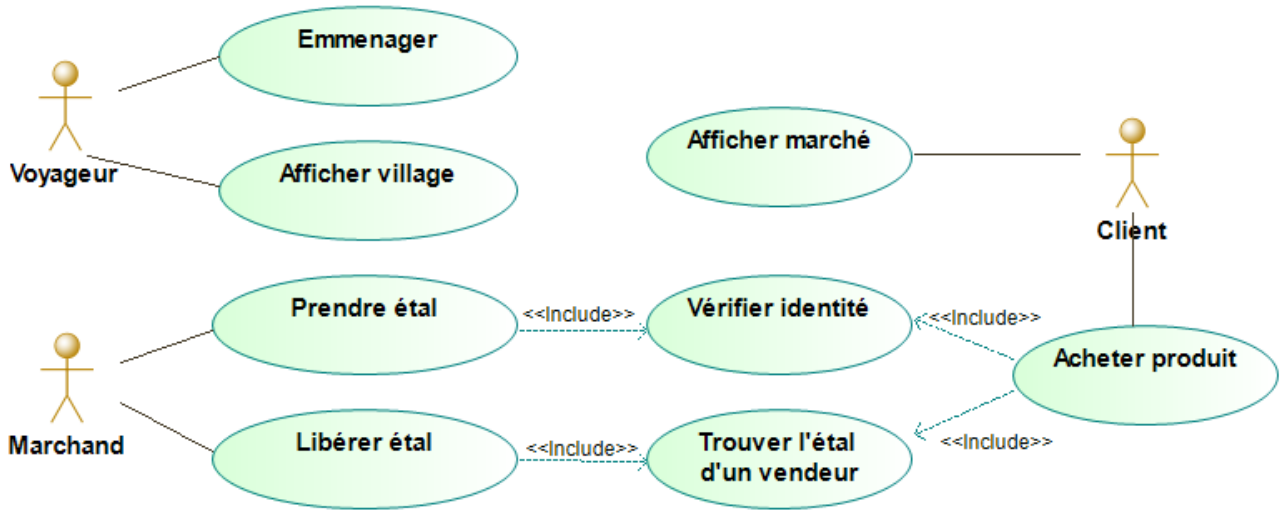
Architecture ECB

Sujet

Nous reprenons le sujet précédent, sauf que cette fois ci nous centrons notre conception point de vue utilisateur. Sur ce sujet nous nous concentrons sur les *contrôleurs* et les *frontières*, nous reviendrons sur les *entités* au TP suivant.

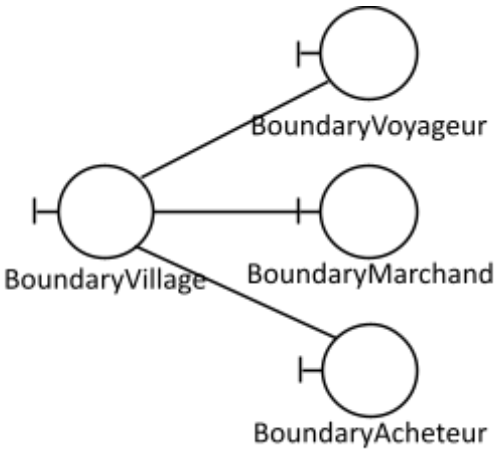


Le diagramme de cas présente les différentes fonctionnalités utilisées par chacun des acteurs. Nous allons adopter une architecture ECB.

Il nous manque encore deux points techniques qui limitent notre utilisation de cette architecture :

- le multi-thread qui nous permettrait de faire un boundary par acteur comme le préconise l'architecture ECB,
- le pattern Observateur qui permet de faire remonter des actions des entités vers les boundary.

