

Contrôle long – UML

13 décembre 2005
Durée : 3 heures

Aucun document n'est autorisé. Si vous êtes amenés à émettre des hypothèses, explicitez les sur la copie.
Les durées indiquées pour chaque exercice n'ont qu'une valeur indicative.

Exercice 1. Diagramme de cas d'utilisation et diagramme de séquence système (20+20 min)

Le déroulement normal d'utilisation de la caisse est le suivant :

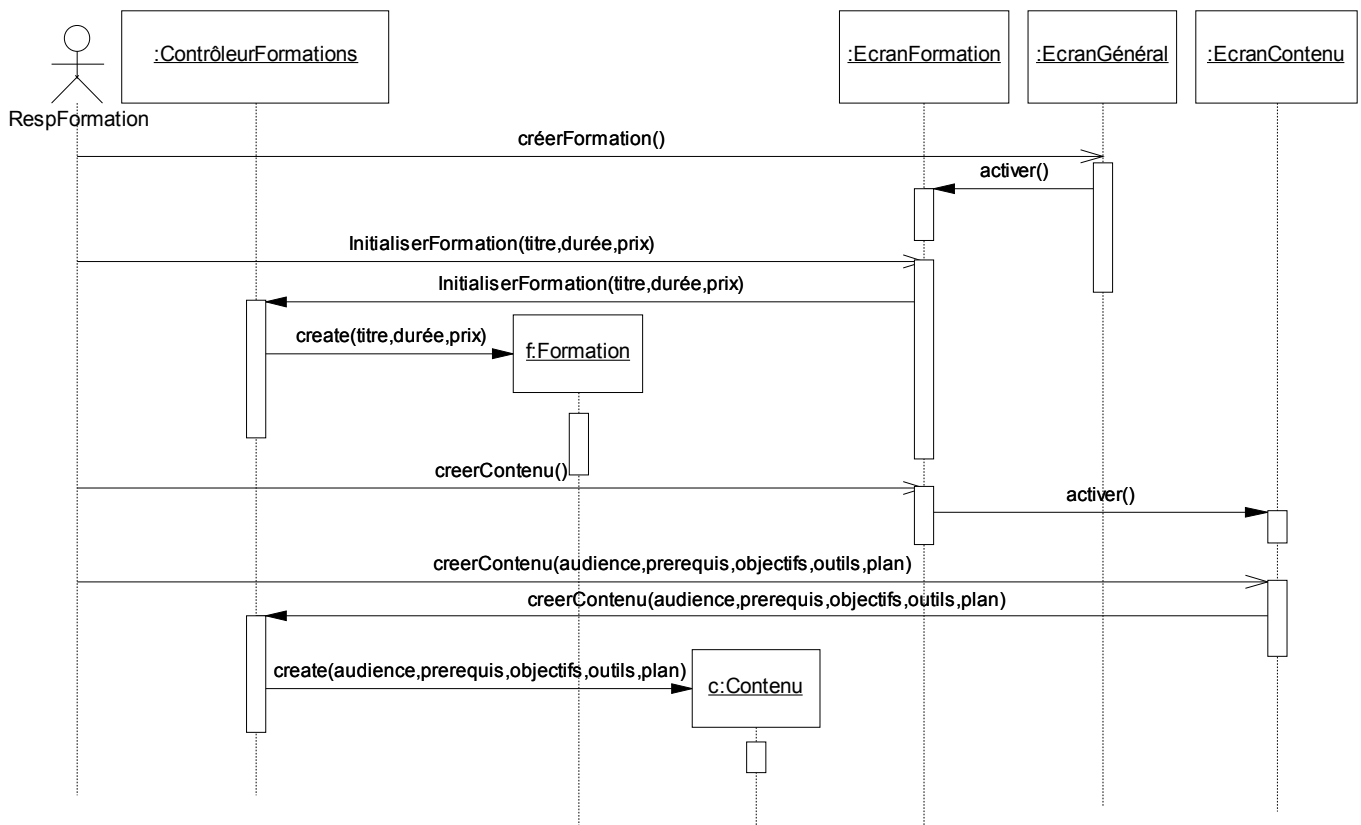
- un client arrive à la caisse avec des articles
- le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si celle-ci est supérieure à 1
- la caisse affiche le prix de chaque article et son libellé
- lorsque tous les articles ont été enregistrés, le caissier signale la fin de la vente
- la caisse affiche le total des achats
- le client choisit son mode de paiement :
 - liquide : le caissier encaisse l'argent et la caisse indique le montant éventuel à rendre au client
 - chèque : le caissier note l'identité du client et la caisse enregistre le montant sur le chèque
 - carte de crédit : un terminal bancaire fait partie de la caisse, il transmet la demande à un centre d'autorisation multi-banques
- la caisse enregistre la vente et imprime un ticket
- le caissier transmet le ticket imprimé au client

