Université Joseph Fourier Grenoble **Année 1996-97**

Deug Technologie Industrielle 2ème année
Examen de Programmation
Mercredi 29 janvier 1997

Durée 3 heures.
Tous documents autorisés
Les 2 exercices sont indépendants.
Le barême fourni est indicatif, il pourra être adapté lors de la correction.

Exercice 1 : Ecriture d'une classe simple - Tableaux - Création d'instances - Envois de messages
(barême indicatif 10 points)

On désire écrire une application éducative pour apprendre la notion d'ensemble à des enfants, et les initier aux notions d'appartenance, d'intersection, d'union et d'inclusion. Dans ce cadre on souhaite faire manipuler aux enfants des ensembles de chiffres (chiffres de `'0'` à `'9'`) sans répétition (un chiffre ne peut être présent qu'une seule fois).

Pour écrire cette application en langage JAVA, on définit donc une classe `EnsembleDeChiffres` qui permet de réprésenter et manipuler des ensembles de chiffres.

Un ensemble de chiffres est représenté par un tableau `present` de 10 booléens, `present[i]` ayant la valeur `true` si l'entier `i` appartient à l'ensemble et `false` sinon.

Par exemple l'ensemble {0,3,4,6,7} sera représenté par le tableau ci dessous

0	1	2	3	4	5	6	7	8	9
true	false	false	true	true	false	true	true	false	false

Les opérations définies sur un ensemble de chiffres sont : l'ajout d'un chiffre à l'ensemble, le retrait d'un chiffre à l'ensemble, la vérification qu'un chiffre est présent dans l'ensemble, le calcul de l'intersection de l'ensemble avec un autre ensemble, le calcul de l'union de l'ensemble avec un autre ensemble, le test de l'inclusion de l'ensemble dans un autre ensemble et l'affichage à l'écran de l'ensemble.

Le squelette de la classe `EnsembleDeChiffres` est donc le suivant :

```
public class EnsembleDeChiffres {

      private boolean[] present = new boolean[10];

      // ---------- constructeur -----------------------------------------------

      EnsembleDeChiffres()
      // initialise present de sorte que l'objet EnsembleDeChiffres créé corresponde à
      // l'ensemble vide.
      { ...
      }

      // ---------- méthodes ---------------------------------------------------

      public void ajoute(int i)
      // marque la présence du chiffre i dans l'ensemble
      // si i n'est pas dans l'intervalle 0..9, le programme affiche à l'écran le message d'erreur suivant :
      // ERREUR AJOUT, nombre pas dans intervalle 0..9
      { ...
      }
```

```
    public void retire(int i)
    // marque le chiffre i comme n'étant plus présent dans l'ensemble
    // si i n'est pas dans l'intervalle 0..9, le programme affiche à l'écran le message d'erreur suivant :
    // ERREUR RETRAIT, nombre pas dans intervalle 0..9
    { ...
    }

    public void tirerAuHasard()
    // initialise au hasard[1] l'ensemble (la présence ou non de chaque chiffre est tirée au hasard)
    { ...
    }

    public boolean appartient(int i)
    // retourne true si le chiffre i appartient à l'ensemble false sinon
    // si i n'est pas dans l'intervalle 0..9, le programme affiche à l'écran le message d'erreur suivant :
    // ERREUR APPARTIENT, nombre pas dans intervalle 0..9 et retourne la valeur false
    { ...
    }

    public EnsembleDeChiffres intersectionAvec(EnsembleDeChiffres e)
    // construit et retourne l'ensemble qui est l'intersection de l'ensemble et de l'ensemble e passé
    // en paramètre
    { ...
    }

    public EnsembleDeChiffres unionAvec(EnsembleDeChiffres e)
    // construit et retourne l'ensemble qui est l'union de l'ensemble avec l'ensemble e passé en
    // paramètre
    { ...
    }

    public boolean estInclusDans(EnsembleDeChiffres e)
    // retourne true si l'ensemble est inclus dans l'ensemble e passé en paramètre false sinon
    { ...
    }

    public void affiche()
    // affiche à l'écran l'ensemble. l'affichage de l'ensemble se fait sous la forme suivante :
    // les chiffres présents dans l'ensemble sont affichés séparés par des espaces. Une accolade
    // ouvrante ("{") précedant le premier chiffre, et une accolade fermante ("}") suivant le dernier
    // chiffre. Si l'ensemble est vide seules les accolades sont affichées
    // exemples : {} affichage produit pour un ensemble vide
    //            {1 4 5 6 } affichage produit pour un ensemble contenant les chiffres 1, 4 , 5 et 6
    { ...
    }

}
```