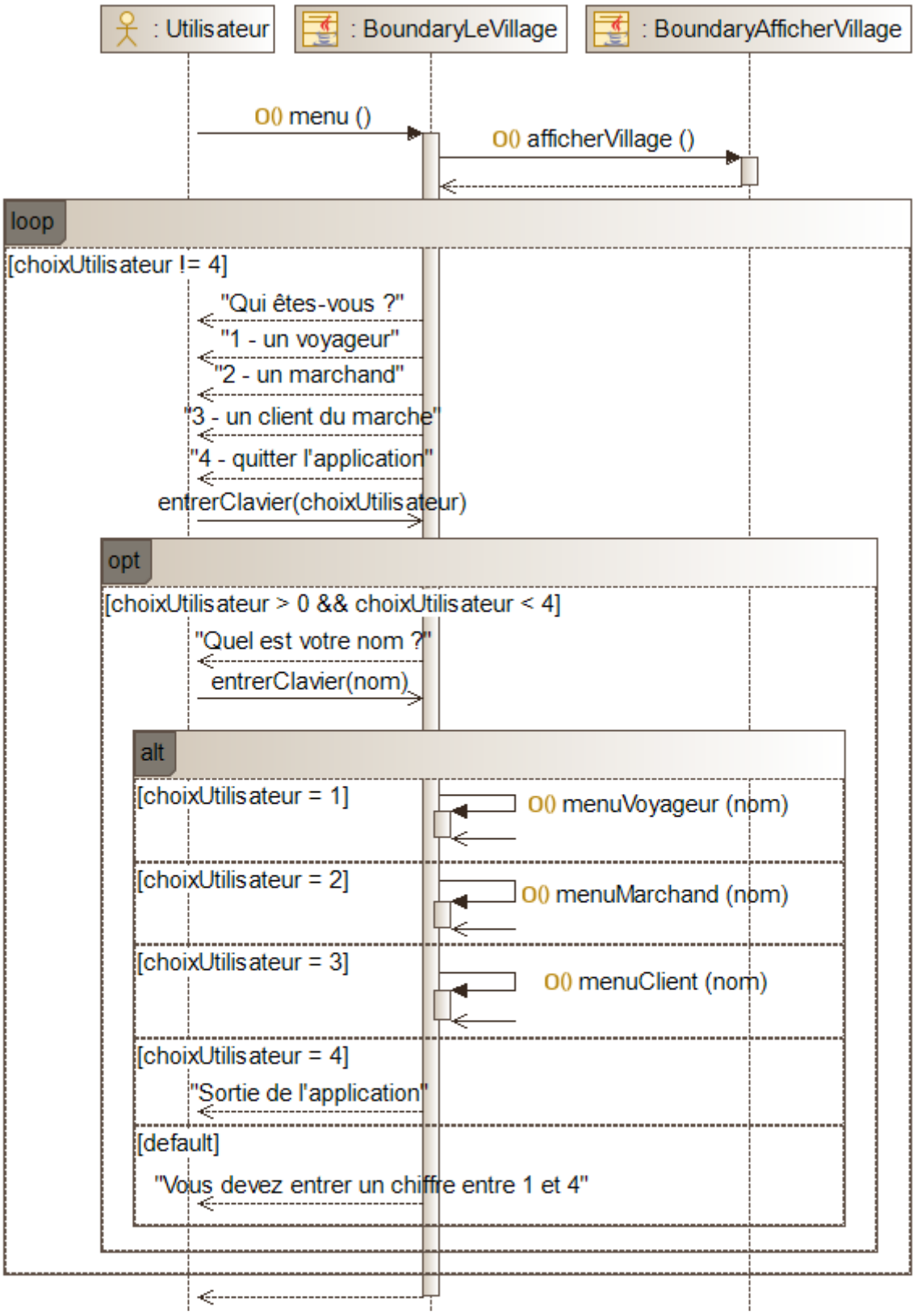
Pour pallier ce manque de technique et n'utiliser qu'un thread (le main), nous allons donc ajouter un boundary `BoundaryLeVillage` qui permettra d'indiquer si vous êtes un voyageur, un marchand ou un client.



A partir de ce boundary nous accèderons au menu de chacun des acteurs.