Un client peut présenter des coupons de réduction avant le paiement. Lorsque le paiement est terminé, la caisse transmet les informations relatives aux articles vendus au système de gestion des stocks. Tous les matins, le responsable du magasin initialise les caisses pour la journée.

- **Donnez un diagramme de cas d'utilisation pour la caisse enregistreuse.**
- **Donnez un diagramme de séquence système pour illustrer la manière dont les articles d'un client sont enregistrés** (peu importe le nombre de cas d'utilisation que ceci peut recouvrir).

Exercice 2. Diagramme de communication (15 min)

Donnez le diagramme de communication correspondant au diagramme de séquences ci-dessous.



Exercice 3. Diagramme de classes et OCL (15+20 min)

Question 1.

Etablissez le diagramme des classes correspondant au code source ci-dessous. Utilisez une classe d'association lorsque c'est possible et pertinent.

```

class Personne{
    private String nom, prenom;
    public final void setNom(String nom){this.nom = nom;}
    public final void setPrenom(String prenom){this.prenom = prenom;}
    public Personne(String nom, String prenom){setNom(nom); setPrenom(prenom);}
    public final void setNom(String nom) throws Exception{
        if (!(nom == null) || (!"".equals(nom))){this.nom = nom;}
        else throw new Exception("Chaine " + nom + " est vide ou null");
    }
    public boolean equals(Object autre){
        boolean b = false;
        if (autre instanceof personne){
            autre = (personne)Autre;
            b = getNom().equals(autre.getNom()) && getPrenom().equals(autre.getPrenom());
        }
        return b;
    }
}

class Musique{
    private Personne compositeur;
    public final Personne getCompositeur(){return compositeur;}
    public final void setCompositeur(personne compositeur){this.compositeur = compositeur;}
    private String titre;
    public final String getTitre(){return titre;}
}
    
```

```
public final void setTitre(){this.titre = titre;}
...
}

class Chanson extends Musique{
    private Personne auteur;
    public final Personne getAuteur(){return auteur;}
    public final void setAuteur(Personne auteur){this.auteur = auteur;}
    ...
}

class Interpretation{
    private Chanson la_chanson;
    public final Chanson getChanson(){return chanson;}
    public final void setChanson(Chanson la_chanson){this.la_chanson = la_chanson;}
    private Personne interprete;
    public final Personne getInterprete(){return interprete;}
    public final void setInterprete(Personne interprete){this.interprete = interprete;}
    ...
}
```

Question 2.