Questions :

Les deux questions sont indépendantes,elles peuvent être traitées dans n'importe quel ordre. En particulier vous pouvez répondre à la question 2) sans nécessairement avoir répondu à la question 1) (les spécifications de la classe `EnsembleDeChiffres` données ci-dessus sont suffisantes pour vous permettre de répondre à cette question).

---

[1] Pour cela on utilisera la méthode random() de la classe Math qui renvoie un double tiré de manière aléatoire dans l'intervalle [0, 1[. Pour tirer un booléen `b` au hasard (avec unechance sur deux d'obtenir la valeur `true`) on pourra écrire l'instruction `b=(Math.random() < 0.5)`

1) (barême indicatif 6 points)
Ecrire le corps du constructeur et de chacune des méthodes de la classe `EnsembleDeChiffres`.

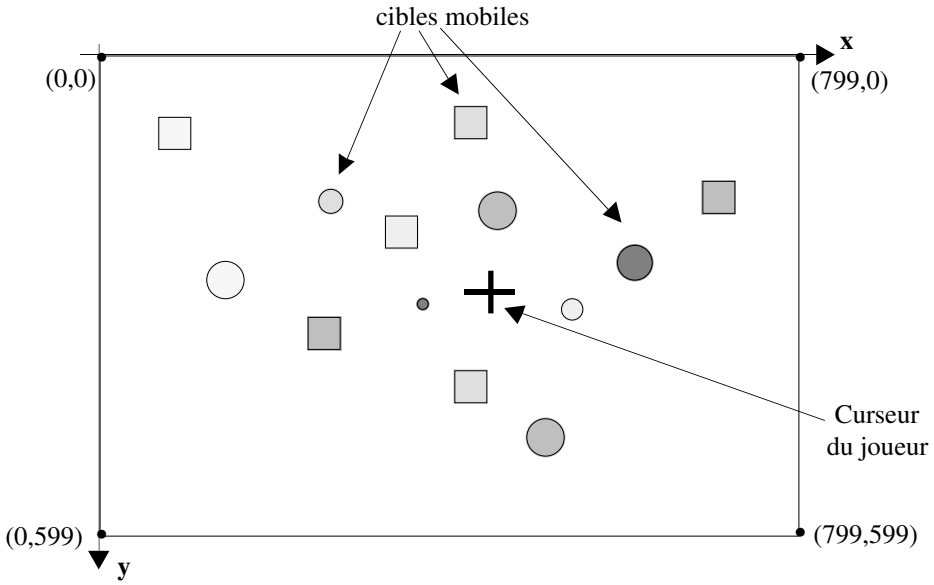
2) (barême indicatif 4 points)
Ecrire les instructions qui déclarent deux ensembles de chiffres `e1` et `e2`, initialisent `e1` et `e2` de manière aléatoire, les affichent, affichent leur union et leur intersection, indiquent ensuite si `e1` est inclus dans `e2` puis si `e2` est inclus dans `e1`.

exemple de trace d'exécution de ces instructions

```
e1 = {1 4 6 7 }
e2 = {1 2 3 4 6 7 8 }
l'intersection de e1 et e2 est : {1 4 6 7 }
l'union de e1 et e2 est : {1 2 3 4 6 7 8 }
e1 est inclus dans e2
e2 n'est pas inclus dans e1
```

Exercice 2 : Héritage
(barême indicatif 10 points)

Il s'agit d'écrire en JAVA une application de jeu simple. Le jeu, très simple, consiste à deplacer un curseur sur l'écran et à détruire le plus rapidement possible des cibles mobiles. Une cible est détruite lorsque le curseur est positionné sur celle ci et que le joueur actionne le tir (en enfonçant la touche espace par exemple).



Dans ce qui suit on ne s'intéressera qu'à la programmation des cibles.

L'aire de jeu est une fenêtre de 800 sur 600 points.