Pour le deuxième point, à chaque fois qu'un utilisateur voudra observer un changement dans le logiciel il devra le faire à partir d'une requête.

Diagramme de séquence détaillant la méthode menu permettant de lancer les menus des différents acteurs afin de pallier la non connaissance du multi-thread.



Extrait de la classe BoundaryLeVillage (correspondant à l'intérieur de la loop)

```
StringBuilder question = new StringBuilder();
      question.append("Qui êtes vous ?\n");
      question.append("1 - un voyageur\n");
      question.append("2 - un marchand\n");
      question.append("3 - un client du marche\n");
      question.append("4 - quitter l'application");

      choixUtilisateur = Clavier.entrerEntier(question.toString());

      if (choixUtilisateur > 0 && choixUtilisateur < 4) {
            System.out.println("Quel est votre nom ?");
            String nom = scan.next();
            switch (choixUtilisateur) {
            case 1:
                  menuVoyageur(nom);
                  break;
            case 2:
                  menuMarchand(nom);
                  break;
            case 3:
                  menuClient(nom);
                  break;
            case 4:
                  System.out.println("En revoir");
                  break;
            default:
                  System.out.println(
                              "Vous devez entrer un chiffre entre 1 et 4");
                  break;
            }
      }
```

Par contre quand vous utilisez en IHM la bibliothèque Swing vous êtes en multi-thread donc vous avez bien une frame pour chaque acteur.

Explication des différentes étapes

1) Aller sur Github et récupérer le projet ILU2-POO-TP2.

Ce source contient les paquetages contenant des entités :

- le paquetage `personnages` avec les classes :
  - `Personnage`,
  - `Gaulois`,
  - `Chef`,
  - `Druide`
- le paquetage `villagegaulois` avec les classes :
  - `Village`
  - `Etal`
- le paquetage `histoire` avec la classe `Scenario`.

Il contient aussi les deux paquetages frontière et contrôleur

- le paquetage `frontiere` avec les classes :
  - `BoundaryEmmenager`
  - `BoundaryAfficherVillage`
- le paquetage `controleur` avec les classes :
  - `ControlEmmenager`
  - `ControlAfficherVillage`

Les entités sont maintenant dépourvues de texte à afficher, la forme des sorties ne devant pas être décidée au niveau des *entités* mais au niveau des *frontières* afin de pouvoir les modifier selon les cas d'utilisation.