- Plutôt que de lever une exception dans certains cas comme le fait la méthode `Personne::setNom(...)` dans le code source présenté, on préfère en interdire l'exécution. **Utilisez OCL pour écrire cette contrainte. Complétez la contrainte pour définir complètement la méthode.**
- **Remplacer la première contrainte du point précédent par une autre, de type invariant. Est-ce tout à fait équivalent ?**
- **Ecrivez des contraintes OCL pour spécifier les points suivants :**
 - Une chanson ne peut pas avoir d'auteur si elle n'a pas de compositeur.
 - Une interprétation doit à la fois avoir un interprète et concerner une chanson.
- **Spécifiez le comportement de la méthode `Personne::equals(...)` en OCL**

Exercice 4. Diagramme de classes et d'objets (15+20+5 min)

On voudrait réaliser un jeu de bataille navale. Un jeu de bataille navale fait intervenir un tableau et un ensemble de bateaux, chaque bateau se composant d'un ensemble de taille fixe d'éléments. Un croiseur comprend 3 éléments, un escorteur 2 éléments et un sous-marin un seul. Chaque élément est caractérisé par sa position et par son état : « sauf » ou « touché ». Les sous-marins ont la possibilité de plonger ou de refaire surface. Lorsqu'ils plongent, ils ne peuvent plus être touchés. Les bateaux peuvent être placés horizontalement ou verticalement.

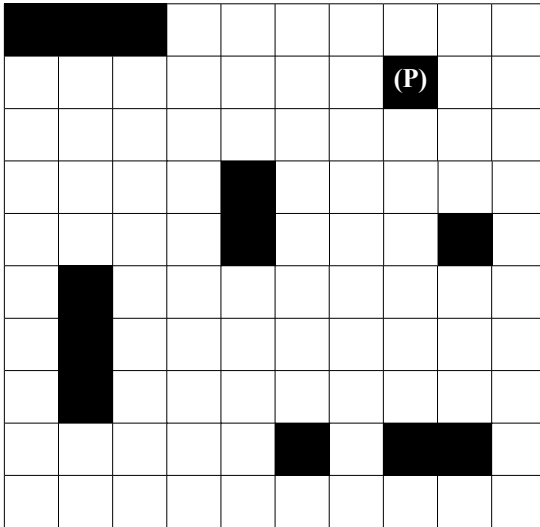
Une partie à deux se déroule de la manière suivante :

- Chaque joueur ajoute ses bateaux sur son tableau. Les bateaux peuvent être disposés horizontalement ou verticalement.
- Les joueurs attaquent à tour de rôle
- A chaque tour, le joueur attaquant désigne les coordonnées d'une cible sur le tableau adverse. Si les coordonnées sont celle d'un élément d'un bateau, alors l'élément est touché. Lorsque tous les éléments d'un bateau sont touchés, le bateau est coulé : il doit être supprimé du tableau. Lorsque tous les bateaux d'un joueur sont coulés, le joueur a perdu.

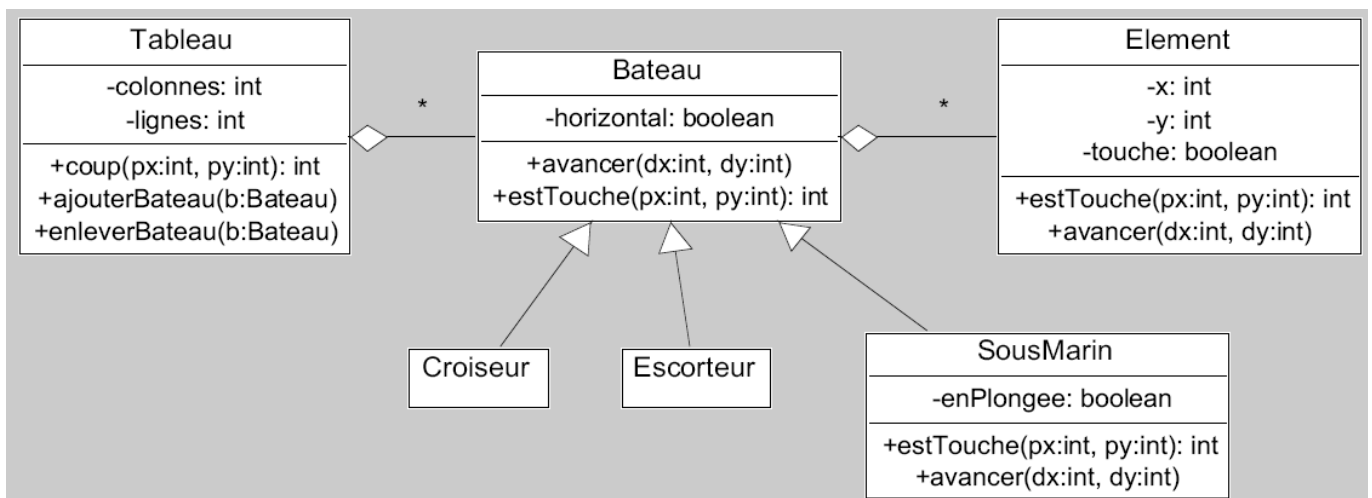
Au lieu de désigner une cible, un joueur peut entreprendre de faire plonger un sous-marin. Un sous marin ne peut rester que 4 tours en plongée. Passé ce délai, il est coulé automatiquement. Au lieu de faire plonger un sous marin, un joueur peut le faire remonter à la surface.