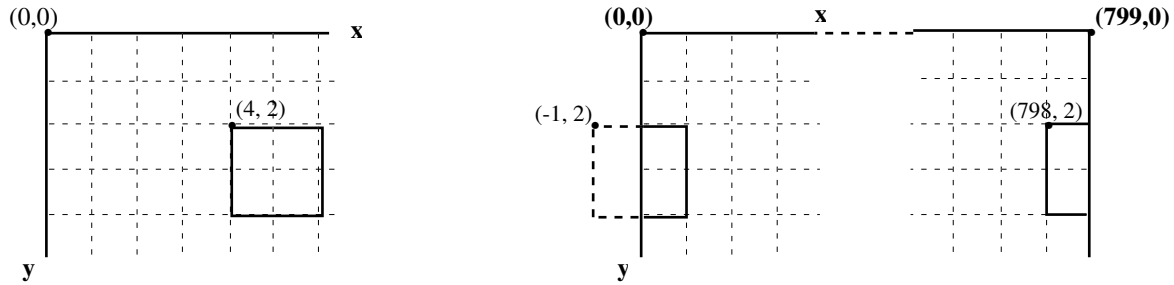
Un cible est définie par une position (couple de coordonnées entières) et une forme, qui peut être soit un cercle soit un carré. Si la cible est circulaire sa position donne le centre du cercle la matérialisant, si la cibleest carrée sa position indique les coordonnées de son coin supérieur gauche.

L'utilisateur peut déplacer le curseur de tir. Lorsque le curseur se trouve à l'intérieur d'une cible (carré ou cercle) et que l'utilisateur enclenche le tir la cible est atteinte.

A chaque cible est associée une énergie (un entier > 0), cette énergie est fixée à la création de la cible. Lorsque la cible est atteinte par un tir de l'utilisateur, son énergie diminue d'une unité. Lorsque l'énergie de la cible est nulle, celle-ci est détruite.

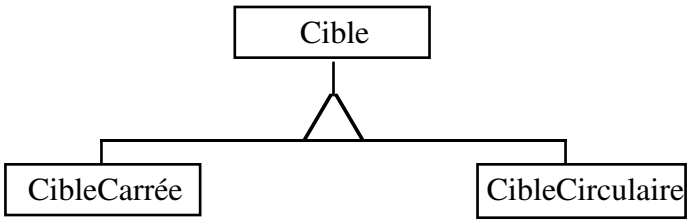
Le déplacement d'une cible s'effectue de manière aléatoire. A chaque tour de jeu, la poistion de la cible est déplacé de +5 ou -5 unités en x et en y.

Si lors d'un déplacement une cible sort de l'aire de jeu, elle ne disparait pas, mais réapparait sur le coté opposé de l'aire de jeu. Plus précisément, quand la position d'une cible sort de l'aire de jeu, cette position est ramenée à l'intérieur de l'aire de jeu du coté opposé au coté de sortie, et d'un nombre d'unités équivalent au nombre d'unités duquel elle est sortie.

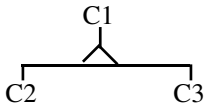


Exemple : déplacement du carré de position (4,2) de -5 en x et 0 en y. La nouvelle position devient (798,2), et la moitié du carré qui est sortie à gauche de l'aire de jeu réapparait sur le coté droit de celle-ci.

La hiérarchie de classes utilisée pour représenter les cibles est la suivante :



Rappel : L'héritage est représenté par le symbole ci-contre qui signifie que C2 et C3 héritent de C1



La classe `Cible`

La classe `Cible` définit ce qui est commun à toutes les cibles :

Les informations (attributs) associées à une cible sont :
- la position de la cible définie par un couple x,y de coordonnées entières.
- une énergie définie par un entier.

La position et l'énergie initiale d'une cible sont fixées à sa création (données en paramètre du constructeur).

Les opérations (méthodes) définies pour une cible sont :
- la possibilité de tester si un point (le curseur) est à l'intérieur ou non de la cible. Cette méthode prend en paramètres un couple de coordonnées entières et retourne une valeur booléenne (`true` si le point passé en paramètre est à l'intérieur de la cible, `false` sinon).
- la diminution de l'énergie de la cible, (dans le cas où la cible est atteinte par un tir de l'utilisateur, la cible pert un point d'énergie, c'est cette méthode qui devra alors être appelée).
- la consultation de l'énergie de la cible, cette méthode retourne un entier qui est le niveau d'énergie courant de la cible.