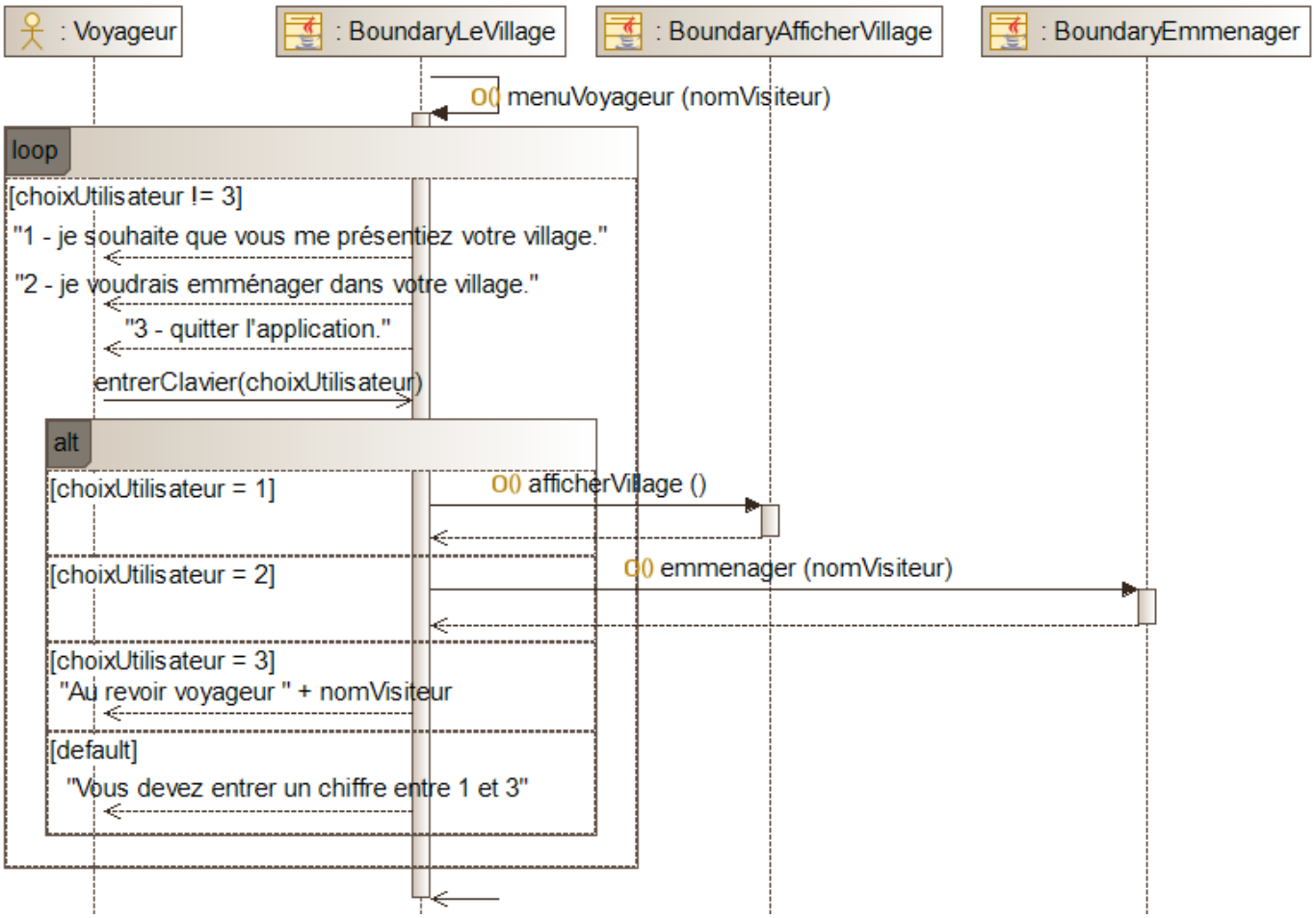
Les méthodes des *entités* retournent donc des données transformées en types primitifs (ou énumérés) par le *contrôleur* puis mis en forme par la *frontière* afin de les afficher dans la console.

2) Donner le code de chacun des cas d'utilisation du logiciel

Pour chaque cas d'utilisation vous aurez un document disponible sous Moodle qui contiendra :

- un extrait du diagramme de cas,
- le diagramme de classes participantes au cas étudié,
- le diagramme de séquence détaillé décrivant la méthode qui sera appelé afin de réaliser la fonctionnalité du cas étudié,
- le travail à réaliser sur ce cas d'utilisation.