Au lieu de désigner une cible, un joueur peut aussi décider de déplacer un bateau dans une direction donnée. Un tel déplacement entraîne naturellement le déplacement de tous ses éléments. Le tableau est considéré comme « torique », c'est à dire que si l'un des éléments était amené, par un déplacement, à quitter le tableau, il apparaîtrait en fait de l'autre côté.

- **Donnez un diagramme d'objets correspondant au tableau ci-dessous (la marque « P » indique un sous-marin en plongée).**

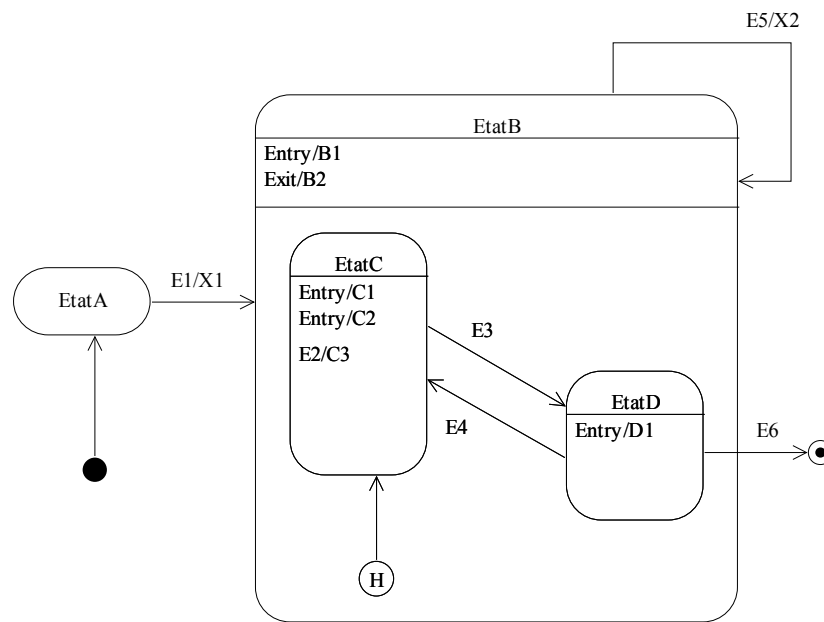


- **Donnez un diagramme de classes permettant de modéliser les objets intervenant dans une partie de bataille navale.** Le diagramme de classes devra être complet (attributs, opérations, multiplicités et navigabilité). On prendra garde à encapsuler les attributs par des méthodes judicieusement choisies.
- **Comment modéliseriez-vous le fait qu'un sous-marin en plongée ne peut pas être touché ?** Détaillez votre réponse.