• le déplacement de la cible. Cette méthode modifie la position du centre de la cible de manière aléatoire (en ajoutant +5 ou -5 unités à l'abscisse et l'ordonnée du centre de la cible). Lorsque la cible sort de la fenêtre de jeu, elle est repositionnée à l'intérieur de celle-ci du coté opposé à celui où elle est sortie.
• une méthode pour afficher la cible.

La classe `CibleCarrée`

La classe `CibleCarrée` est une sous-classe de la classe `Cible`, elle enrichit la classe `Cible` d'une nouvelle variable d'instance :

• un entier qui défini la longueur du coté du carré matérialisant la cible.

La position d'une cible carrée (héritée de la classe `Cible)` définit le coin supérieur gauche du carré matérialisant la cible.

La taille du coté d'une cible carrée est fixée à sa création (donnée en paramètre du constructeur).

La classe `CibleCarrée` redéfinit les méthodes heritées de la classe `Cible` permettant de dessiner la cible et de tester l'appartenance d'un point à celle ci.

La classe `CibleCirculaire`

La classe `CibleCirculaire` est elle aussi une sous-classe de la classe `Cible`, elle enrichit la classe `Cible` d'une nouvelle variable d'instance :

• un entier qui définit le rayon du cercle matérialisant la cible.

La position d'une cible circulaire (héritée de la classe `Cible)` définit le centre du cercle matérialisant la cible.

La valeur du rayon initial d'une cible circulaire est fixée à sa création (donnée en paramètre du constructeur).

La classe `CibleCirculaire` redéfinit les méthodes heritées de la classe `Cible` permettant de dessiner la cible et de tester l'appartenance d'un point à celle ci.

Questions

1) (barême indicatif 3 points)
Ecrire le code JAVA de la classe `Cible` , définir ses variables d'instance (attributs), écrire son constructeur et ses différentes méthodes (à l'exception de la méthode destinée à l'affichage) en respectant les spécifications ci-dessus.

2) (barême indicatif 2,5 points)
Ecrire le code JAVA de la classe `CibleCarrée` , définir ses variables d'instance (attributs), écrire son constructeur et ses différentes méthodes (à l'exception de la méthode destinée à l'affichage) en respectant les spécifications ci-dessus.

3) (barême indicatif 2,5 points)
Ecrire le code JAVA de la classe `CibleCirculaire` , définir ses variables d'instance (attributs), écrire son constructeur et ses différentes méthodes (à l'exception de la méthode destinée à l'affichage) en respectant les spécifications ci-dessus.

4) (barême indicatif 2 points)
On souhaite enrichir le jeu d'un nouveau type de cibles : les cibles circulaires "variables" dont le comportement est identique à celui des objets de la classe `CibleCirculaire`, sauf que leur rayon varie lors des déplacements.

Le rayon varie dans un intervalle dont la borne inférieure et la borne supérieure sont données pour une cible circulaire variable donnée. Cette variation s'effectue de manière cyclique : le rayon diminue jusqu'à atteindre la valeur minimale puis grandi jusqu'à atteindre la valeur maximale puis diminue à nouveau, etc... A chaque déplacement la variation du rayon est donc de +1 ou -1 selon que la cible variable est dans une phase de croissance ou décroissance.

Les valeurs de rayon minimal, maximal et du rayon initial d'une cible circulaire variable sont fixées à sa création. Le sens de variation initial (croissant ou décroissant) est quand à lui tiré au hasard.

Ecrire le code JAVA d'une classe `CibleCirculaireVariable` en précisant :
a) le nom sa super classe (la classe dont elle hérite)
b) les variables d'instance qu'elle défini (on donnera leur type leur nom et en commentaire leur rôle),
c) son constructeur (on donnera ses paramètres et son corps)
d) la ou les méthodes qu'elle défini (ou redéfini ou enrichi) (pour chaque méthode on donnera sa signature (son nom et ses paramètres), un commentaire expliquant son fonctionnement, et les instructions de son corps).