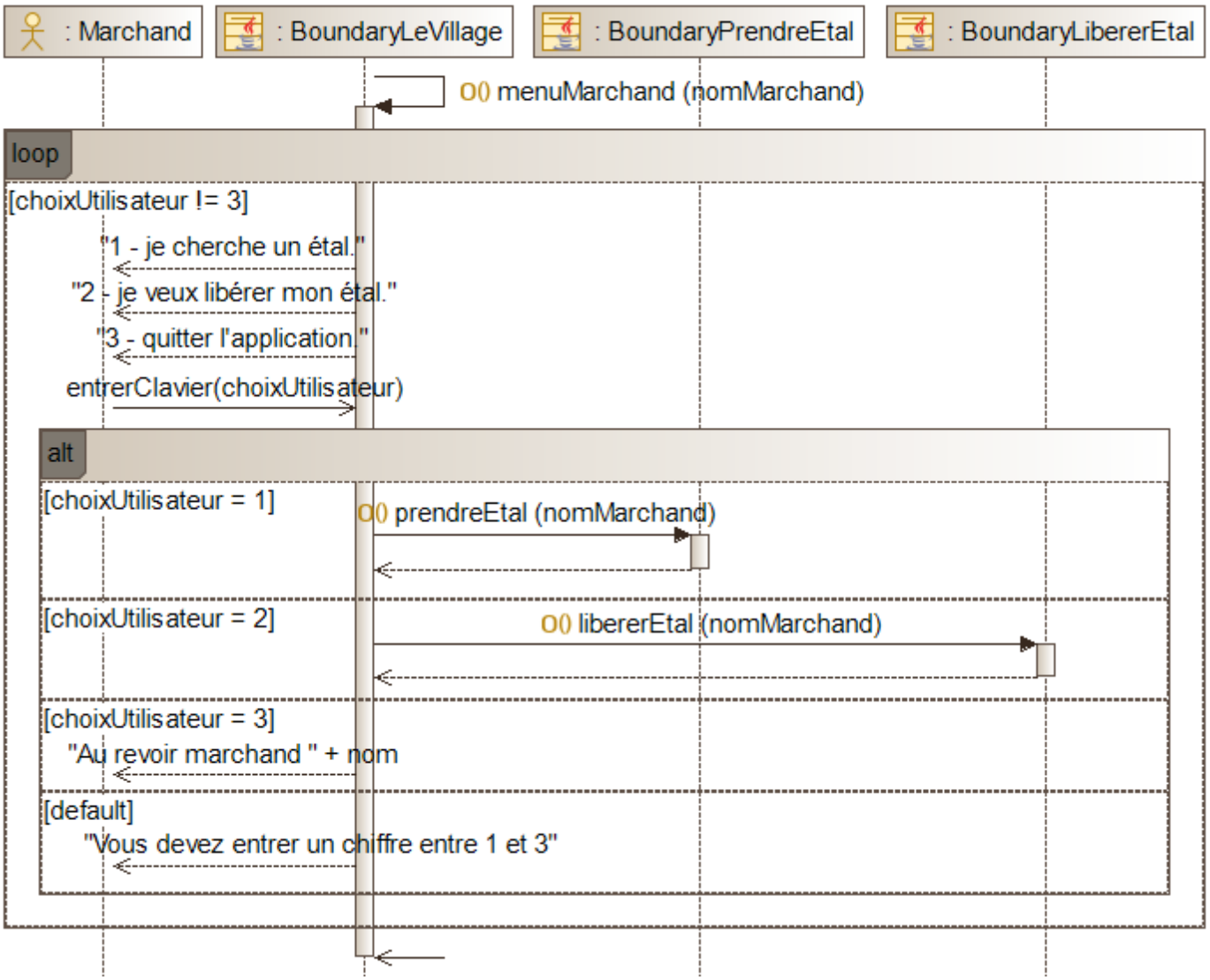
a) Nous commencerons par les fonctionnalités accessible au voyageur :



Cas d'utilisation pour un voyageur :