Exercice 5. Diagramme d'états-transition (10 min)

Considérez le diagramme d'états suivant :



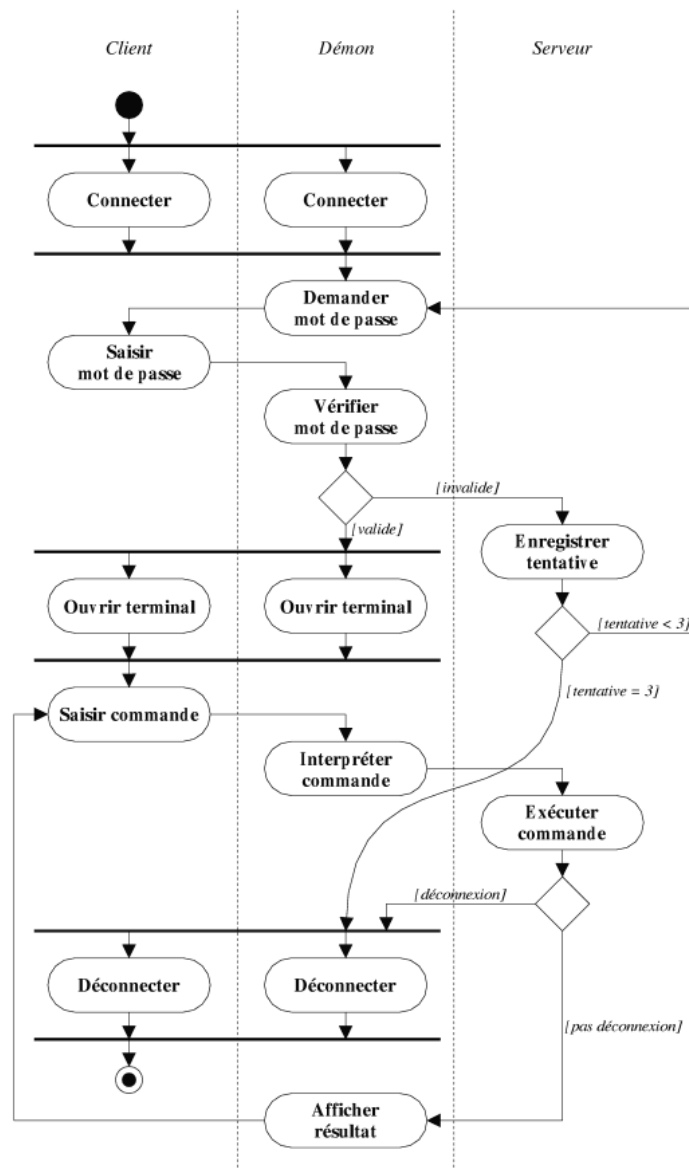
Quelles activités sont exécutées lorsque la séquence d'événements « E1, E2, E3, E5, E3, E4, E6 » se produit à partir de l'état A ? Pour répondre à cette question, on donnera les activités successivement produites par chaque événement dans un tableau comme celui ci-dessous :

<i>Evénement</i>	<i>Activités produites par l'événement</i>
E1	
E2	
E3	
E5	
E3	
E4	
E6	

Evénement	Activités produites par l'événement
E1	X1, B1, C1
E2	C3
E3	C2,D1
E5	B2,X2,B1,D1
E3	-
E4	C1
E6	-

Exercice 6. Diagramme d'activités (20+5 min)

- Décrivez la connexion d'un client à un serveur *telnet* en utilisant un diagramme d'activités. On considère trois protagonistes: le client, le démon *telnet* (i.e. le serveur logiciel) et la machine serveur. Une fois la connexion établie de part et d'autre du client et du démon, ce dernier demande un mot de passe au client qui dispose de trois tentatives avant que la connexion ne soit rompue. Les tentatives infructueuses sont enregistrées dans un fichier sur le serveur. Une fois l'identification faite, un terminal est ouvert conjointement par le démon et le client, et l'utilisateur peut alors saisir des commandes. Les commandes sont interprétées par le démon, exécutées sur le serveur et leur résultat est transmis au client par l'intermédiaire du démon. La commande *exit* déconnecte le client du serveur.



- Quel lien y a-t-il entre les diagrammes d'activités et les diagrammes d'états ?