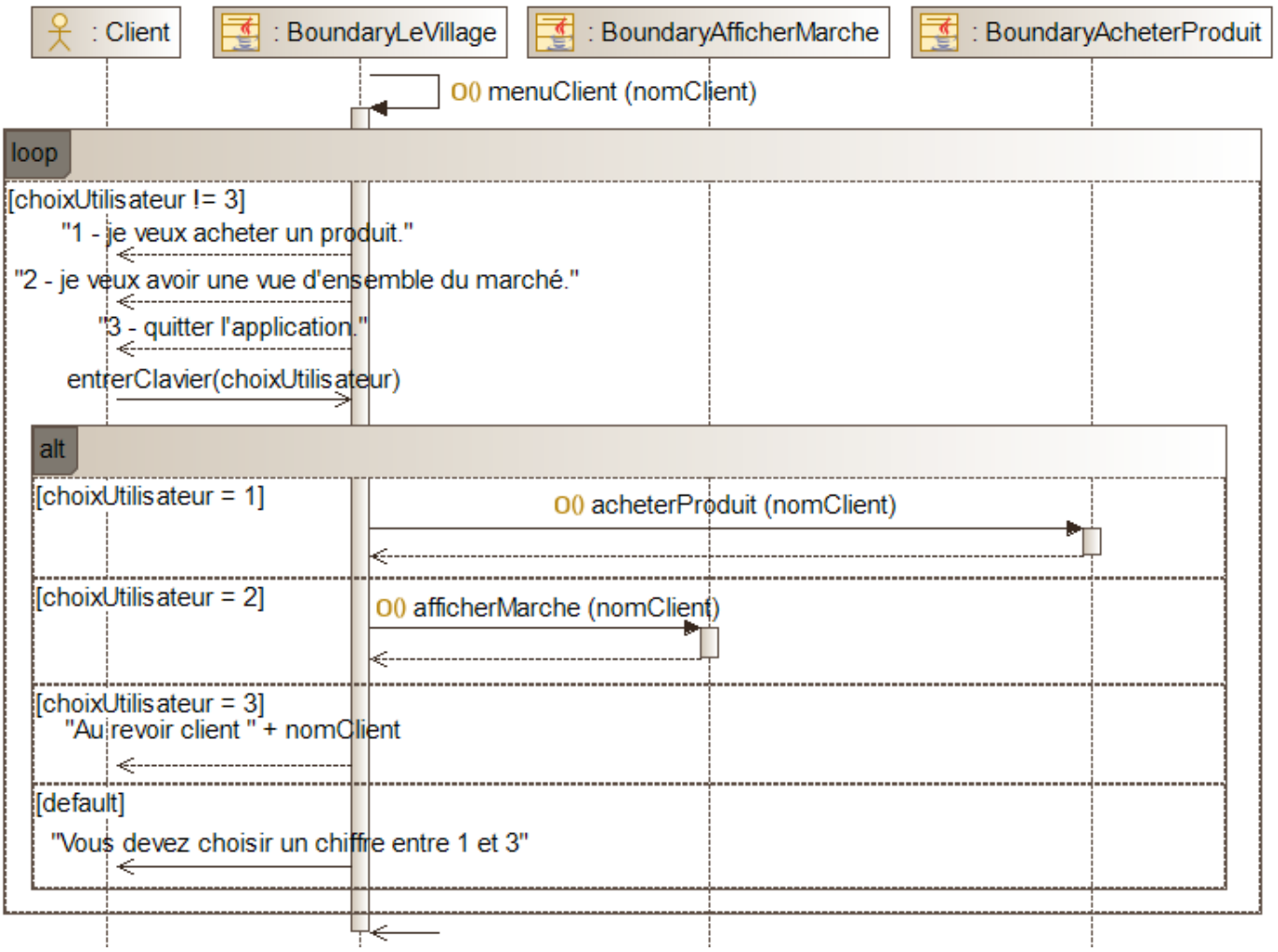
- "`afficherVillage`" (code déjà écrit, questionnaire à remplir sous Moodle),
- "`emmenager`" (code écrit en parti, à compléter à la question 2.2),

b) Nous continuons par les fonctionnalités accessible au marchand :



- Pour cela nous commencerons par implémenter le cas inclus "`verifierIdentite`". Ce code est utilisé dans deux cas : "`PrendreEtal`" et "`AcheterProduit`" ,
- implémenter le cas "`PrendreEtal`",
- implémenter le cas inclus "`trouverVendeur`". Ce code est utilisé dans deux cas : "`LibererEtal`" et "`AcheterProduit`" ,
- implémenter le cas "`LibererEtal`".

c) Nous finirons par les fonctionnalités accessible au client:



- implémenter le cas "`afficherMarche`",
- implémenter le cas "`acheterProduit`".